CS Exit Ethiopia

Questions & Worked Answers

Computer Science Exit Examination

239 complete questions and 4 source variants

1 selected course | 18 November 2026

Selected courses

01. Advanced Database Systems

https://csexitethiopia.com

abelasnake2003@gmail.com | Telegram: @Abechu7

Course contents

Each course begins on a new page. PDF bookmarks provide question, study-note section and revision-topic navigation.

Advanced Database Systems 3

Reading the answers

Every question includes its answer, explanation, choice-by-choice contrasts and available evidence. Primary note proof quotes the study notes. Related reading and module passages are labelled separately; a module reference is not presented as primary note proof.

Advanced Database Systems

Questions & Worked Answers

239 questions + 4 source variants

Question 1 | Q00005

_____ is the process of selecting appropriate query execution strategy?

A. Query graph

B. Query optimization

C. Query tree

D. Query processing

Hint

Separate query representation, strategy selection and execution.

Answer B: Query optimization

Selecting an efficient execution strategy is optimization. A tree or graph represents the query; processing is the broader transformation and execution activity.

Why each choice is right or wrong

A - Incorrect: A graph corresponds to relational calculus, not strategy selection. (PDF pp. 27, 34.)

B - Correct: The module explicitly defines optimization as choosing an efficient strategy. (PDF pp. 27, 34.)

C - Incorrect: A query tree represents relational algebra and operations. (PDF pp. 27, 34.)

D - Incorrect: Processing includes decomposition and execution as well as optimization. (PDF pp. 27, 34.)

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB-S01

A query graph represents the predicate relationships of a relational-calculus expression without itself prescribing an operator execution order.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 27 | lines 7-13

What is Query Optimization?
✓ The activity of choosing a single —efficient‖ execution strategy (from hundreds) as determined by database catalog statistics.
✓ The aim is to answer which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?
✓ For each algebraic operator, what algorithm (of several available) do we use to compute that operator?

Fundamental of Database Systems module | PDF page 34 | lines 5-7

2. Apply heuristics rules to optimize the internal representation.
3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

Fundamental of Database Systems module | PDF page 34 | lines 17-27

• Query tree:
1. A tree data structure that corresponds to a relational algebra expression. It represents the input relations of the query as leaf nodes of the tree, and represents the relational algebra operations as internal nodes.
• An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.
• Query graph:
1. A graph data structure that corresponds to a relational calculus expression. It does not indicate an order on which operations to perform first. There is only a single graph corresponding to each query.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 5

Question 2 | Q00022

Which update-timing policy permits a transaction's changes to reach the persistent database before the transaction commits?

A. In-Place Update

B. Immediate update

C. Shadow update

D. Deferred update

Hint

Separate update timing from update location and follow the commit boundary.

Answer B: Immediate update

Immediate update permits a transaction's changes to be written to the persistent database before commit. If that transaction later aborts, recovery may need to undo those changes. Deferred update keeps uncommitted changes outside the persistent database. In-place update describes overwriting an existing location, while shadow update describes writing a separate version. Those location strategies do not, by their names alone, specify the timing policy requested here. Immediate update need not flush every cache write immediately.

Why each choice is right or wrong

A - Incorrect: Describes overwriting the existing location, not the commit-timing policy.

B - Correct: Correct: immediate update permits persistent database changes before commit and may require undo.

C - Incorrect: Describes writing a separate version while preserving the committed one, not the requested timing policy.

D - Incorrect: Deferred update excludes uncommitted changes from the persistent database.

Primary note proof

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

> Shadow/copy-on-write update writes a new location while preserving the committed version.

Study-note section ADB29

> In-place update overwrites the original location.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

> ◦ Deferred Update:
> • All transaction updates are recorded in the local workspace (cache)
> • All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
> • During commit the updates are first recorded on the log and then on the database
> • If a transaction fails before reaching its commit point undo is not needed

because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Fundamental of Database Systems module | PDF page 111 | lines 2-11

◦ Shadow update:
• The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.
• Multiple version of the same data item can be maintained
• Thus the old value (before image BFIM) and the new value (AFIM) are kept in the disk
• No need of Log for recovery
◦ In-place update: The disk version of the data item is overwritten by the cache version.
• Information for ARIES to accomplish recovery procedures includes:

Question sources

MOE Computer Science Exit Exam · 2016 · Question 22

Question 3 | Q00026

Which query-optimization approach applies relational-algebra properties and equivalences to rewrite a query and reduce intermediate work?

A. Database storage without query rewriting

B. Heuristic approach

C. Deterministic execution without query rewriting

D. Physical-plan cost estimation

Hint

Consider which equivalent transformation changes intermediate work and what assumptions preserve meaning.

Answer B: Heuristic approach

Heuristic optimization uses algebraic rules to reduce intermediate work, for example applying selective filters earlier, replacing a product plus join predicate with a join and projecting away unneeded columns. Storage and execution without rewriting do not perform those transformations. Cost estimation predicts resources for candidate plans rather than naming the algebraic rewrite approach itself. A practical optimizer can combine rewriting and costing, but the question isolates the rule-based transformation role.

Why each choice is right or wrong

A - Incorrect: Incorrect: storage without rewriting does not transform the algebraic query representation.

B - Correct: Correct: heuristic rewriting uses operator properties and equivalences to reduce intermediate work.

C - Incorrect: Incorrect: repeatable execution without rewriting does not apply the requested query transformations.

D - Incorrect: Incorrect: cost estimation predicts candidate-plan resource use rather than identifying the algebraic rewrite approach.

Primary note proof

Study-note section ADB18

> Heuristics reduce intermediate work: push selective row filters toward base tables; replace product followed by a join predicate with a join; project unneeded columns away while retaining join/filter columns; reorder inner joins when equivalence permits; reuse common expressions when worthwhile. Split conjunctive selections when each predicate can be applied earlier to its own input.

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 31 | lines 19-23

> • Heuristics Approach uses the knowledge of the characteristics of the relational algebra operations and the relationship between the operators to optimize the query.
> • Thus the heuristic approach of optimization will make use of:
> ◦ Properties of individual operators
> ◦ Association between operators

Question sources

MOE Computer Science Exit Exam · 2016 · Question 26

Question 4 | Q00046

In which phase of the two phase locking protocol can a transaction obtain locks?

A. Release phase

B. Roll back phase

C. Shrinking phase

D. Growing phase

Hint

Track the first lock release and the transaction-completion boundary.

Answer D: Growing phase

Two-phase locking acquires locks in the growing phase, then releases them in the shrinking phase without acquiring new ones.

Why each choice is right or wrong

A - Incorrect: Release is the unlocking portion, not acquisition. (PDF pp. 86.)

B - Incorrect: Rollback is not one of the two acquisition/release phases. (PDF pp. 86.)

C - Incorrect: Shrinking forbids new locks. (PDF pp. 86.)

D - Correct: Growing allows acquisition and forbids release. (PDF pp. 86.)

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB12

> Strict 2PL retains exclusive locks until commit/abort.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 86 | lines 16-24

> ◦ Locking (Growing) Phase:
> • A transaction applies locks (read or write) on desired data items one at a time.

• acquires all locks but cannot release any locks.
◦ Unlocking (Shrinking) Phase:
• A transaction unlocks its locked data items one at a time.
• Releases locks but cannot acquire any new locks.
● Requirement:
◦ For a transaction these two phases must be mutually exclusively, that is, during

Question sources

MOE Computer Science Exit Exam · 2016 · Question 46

Computer Science Exit Exam practice questions · Question 38

Question 5 | Q00053

Which one of the following is not problem if concurrency is not controlled properly?

A. Lost update problem

B. Consistent database

C. Incorrect summary

D. Dirty read

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Consistent database

Lost update, temporary update/dirty read, and incorrect summary are concurrency problems. A consistent database is the desired outcome of concurrency control.

Why each choice is right or wrong

A - Incorrect: Lost update is explicitly a concurrency anomaly. (PDF pp. 68, 70, 79.)

B - Correct: Consistency is a desired state, so this is the exception. (PDF pp. 68, 70, 79.)

C - Incorrect: Incorrect summary mixes values from different update stages. (PDF pp. 68, 70, 79.)

D - Incorrect: Dirty read exposes an update before its transaction has safely committed. (PDF pp. 68, 70, 79.)

Primary note proof

Study-note section ADB08

Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB06

Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure.

Study-note section ADB06

A transaction is a logical unit of database work, including read-only work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 68 | lines 4-7

i. The Lost Update Problem
◦ This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

ii. The Temporary Update (or Dirty Read) Problem
◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
iii. The Incorrect Summary Problem
◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Fundamental of Database Systems module | PDF page 79 | lines 4-15

● Transaction Processor is divided into:
◦ A concurrency-control manager, or scheduler, responsible for assuring isolation of transactions
◦ A logging and recovery manager, responsible for the durability of transactions.
● The scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed one-at-a-time.
1. Purpose of Concurrency Control
◦ To enforce Isolation (through mutual exclusion) among conflicting transactions.
◦ To preserve database consistency through consistency preserving execution of transactions.
◦ To resolve read-write and write-write conflicts.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 53

Question 6 | Q00078

Which one of the following is not a condition for two operations in a schedule to be said to conflict?

A. If they access the same item X

B. If they belong to the same transactions

C. If Write_item (X) is at least one of the operations

D. If they belong to different transactions

Hint

Distinguish transaction ordering, conflict conditions and full execution equivalence.

Answer B: If they belong to the same transactions

Conflict requires different transactions, the same item, and at least one write. Membership in the same transaction is therefore the exception.

Why each choice is right or wrong

A - Incorrect: Accessing the same item is required. (PDF pp. 70, 71.)

B - Correct: Same-transaction operations fail the different-transaction requirement. (PDF pp. 70, 71.)

C - Incorrect: At least one write is required. (PDF pp. 70, 71.)

D - Incorrect: Different transactions is required, so D is not the exception. (PDF pp. 70, 71.)

Primary note proof

Study-note section ADB09

> A serial schedule executes each transaction's operations consecutively. A serializable schedule is equivalent to some serial order; it may interleave transactions. Equality of one final value is insufficient evidence because different executions can accidentally coincide. Two operations conflict when they belong to different transactions, access the same item and at least one writes. Read/read pairs do not conflict.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

> ii. The Temporary Update (or Dirty Read) Problem
> ◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
> ◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
> iii. The Incorrect Summary Problem
> ◦ If one transaction is calculating an aggregate summary function on a number of

records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Fundamental of Database Systems module | PDF page 71 | lines 15-26

- Recoverable schedule:
  - One where no committed transaction needs to be rolled back.
  - A schedule S is recoverable if no transaction Tj in S commits until all transactions Ti that have written an item that Tj reads have committed. Examples,
- Sc: r1(X); w1(X); r2(X); r1(Y);w2(x);c2;a1; not recoverable
- Sd: r1(X); w1(X); r2(X); r1(Y); w2(X);w1(Y); c1; c2;
- Se: r1(X); w1(X); r2(X); r1(Y); w2(x) ; w1(Y); a1; a2;
- Cascadeless schedule:
  - One where every transaction reads only the items that are written by committed transactions. Eg.
- Sf: r1(X); w1(X); r1(Y); c1; r2(X); w2(X);w1(Y); c2;
- Strict Schedules:

Question sources

MOE Computer Science Exit Exam · 2016 · Question 78

Question 7 | Q00079

During restart recovery after a crash, which operation removes the database effects of a transaction that had not committed?

A. View

B. Commit

C. Rollback

D. Redo

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer C: Rollback

Atomicity requires removing the effects of an incomplete transaction after a crash. Rollback performs that undo work, restoring before-images as needed. Commit instead accepts successful transaction effects, and a view is a query representation rather than a recovery action. Redo installs logged after-images to recover missing work; it does not name the removal of uncommitted effects requested here. These actions occur during restart recovery after the system is operating again.

Why each choice is right or wrong

A - Incorrect: Incorrect: a view is not an operation that reverses transaction effects.

B - Incorrect: Incorrect: commit accepts transaction effects rather than undoing the incomplete work.

C - Correct: Correct: rollback undoes the effects of the uncommitted transaction.

D - Incorrect: Incorrect: redo installs logged after-images; the requested removal requires undo/rollback.

Primary note proof

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects. Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward.

Study-note section ADB25

> A view is a stored query definition, not encryption.

Related reading (context, not primary note proof)

Study-note section ADB06

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

> • Recovery from catastrophic
> • If there is extensive damage to the wide portion of the database
> • This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
> • Recovery from non-catastrophic failure
> • When the database is not physically damaged but has be come in consistent
> • The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
> • For instance,
> • If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
> • If transaction had not committed at failure time, recovery manager has to undo

(rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 79

Question 8 | Q00081

A system is in a _____ state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.

A. Idle

B. Waiting

C. Deadlock

D. Ready

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer C: Deadlock

Mutual waiting among transactions for locks held by one another is a deadlock condition.

Why each choice is right or wrong

A - Incorrect: Idle does not express mutual lock waiting. (PDF pp. 87.)

B - Incorrect: Waiting alone need not involve a cycle. (PDF pp. 87.)

C - Correct: Deadlock is mutual waiting that prevents progress. (PDF pp. 87.)

D - Incorrect: Ready does not describe transactions blocked on one another. (PDF pp. 87.)

Primary note proof

Study-note section ADB13

Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Study-note section ADB13

A timeout aborts long waits but may mistake slow progress for deadlock.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 87 | lines 2-6

Dealing with Deadlock and Starvation
- Deadlock
  - It is a state that may result when two or more transaction are each waiting for locks held by the other to be released
  - Example :

Question sources

MOE Computer Science Exit Exam · 2016 · Question 81

Question 9 | Q00113

Users may request permitted aggregates but must not infer protected individual values by subtracting overlapping aggregate answers. Which security measure specifically addresses this indirect-disclosure threat?

A. Inference control

B. Access control

C. Integrity control

D. Encryption

E. None of the above

Hint

Identify whether the scenario concerns identity, permission, disclosure or valid values.

Answer A: A. Inference control

Inference control targets learning protected individual facts from aggregate answers. For example, subtracting a sum excluding one person from a sum including that person can reveal their value even when their row is hidden. Ordinary access control grants or denies actions, integrity validates data and encryption protects encoded content. The specified indirect-disclosure threat identifies inference control specifically.

Why each choice is right or wrong

A - Correct: This directly prevents disclosure of individual facts inferred from allowed aggregates.

B - Incorrect: Authorization alone does not prevent inference from otherwise allowed answers.

C - Incorrect: Integrity constrains valid data rather than indirect disclosure.

D - Incorrect: Encryption does not stop an authorized user combining decrypted aggregate answers.

E - Incorrect: Inference control directly matches the threat.

Primary note proof

Study-note section ADB24

> Authentication establishes identity; authorization/access control determines permitted actions. Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions. A user who can view but cannot change a record has restricted access privileges. A view can restrict visible rows/columns, but its privilege and execution rules must support that restriction.

Study-note section ADB24

> Inference control concerns learning protected individual facts indirectly from allowed aggregate answers.

Study-note section ADB24

> Releasing SUM for a group and SUM for the same group excluding one known person may disclose that person's value by subtraction.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 52 | lines 22-26

> 2.2 Database Security
> - Views or restricted subschemas
> - Authorization rules to identify users and the actions they can perform
> - User-defined procedures (and rule systems) to define additional constraints or limitations in using the database

Fundamental of Database Systems module | PDF page 53 | lines 4-9

- Database security is about controlling access to information
- Some information should be available freely
- Other information should only be available to certain people or groups
- Many aspects to consider for security
- Legal issues
- Physical security

Question sources

Addis Ababa University mock exam · Question 15

Question 10 | Q00114

Which approach rewrites a query's internal representation using relational-algebra equivalences, without relying on integrity-constraint deductions or comparison of estimated plan costs?

A. Systematic plan-cost estimation

B. Semantic query optimization

C. Cost Estimation

D. Heuristic rules

E. Both A and C

Hint

Consider which equivalent transformation changes intermediate work and what assumptions preserve meaning.

Answer D: Heuristic rules

Heuristic rules rewrite algebraic representations using equivalences, such as applying suitable selections early or replacing a product followed by a join condition with a join. Semantic optimization additionally reasons from trusted constraints. Cost estimation predicts resources for candidate plans, and doing that systematically remains a costing activity. The stem isolates equivalence-based rewriting without those other bases, so D is correct and combining the two cost-estimation descriptions does not change the answer.

Why each choice is right or wrong

A - Incorrect: Incorrect: systematic cost estimation predicts plan costs rather than performing the specified equivalence-based rewriting.

B - Incorrect: Incorrect: semantic optimization additionally uses trusted constraints, which the stem excludes.

C - Incorrect: Incorrect: cost estimation compares resource predictions for candidate plans.

D - Correct: Correct: heuristic rules apply the stated relational-algebra equivalences.

E - Incorrect: Incorrect: A and C both describe costing, not the isolated algebraic-rewrite approach.

Primary note proof

Study-note section ADB18

> Heuristics reduce intermediate work: push selective row filters toward base tables; replace product followed by a join predicate with a join; project unneeded columns away while retaining join/filter columns; reorder inner joins when equivalence permits; reuse common expressions when worthwhile. Split conjunctive selections when each predicate can be applied earlier to its own input.

Study-note section ADB18

> Semantic optimization additionally uses trusted constraints, such as a key proving a join cannot multiply matching rows. Cost-based optimization estimates candidate plans instead of relying on rules alone.

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 34 | lines 5-7

> 2. Apply heuristics rules to optimize the internal representation.
> 3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

Fundamental of Database Systems module | PDF page 42 | lines 19-22

> ◦ Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate. (Compare to heuristic query optimization)
> • Issues

Question sources

Addis Ababa University mock exam · Question 16

Question 11 | Q00115

Which of the following is not true about locks?

A. Locks with large granularity are easier for the DBMS to administer.

B. Locks with small granularity cause more conflicts.

C. Locks with large granularity produce fewer details for the DBMS to track.

D. Locks may have a table-level granularity.

E. Locks may have a database-level granularity.

Hint

Consider conflicts, private work and validation costs, or lock-management overhead.

Answer B: Locks with small granularity cause more conflicts.

Fine granularity raises concurrency, whereas coarse granularity locks more data with fewer lock entries. Thus the claim that smaller units cause more conflicts reverses the tradeoff.

Why each choice is right or wrong

A - Incorrect: A single coarse lock can replace many fine locks and reduce management work. (PDF pp. 83, 95.)

B - Correct: Fine units allow more independent access, so this is the false claim. (PDF pp. 83, 95.)

C - Incorrect: Coarse locks reduce the number of entries tracked. (PDF pp. 83, 95.)

D - Incorrect: The file/relation granularity in the hierarchy permits table-level locking. (PDF pp. 83, 95.)

E - Incorrect: Entire-database locking is expressly listed. (PDF pp. 83, 95.)

Primary note proof

Study-note section ADB16

> Fine-grained row locks permit more independent operations but require tracking many locks. Coarse table/database locks simplify management but block unrelated rows and reduce concurrency. Intention locks advertise planned descendant locks in a hierarchy; escalation replaces many small locks with a coarser one. Small granularity does not inherently cause more logical conflicts. Choose granularity from contention and management overhead, not a universal rule.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 83 | lines 2-8

> • Lock Manager:
> • Managing locks on data
> items.
> • Lock table:
> • Lock manager uses it to store the identify of transaction locking a data
> item, the data item, lock mode and pointer to the next data item locked.
> One simple way to implement a lock table is through linked list.

Fundamental of Database Systems module | PDF page 95 | lines 14-28

> • A lockable unit of data defines its granularity
> • Granularity can be coarse (entire database) or it can be fine (an attribute of a relation).
> • Example of data item granularity:
> • A field of a database record
> • A database record
> • A disk block/ page

• An entire file
• The entire database
• Data item granularity significantly affects concurrency control performance.
• Thus, the degree of concurrency is low for coarse granularity and high for fine granularity.
• Example:
• A transaction that expects to access most of the pages in a file should probably set a lock on the entire file , rather than locking individual pages or records
• If a transaction that requires to access relatively few pages of the file , it is better

Question sources

Addis Ababa University mock exam · Question 17

Question 12 | Q00116

For every transaction T participating in the schedule, if all the operations of T are executed consecutively in the schedule. The schedule is

A. Cascaded schedule

B. Serial schedule

C. Cascadeless schedule

D. Recoverable schedule

E. None

Hint

Distinguish transaction ordering, conflict conditions and full execution equivalence.

Answer B: Serial schedule

Consecutive execution of every transaction's operations is the definition of a serial schedule. Recoverability and cascadelessness instead constrain dependencies on commits.

Why each choice is right or wrong

A - Incorrect: Cascading concerns dependent rollbacks, not contiguous execution. (PDF pp. 72, 71.)

B - Correct: Serial exactly matches the consecutive-operations definition. (PDF pp. 72, 71.)

C - Incorrect: Cascadeless means reading only committed writes. (PDF pp. 72, 71.)

D - Incorrect: Recoverable constrains commit order after a read dependency. (PDF pp. 72, 71.)

E - Incorrect: A listed definition does match, so None is wrong. (PDF pp. 72, 71.)

Primary note proof

Study-note section ADB09

A serial schedule executes each transaction's operations consecutively. A serializable schedule is equivalent to some serial order; it may interleave transactions. Equality of one final value is insufficient evidence because different executions can accidentally coincide. Two operations conflict when they belong to different transactions, access the same item and at least one writes. Read/read pairs do not conflict.

Study-note section ADB10

Recoverable means if Tj reads Ti's value, Tj cannot commit before Ti commits.

Study-note section ADB10

Cascadeless means a transaction reads only values written by committed transactions, avoiding cascading aborts.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 72 | lines 6-18

- Serial schedule:
  - A schedule S is serial if, for every transaction Ti participating in the schedule, all the operations of Ti are executed consecutively in the schedule. Otherwise, the schedule is called nonserial schedule.
  - For example, in the banking example suppose there are two transaction where one transaction calculate the interest on the account and another deposit some money into the account. hence the order of execution is important for the final result.
- Serializable schedule:
  - a schedule whose effect on any consistent database instance is identical to that of some complete serial schedule over the set of committed transactions in S.
  - A nonserial schedule S is serializable , is equivalent to say that it is correct to the result of one of the serial schedule .Example,

Fundamental of Database Systems module | PDF page 71 | lines 15-26

- Recoverable schedule:
  - One where no committed transaction needs to be rolled back.
  - A schedule S is recoverable if no transaction Tj in S commits until all transactions Ti that have written an item that Tj reads have committed. Examples,
- Sc: r1(X); w1(X); r2(X); r1(Y);w2(x);c2;a1; not recoverable
- Sd: r1(X); w1(X); r2(X); r1(Y); w2(X);w1(Y); c1; c2;
- Se: r1(X); w1(X); r2(X); r1(Y); w2(x) ; w1(Y); a1; a2;
- Cascadeless schedule:
  - One where every transaction reads only the items that are written by committed transactions. Eg.
- Sf: r1(X); w1(X); r1(Y); c1; r2(X); w2(X);w1(Y); c2;
- Strict Schedules:

Question sources

Addis Ababa University mock exam · Question 18

Question 13 | Q00117

A transaction property state about changes applied to the database by a committed transaction must persist in the database?

A. Isolation

B. Durability or permanency

C. Atomicity

D. Consistency preservation

E. None

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Durability or permanency

Persisting committed modifications despite subsequent failure is durability or permanency.

Why each choice is right or wrong

A - Incorrect: Isolation regulates visibility between transactions. (PDF pp. 66.)

B - Correct: Durability protects committed changes against loss. (PDF pp. 66.)

C - Incorrect: Atomicity is all-or-nothing execution. (PDF pp. 66.)

D - Incorrect: Consistency preserves valid states rather than persistence. (PDF pp. 66.)

E - Incorrect: Durability is a listed matching property. (PDF pp. 66.)

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

> • Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
> • Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
> • Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary

- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Addis Ababa University mock exam · Question 19

Question 14 | Q00118

Which update-timing policy keeps a transaction's uncommitted changes outside the persistent database until commit?

A. Undo

B. In-place modification

C. Immediate-modification

D. Deferred-modification

Hint

Separate update timing from update location and follow the commit boundary.

Answer D: Deferred-modification

Deferred modification keeps uncommitted updates outside the persistent database. If the transaction fails before commit, those database pages do not require undo for its changes; redo can still be needed after commit in a no-force system. Immediate modification permits precommit persistent changes. In-place modification describes overwriting the original location without specifying the commit timing, while undo is an action that reverses applied effects. D therefore names the requested timing policy.

Why each choice is right or wrong

A - Incorrect: Incorrect: undo reverses effects already applied rather than naming a policy postponing them.

B - Incorrect: Incorrect: in-place modification specifies overwriting location, not postponement until commit.

C - Incorrect: Incorrect: immediate modification permits persistent changes before commit.

D - Correct: Correct: deferred modification keeps uncommitted changes outside the persistent database.

Primary note proof

Study-note section ADB29

Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so

those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

In-place update overwrites the original location.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Question sources

Addis Ababa University mock exam · Question 20

Question 15 | Q00119

Which one of the following is false?

A. In object-oriented database, data are stored as collections of rows and tables

B. In relational database, data are perceived by users as tables

C. Object oriented database allows object identification and communication

D. Object oriented database allows reusability of objects

E. None of the above

Hint

Distinguish an object instance from its class, attribute values and persistent identity.

Answer A: The rows-and-tables claim is false for the object-oriented model.

An OODBMS primarily models persistent objects with identity, state and behavior, rather than relational rows and tables. Objects communicate through methods and object references, and inheritance supports reuse. Relational presentation as tables is the contrasting model. Therefore A is the false statement.

Why each choice is right or wrong

A - Correct: This is false: an OODBMS primarily models persistent objects rather than relational rows and tables.

B - Incorrect: Presenting relations as tables describes the relational contrast correctly.

C - Incorrect: Object identity and method-based collaboration are supported object-model concepts.

D - Incorrect: Inheritance and reusable object behavior support reuse, so this is not the false statement.

E - Incorrect: A is false, so none of the above cannot answer this negative question.

Primary note proof

Study-note section ADB01

> An object combines identity, state and behavior. A class specifies the attributes and operations of its instances; an object is one instance. The object identifier (OID) distinguishes objects independently of their attribute values and remains stable over that object's lifetime. Two objects can have equal values and different identities; changing a name does not create a new identity.

Study-note section ADB01

> It does not primarily present objects as rows and tables.

Study-note section ADB01

> An object-relational DBMS extends a relational system with richer types; ORM maps programming objects into relational tables.

Related reading (context, not primary note proof)

Study-note section ADB02

Study-note section ADB03

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 24 | lines 10-14

> Basic operations:
> Selection (σ) - Selects a subset of rows from relation.
> Projection (π) - Deletes unwanted columns from relation.
> Cross-product (x) - Allows us to combine two relations.
> Set-difference (-) - Tuples in relation 1, but not in relation 2.

Question sources

Addis Ababa University mock exam · Question 21

Question 16 | Q00296

27.----------is a collection of multiple logically related database distributed over a computer network.

A. Data Fragmentation

B. Distributed Database

C. Data Allocation

D. Data Replication

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer B: Distributed Database

A distributed database is a coordinated collection of logically related data across networked sites. Fragmentation divides a relation into row or column subsets; replication creates copies; allocation chooses the sites storing fragments or copies. These are design decisions within a distributed database rather than alternative names for the whole collection. Distribution does not require every fragment to be replicated, and the question's definition identifies the database as a whole.

Why each choice is right or wrong

A - Incorrect: Incorrect: fragmentation splits relations into subsets rather than naming the entire networked collection.

B - Correct: Correct: a distributed database is the coordinated logically related data collection across networked sites.

C - Incorrect: Incorrect: allocation assigns fragments or copies to sites; it is a placement decision.

D - Incorrect: Incorrect: replication creates copies and is a design choice rather than the definition of the collection.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system.

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 7-13

> • A distributed database (DDB) can be defined as :
> - A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
> - The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Fundamental of Database Systems module | PDF page 117 | lines 18-31

> I. Data Replication
> • The system maintain several identical copies of the relation & store each copy at a different site
> • In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
> II. Data Fragmentation
> - Split a relation into logically related and correct parts.
> - The main reasons for fragmenting a relation are

• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science model exam · Question 27

University of Gondar model exam · Question 27

Question 17 | Q00297

..... Means that a transaction must execute exactly once completely or not at all.

A. Consistency

B. Durability

C. Isolation

D. Atomicity

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Atomicity

Atomicity means a transaction is performed in its entirety or not performed at all. 'Exactly once' is interpreted here as all-or-nothing completion, rather than exactly-once delivery semantics.

Why each choice is right or wrong

A - Incorrect: Consistency concerns valid before/after states. (PDF pp. 66.)

B - Incorrect: Durability concerns survival after commit. (PDF pp. 66.)

C - Incorrect: Isolation concerns interaction with other transactions. (PDF pp. 66.)

D - Correct: Atomicity supplies the all-or-nothing rule. (PDF pp. 66.)

Primary note proof

Study-note section ADB06

A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Computer Science model exam · Question 28

University of Gondar model exam · Question 28

Question 18 | Q00300

Checkpoints are a part of

A. Recovery measures

B. Concurrency measures

C. Authorization measures

D. Security measures

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer A: Recovery measures

A checkpoint synchronizes database and log so recovery need not search the entire history. Its purpose is recovery work reduction.

Why each choice is right or wrong

A - Correct: Checkpointing is described in database recovery. (PDF pp. 105, 107.)

B - Incorrect: Locking and scheduling are concurrency controls; this checkpoint has a recovery role. (PDF pp. 105, 107.)

C - Incorrect: Privileges authorize access rather than synchronize recovery state. (PDF pp. 105, 107.)

D - Incorrect: Checkpointing does not authenticate users or grant privileges. (PDF pp. 105, 107.)

Primary note proof

Study-note section ADB30

A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB30

Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB30

Without checkpoints or another valid truncation boundary, long-running systems may need expensive log scans; deferred update does not avoid that problem.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Question sources

Computer Science model exam · Question 31

University of Gondar model exam · Question 31

Question 19 | Q00304

Which of the following is not a recovery technique?

A. ARIES Algorithm

B. Two-phase commit

C. Immediate update

D. Deferred update

Hint

Distinguish coordinated transaction outcome from concurrent access and local storage.

Answer B: Two-phase commit

Deferred update, immediate update, and ARIES are recovery approaches. Two-phase commit coordinates the distributed commit decision; it is the intended exception among the named recovery algorithms, though commit protocols also support distributed recovery.

Why each choice is right or wrong

A - Incorrect: ARIES is explicitly a recovery algorithm. (PDF pp. 107, 109, 123.)

B - Correct: Two-phase commit is a distributed commit approach, not the listed update-recovery algorithm. (PDF pp. 107, 109, 123.)

C - Incorrect: Immediate update appears under recovery techniques. (PDF pp. 107, 109, 123.)

D - Incorrect: Deferred update is a recovery scheme. (PDF pp. 107, 109, 123.)

Primary note proof

Study-note section ADB39

> Two-phase commit (2PC) first asks participants to prepare and durably promise they can commit. If all vote yes, the coordinator records commit durably and tells participants; otherwise it decides abort. A prepared participant must retain enough state to obey the final decision. Coordinator/communication failure can leave it in doubt and block progress until the decision is learned. Timeout after preparation is not permission to invent a commit or abort.

Study-note section ADB39

> 2PC coordinates atomic outcome; locking/other concurrency protocols provide isolation.

Study-note section ADB29

> Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Related reading (context, not primary note proof)

Study-note section ADB31

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Fundamental of Database Systems module | PDF page 109 | lines 13-22

iii. Shadow Paging
• Maintain two page tables during life of a transaction: current page and shadow page table.
• When transaction starts, two pages are the same.
• Shadow page table is never changed thereafter and is used to restore database in event of failure.
• During transaction, current page table records all updates to database.
• When transaction completes, current page table becomes shadow page table.
X and Y: Shadow copies of data items
X' and Y': Current copies of data items

Fundamental of Database Systems module | PDF page 123 | lines 23-29

i. Primary site technique: A single site is designated as a primary site which serves as a coordinator for transaction management.
• Transaction management:
- Concurrency control and commit are managed by this site.
- In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed.
- Advantages:

Question sources

Computer Science model exam · Question 35

University of Gondar model exam · Question 35

Question 20 | Q00305

Which of the following is not true about OODBMS (object-oriented database management system).

A. The key concepts is objects

B. The object is persistent

C. The key concept is relation

D. Object Identity is unchangeable

Hint

Distinguish an object instance from its class, attribute values and persistent identity.

Answer C: Relation is not the defining concept of an OODBMS.

The defining unit of an object database is an object with identity, state and behavior. ADB01 explains both persistence beyond program execution and identifier stability throughout an object lifetime. Relation is instead the characteristic abstraction of the relational model. Thus C is the one statement that fails to describe an OODBMS.

Why each choice is right or wrong

A - Incorrect: Objects are the defining units of the object-oriented database model.

B - Incorrect: An OODBMS provides persistence so objects survive the program execution that created them.

C - Correct: This is false for OODBMS: relation is the central abstraction of the relational model.

D - Incorrect: The object identifier remains stable throughout the lifetime of its object.

Primary note proof

Study-note section ADB01

> An object combines identity, state and behavior. A class specifies the attributes and operations of its instances; an object is one instance. The object identifier (OID) distinguishes objects independently of their attribute values and remains stable over that object's lifetime. Two objects can have equal values and different identities; changing a name does not create a new identity.

Study-note section ADB01

> Persistence lets an object survive beyond the program execution that created it.

Study-note section ADB01

> Complex CAD or multimedia structures can suit an object model, but many-to-many relationships alone do not require abandoning relational design.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 119 | lines 11-19

• Heterogeneous
- At least one of the database must be from different vendor : two variants
- Federated: Each site may run different database system but the data access is managed through a single conceptual schema.
• This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.
- Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.
Site 2

Question sources

Computer Science model exam · Question 36

University of Gondar model exam · Question 36

Question 21 | Q00306

Which optimization approach uses algebraic rules, such as pushing suitable selections before joins, to choose an operation order without exhaustively enumerating plans or comparing estimated costs?

A. Semantic query optimization based on trusted constraints

B. Heuristic- based

C. Cost estimation

D. Exhaustive search

Hint

Consider which equivalent transformation changes intermediate work and what assumptions preserve meaning.

Answer B: Heuristic- based

Heuristic optimization orders or rewrites operations using algebraic rules intended to reduce intermediate work. Applying suitable selections early is a typical example. Semantic optimization additionally uses trusted constraints, cost estimation predicts resource usage for candidate strategies, and exhaustive search enumerates alternatives rather than selecting a small set through rules alone. Although practical optimizers can combine these methods, the stated stem asks specifically for rule-based algebraic ordering, making B unique.

Why each choice is right or wrong

A - Incorrect: Incorrect: this alternative bases deductions on trusted constraints rather than the isolated algebraic ordering rules.

B - Correct: Correct: heuristic optimization orders operations through algebraic rules such as early selection.

C - Incorrect: Incorrect: cost estimation predicts candidate-plan resources instead of the specified rule-only choice.

D - Incorrect: Incorrect: exhaustive search enumerates alternatives, which the stem explicitly excludes.

Primary note proof

Study-note section ADB18

> Heuristics reduce intermediate work: push selective row filters toward base tables; replace product followed by a join predicate with a join; project unneeded columns away while retaining join/filter columns; reorder inner joins when equivalence permits; reuse common expressions when worthwhile. Split conjunctive selections when each predicate can be applied earlier to its own input.

Study-note section ADB18

> Semantic optimization additionally uses trusted constraints, such as a key proving a join cannot multiply matching rows. Cost-based optimization estimates candidate plans instead of relying on rules alone.

Related reading (context, not primary note proof)

Study-note section ADB19

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 31 | lines 19-23

> • Heuristics Approach uses the knowledge of the characteristics of the relational algebra operations and the relationship between the operators to optimize the query.
> • Thus the heuristic approach of optimization will make use of:
> ◦ Properties of individual operators
> ◦ Association between operators

Fundamental of Database Systems module | PDF page 34 | lines 5-7

> 2. Apply heuristics rules to optimize the internal representation.
> 3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

Question sources

Computer Science model exam · Question 37

University of Gondar model exam · Question 37

Question 22 | Q00533

A transaction in which either all of the database actions occur or none of them does is called

A. Atomicity

B. Consistency

C. Isolation

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer A: Atomicity

All database actions occurring or none occurring is atomicity, the first ACID property.

Why each choice is right or wrong

A - Correct: Atomicity is the entire-or-none requirement. (PDF pp. 66.)

B - Incorrect: Consistency is preservation of valid states. (PDF pp. 66.)

C - Incorrect: Isolation separates concurrent effects. (PDF pp. 66.)

D - Incorrect: Durability preserves completed commits. (PDF pp. 66.)

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

> • Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
> • Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
> • Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
> • Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.
> Example

Question sources

Computer Science model exam · 2015 · Question 2

Question 23 | Q00540

A transaction is still active when one of its changes is written into the persistent database. Which update-timing policy permits this?

A. Deferred-Update

B. In-place update

C. Immediate-Update

D. Undo

Hint

Separate update timing from update location and follow the commit boundary.

Answer C: Immediate-Update

Immediate update permits persistent database changes while the transaction is still active, before commit. An abort may therefore require undo. Deferred update keeps uncommitted changes outside the persistent database. In-place update describes overwriting a location rather than the commit-timing policy, and undo is the reversal of applied effects. The question concerns which timing policy permits the observed precommit write, not whether all immediate-update systems must flush every cache change immediately.

Why each choice is right or wrong

A - Incorrect: Incorrect: deferred update excludes uncommitted changes from the persistent database.

B - Incorrect: Incorrect: in-place update describes overwriting the existing location rather than the commit-timing policy.

C - Correct: Correct: immediate update permits persistent changes before commit.

D - Incorrect: Incorrect: undo reverses effects; it is not the update-timing policy.

Primary note proof

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

> In-place update overwrites the original location.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Question sources

Computer Science model exam · 2015 · Question 9

Question 24 | Q00556

Which of the following occurs when a transaction cannot proceed for an indefinite period while other transactions in the system continue normally?

A. Starvation

B. Deadlock

C. checkpoint

D. lock point

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer A: Starvation

Starvation means indefinite postponement of a transaction while other transactions continue. It can result from repeated waiting or repeated victim selection without any waiting cycle. Deadlock instead involves a cycle of mutually blocked transactions. A checkpoint records recovery state, and the lock point in two-phase locking is the instant a transaction acquires its final lock. Neither is the name of indefinite unfair postponement, so A remains the unique answer.

Why each choice is right or wrong

A - Correct: Correct: indefinite postponement while others proceed is starvation.

B - Incorrect: Incorrect: deadlock requires a waiting cycle, which is not required for starvation.

C - Incorrect: Incorrect: a checkpoint records durable recovery state; it is not an indefinite waiting condition.

D - Incorrect: Incorrect: the lock point is the instant the transaction acquires its final lock in 2PL.

Primary note proof

Study-note section ADB13

> Fixed resource ordering and conservative 2PL prevent cycles. A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp. A timeout aborts long waits but may mistake slow progress for deadlock. Starvation is indefinite postponement while other work proceeds, without a required cycle. Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Study-note section ADB12

> The lock point is the instant the final lock is obtained.

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 90 | lines 12-21

◦ Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further while other transaction continue normally
◦ This may occur , if the waiting method for item locking:
• Gave priority for some transaction over others
• Problem in Victim selection algorithm- it is possible that the same transaction may consistently be selected as victim and rolled-back .example In Wound-Wait
◦ Solution
• FIFO
• Allow for transaction that wait for a longer time

Question sources

Computer Science model exam · 2015 · Question 25

Question 25 | Q00565

The atom constructor can represent the following basic atomic values except.

A. Integers

B. Real Numbers

C. Strings

D. Array

Hint

Classify the shape of a value separately from an operation that creates an instance.

Answer D: Array is not an atom constructor.

ADB04 lists integer, real, Boolean and string as examples of basic atomic values. An array instead supplies indexed positions and is a collection constructor, even when a programming variable holds it as one value. Consequently integers, reals and strings fit the atom category while an array is the requested exception.

Why each choice is right or wrong

A - Incorrect: An integer is explicitly listed as a basic atomic value.

B - Incorrect: A real number is explicitly included among atomic values.

C - Incorrect: A string is a basic atomic value under the object-model convention used here.

D - Correct: An array contains indexed positions and is a collection constructor rather than an atomic one.

Primary note proof

Study-note section ADB04

> In a common object model, an object is described by (OID,type constructor,state). An atom is one basic value, such as an integer, real, Boolean or string. A tuple constructor groups named component values into one structured record, such as (Street:'A',City:'B'). It is not a collection of records.

Study-note section ADB04

> An array is not an atomic constructor merely because stored as one programming variable.

Study-note section ADB04

> A set is an unordered collection without duplicates; a bag allows duplicates; a list has order; an array provides indexed positions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 24 | lines 10-14

> Basic operations:
> Selection (σ) - Selects a subset of rows from relation.
> Projection (π) - Deletes unwanted columns from relation.
> Cross-product (x) - Allows us to combine two relations.
> Set-difference (-) - Tuples in relation 1, but not in relation 2.

Question sources

Computer Science model exam · 2015 · Question 34

Question 26 | Q00584

A system crashes again during ARIES recovery. Durable WAL, pageLSNs and compensation log records survive; storage is intact. No checkpoints were taken. Which statement correctly describes the second restart?

A. The same undo and redo list will be used while recovering again

B. The database will become inconsistent

C. Durable compensation records and pageLSNs preserve recovery progress, so already-installed actions can be skipped or safely redone as required.

D. The system cannot any further

Hint

Trace durable recovery metadata and what progress survives a second crash.

Answer C: C. Durable compensation records and pageLSNs preserve recovery progress, so already-installed actions can be skipped or safely redone as required.

ARIES records recovery progress in durable metadata. PageLSNs help determine whether a redo action is already installed, and compensation records preserve undo progress and its continuation. A second restart can therefore recover safely without assuming that the same undo and redo work lists recur. Lack of checkpoints may increase scanning work, but it does not erase this progress or make further recovery impossible.

Why each choice is right or wrong

A - Incorrect: Durable progress can change the required work, so identical lists are not guaranteed.

B - Incorrect: Restartable recovery does not make inconsistency inevitable.

C - Correct: PageLSNs and durable CLRs support safe continuation after repeated crashes.

D - Incorrect: ARIES can restart using surviving durable recovery metadata.

Primary note proof

Study-note section ADB31

> ARIES uses WAL, page log sequence numbers (pageLSN), a transaction table and a dirty-page table. Analysis reconstructs active/loser transactions and dirty-page information. Redo repeats history, starting from the relevant earliest dirty-page update; pageLSNs help skip already-installed actions. Undo reverses loser transactions.

Study-note section ADB31

> This makes recovery restartable: after a second crash, durable undo progress need not be undone again as if no progress occurred.

Study-note section ADB31

> Compensation log records (CLRs) describe undo actions and their continuation information.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 108 | lines 24-29

> • During recovery, all transactions of the commit table are redone and all transactions of active tables are ignored since none of their AFIMs reached the database.
> • It is possible that a commit table transaction may be redone twice but this does not create any inconsistency because of a redone is —idempotent‖,
> • that is, one redone for an AFIM is equivalent to multiple redone for the same AFIM.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Question sources

Computer Science model exam · 2015 · Question 53

Question 27 | Q00590

The rigorous two-phase locking protocol permits releasing all locks at the_____

A. Beginning of the transaction

B. During the execution of transaction

C. End of transaction

D. Never in the life-time of transaction

Hint

Track the first lock release and the transaction-completion boundary.

Answer C: At transaction end, on commit or abort.

Rigorous 2PL retains both shared and exclusive locks until commit or abort, unlike basic 2PL, which permits release in its shrinking phase. FDB-style early release would allow interference before completion and would not meet rigorous locking. The locks are eventually released at the transaction boundary, so never is also false.

Why each choice is right or wrong

A - Incorrect: Releasing locks at the beginning would not protect the transaction's later accesses.

B - Incorrect: Rigorous 2PL does not permit early release during ongoing transaction execution.

C - Correct: Both shared and exclusive locks remain held until commit or abort ends the transaction.

D - Incorrect: Locks are released at completion; the protocol does not hold them forever.

Primary note proof

Study-note section ADB12

Strict 2PL retains exclusive locks until commit/abort. Rigorous 2PL retains both shared and exclusive locks until that point. Conservative/static 2PL obtains all needed locks before execution and does not begin while only a subset is available, preventing hold-and-wait deadlock. It requires knowing the lock set and can reduce concurrency. Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 86 | lines 16-24

◦ Locking (Growing) Phase:
• A transaction applies locks (read or write) on desired data items one at a time.
• acquires all locks but cannot release any locks.
◦ Unlocking (Shrinking) Phase:
• A transaction unlocks its locked data items one at a time.
• Releases locks but cannot acquire any new locks.
● Requirement:
◦ For a transaction these two phases must be mutually exclusively, that is, during

Fundamental of Database Systems module | PDF page 94 | lines 13-20

4.2.5 Validation (Optimistic) Concurrency Control Schemes
◦ This technique allow transaction to proceed asynchronously and only at the time of commit, serializability is checked &
◦ transactions are aborted in case of non-serializable schedules.
◦ Good if there is little interference among transaction
◦ It has three phases: Read, Validation , and Write phase
i.. Read phase:
◦ A transaction can read values of committed data items. However, updates are

Question sources

Computer Science model exam · 2015 · Question 59

Question 28 | Q00594

In which of the following state of the transaction executes all its operations successfully and all the effects are now permanently saved on the database system?

A. Active

B. Partial committed

C. Committed

D. Failed

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: Committed

The committed state guarantees persistent changes. Partially committed only signals the end of read/write operations and does not yet ensure permanence.

Why each choice is right or wrong

A - Incorrect: Active means execution is in progress. (PDF pp. 65.)

B - Incorrect: Partially committed explicitly does not guarantee permanent modification. (PDF pp. 65.)

C - Correct: Committed ensures persistent changes. (PDF pp. 65.)

D - Incorrect: Failed is an aborted or unsuccessful transaction. (PDF pp. 65.)

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 65 | lines 4-7

> ◦ Committed state -ensures that all the changes done on a record by a transition were done persistently
> ◦ Failed state happens when a transaction is aborted during its active state or if one of the rechecking is fails

Question sources

Computer Science model exam · 2015 · Question 63

Question 29 | Q00598

Let transaction T1 has obtained a shared mode lock S on data item Y and transaction T2 has obtained an exclusive lock X on data item Z , according to the above statement which one is correct?

A. T1 can read Y but cannot write Y

B. T2 can read Z but cannot write Z

C. T1 can read Z but cannot write Z

D. T2 can read and write item Y

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer A: T1 can read Y but cannot write Y

Shared locks permit reads but not updates. Exclusive locks permit both reading and updating while excluding other transactions' locks on that item.

Why each choice is right or wrong

A - Correct: T1's shared lock authorizes reading Y, not writing it. (PDF pp. 82, 83.)

B - Incorrect: T2's exclusive lock permits both read and update of Z. (PDF pp. 82, 83.)

C - Incorrect: T1 cannot read Z while another transaction holds its exclusive lock. (PDF pp. 82, 83.)

D - Incorrect: T2 cannot obtain an exclusive write lock on Y while T1 holds its shared lock. (PDF pp. 82, 83.)

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks.

Study-note section ADB11

> One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 82 | lines 20-26

• Shared mode: shared lock (X)
• More than one transaction can apply share lock on X for reading its value but no write lock can be applied on X by any other transaction.
• Exclusive mode: Write lock (X)
• Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X.
• Conflict matrix

Fundamental of Database Systems module | PDF page 83 | lines 2-8

• Lock Manager:
• Managing locks on data items.
• Lock table:
• Lock manager uses it to store the identify of transaction locking a data item, the data item, lock mode and pointer to the next data item locked. One simple way to implement a lock table is through linked list.

Question sources

Computer Science model exam · 2015 · Question 67

Question 30 | Q00615

Which claim about deadlock prevention and starvation is false?

A. In conservative 2PL, starvation may occur

B. Wound-wait guarantees starvation freedom regardless of restart priorities or scheduling fairness

C. Deadlock can be prevented by wait-die

D. Some deadlock-prevention methods may cause starvation

Hint

Identify requester and holder ages before applying the protocol's action.

Answer B: Wound-wait guarantees starvation freedom regardless of restart priorities or scheduling fairness

Deadlock freedom does not itself guarantee that each transaction eventually completes. Wound-wait uses transaction age to decide whether a requester waits or aborts a younger holder, but a starvation-freedom argument also needs appropriate restart priorities, fair scheduling and holders that eventually finish. Its unconditional guarantee in B is therefore false. Conservative 2PL can postpone a transaction indefinitely under unfair admission, and wait-die prevents waiting cycles. Some prevention policies can cause repeated waiting or victimization.

Why each choice is right or wrong

A - Incorrect: True: conservative 2PL prevents cycles but does not by itself guarantee fair admission.

B - Correct: Correct selection: the unconditional starvation-freedom claim omits necessary progress and priority assumptions.

C - Incorrect: True: wait-die is a timestamp-based deadlock-prevention policy.

D - Incorrect: True: preventing deadlock can still leave a transaction repeatedly postponed or aborted.

Primary note proof

Study-note section ADB14

> Retain the original timestamp when restarting an aborted transaction so it gains relative priority as new transactions arrive. Starvation avoidance additionally assumes holders eventually finish and scheduling is fair.

Study-note section ADB13

> Fixed resource ordering and conservative 2PL prevent cycles. A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp. A timeout aborts long waits but may mistake slow progress for deadlock. Starvation is indefinite postponement while other work proceeds, without a required cycle. Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 88 | lines 19-27

> ● Wait-die
> ◦ If Ti has higher priority, it is allowed to wait; otherwise it is aborted.
> ◦ An older transaction is allowed to wait on a younger transaction.
> ◦ A younger transaction requesting an item held by an older transaction is aborted
> ◦ If TS(Ti) < TS(Tj), then (Ti older than Tj)Ti is allowed to wait.
> ◦ Otherwise (Ti younger than Tj)Abort Ti (Ti dies) and restart it later with the same timestamp
> T1(ts =10): M is locked , K is requested
> T2 (ts =20): K is locked, Z is requested

Fundamental of Database Systems module | PDF page 89 | lines 4-13

> • Wound-wait
> - The opposite of wait-die
> - If Ti has higher priority, abort Tj; otherwise Ti waits.
> - A younger transaction is allowed to wait on an older one
> - An older transaction requesting an item held by a younger transaction preempts the younger transaction by aborting it.
> - If TS(Ti) < TS(Tj), then (Ti older than Tj),Abort Tj (Ti wounds Tj) and restart Tj later with the same timestamp
> - Otherwise (Ti younger than Tj)Ti is allowed to wait
> T2 (ts =20)

Fundamental of Database Systems module | PDF page 90 | lines 12-21

◦ Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further while other transaction continue normally
◦ This may occur , if the waiting method for item locking:
• Gave priority for some transaction over others
• Problem in Victim selection algorithm- it is possible that the same transaction may consistently be selected as victim and rolled-back
.example In Wound-Wait
◦ Solution
• FIFO
• Allow for transaction that wait for a longer time

Question sources

Computer Science model exam · 2015 · Question 84

Question 31 | Q00616

Concurrency control in DBMS is important for which of the following reasons?

A. To ensure data integrity when updates occur to the database in a single user environment

B. To ensure data integrity when reads occur to the database in a single user environment

C. To ensure data integrity when updates occur to the database in a multi user environment

D. To ensure data integrity when reads occur to the database in a multi user environment

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer C: To ensure data integrity when updates occur to the database in a multi user environment

Concurrency control preserves consistency when multiple users' updates interleave and conflict. Read-only sharing alone does not cause lost updates; the intended scenario is a multiuser system with updates.

Why each choice is right or wrong

A - Incorrect: Single-user updates lack concurrent intertransaction conflicts. (PDF pp. 68, 70, 79.)

B - Incorrect: Single-user reads do not create the cited conflicts. (PDF pp. 68, 70, 79.)

C - Correct: Multiuser updates can create read/write and write/write conflicts requiring scheduling. (PDF pp. 68, 70, 79.)

D - Incorrect: Reads can be involved when others update, but reads alone do not capture the source's motivating condition. (PDF pp. 68, 70, 79.)

Primary note proof

Study-note section ADB08

Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB06

Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure.

Study-note section ADB09

Read/read pairs do not conflict.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 68 | lines 4-7

i. The Lost Update Problem
◦ This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

ii. The Temporary Update (or Dirty Read) Problem
◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
iii. The Incorrect Summary Problem
◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Fundamental of Database Systems module | PDF page 79 | lines 4-15

- Transaction Processor is divided into:
◦ A concurrency-control manager, or scheduler, responsible for assuring isolation of transactions
◦ A logging and recovery manager, responsible for the durability of transactions.
- The scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed one-at-a-time.
1. Purpose of Concurrency Control

◦ To enforce Isolation (through mutual exclusion) among conflicting transactions.
◦ To preserve database consistency through consistency preserving execution of transactions.
◦ To resolve read-write and write-write conflicts.

Question sources

Computer Science model exam · 2015 · Question 85

Question 32 | Q00624

A lock wait-for graph may contain several independent deadlock cycles. Which termination methods guarantee eliminating all deadlock cycles? Assume each aborted transaction is undone and releases its locks. Select all that apply. 'Half' and 'single' mean arbitrary selections without a repeated cycle check.

A. Abort all deadlocked transaction

B. Abort half of the deadlocked transaction

C. Aborting single deadlocked transactio

D. Aborting one transaction at a time until the deadlock cycle is eliminated

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer A, D: A. Abort all deadlocked transaction; D. Aborting one transaction at a time until the deadlock cycle is eliminated

Aborting all deadlocked transactions removes every participating wait, although it wastes work. A more selective complete method aborts a suitable victim, releases its locks and repeats until no cycles remain. An arbitrary half of the participants can leave a cycle intact, and one arbitrary victim may resolve only one of several independent cycles. The guarantee therefore applies to A and D.

Why each choice is right or wrong

A - Correct: Removing all deadlocked participants eliminates all their cycles.

B - Incorrect: An arbitrary half may miss an entire remaining cycle.

C - Incorrect: One victim need not break other independent cycles.

D - Correct: Repeated victim selection and graph checking continues until every cycle is gone.

Primary note proof

Study-note section ADB13

Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if

other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp.

Study-note section ADB13

Starvation is indefinite postponement while other work proceeds, without a required cycle.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 89 | lines 4-13

• Wound-wait
- The opposite of wait-die
- If Ti has higher priority, abort Tj; otherwise Ti waits.
- A younger transaction is allowed to wait on an older one
- An older transaction requesting an item held by a younger transaction preempts the younger transaction by aborting it.
- If TS(Ti) < TS(Tj), then (Ti older than Tj),Abort Tj (Ti wounds Tj) and restart Tj later with the same timestamp
- Otherwise (Ti younger than Tj)Ti is allowed to wait
T2 (ts =20)

Fundamental of Database Systems module | PDF page 90 | lines 12-21

◦ Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further while other transaction continue normally
◦ This may occur , if the waiting method for item locking:
• Gave priority for some transaction over others
• Problem in Victim selection algorithm- it is possible that the same transaction may consistently be selected as victim and rolled-back
.example In Wound-Wait
◦ Solution
• FIFO
• Allow for transaction that wait for a longer time

Question sources

Computer Science model exam · 2015 · Question 93

Question 33 | Q00643

Among the components of the DBMS, which one is responsible for keeping the isolation of transactions?

A. Recovery protocol

B. Application programmer

C. Concurrency -control protocol

D. Database administrator

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: Concurrency -control protocol

Concurrency control regulates interactions between simultaneous transactions and enforces isolation. Recovery supports atomicity and durability by undoing incomplete effects and preserving committed work. Application programmers must supply correct transaction logic, and the DBA administers the database rather than serving as the executing concurrency-control mechanism. The question asks for that mechanism's primary responsibility, so C is the precise answer even though administration, applications and recovery all contribute to a working database.

Why each choice is right or wrong

A - Incorrect: Incorrect: recovery primarily supports atomicity and durability, rather than scheduling concurrent interactions for isolation.

B - Incorrect: Incorrect: an application programmer supplies transaction logic but is not the executing isolation-control mechanism.

C - Correct: Correct: concurrency control regulates conflicting concurrent operations to enforce isolation.

D - Incorrect: Incorrect: the DBA is an administrative human role, not the DBMS mechanism scheduling transaction interactions.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and

durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Fundamental of Database Systems module | PDF page 79 | lines 4-15

- Transaction Processor is divided into:
  - A concurrency-control manager, or scheduler, responsible for assuring isolation of transactions
  - A logging and recovery manager, responsible for the durability of transactions.
- The scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed one-at-a-time.

1. Purpose of Concurrency Control
   - To enforce Isolation (through mutual exclusion) among conflicting transactions.
   - To preserve database consistency through consistency preserving execution of transactions.
   - To resolve read-write and write-write conflicts.

Question sources

Computer Science model exam · 2015 · Question 112

Question 34 | Q00645

_____Occurs when two transactions update the same data item, but both read the same original value before update

A. Dirty read

B. Lost update

C. Incorrect summary

D. Unrepeatable read

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Lost update

Both transactions reading the original value lets a later write overwrite the earlier transaction's update; that update is lost.

Why each choice is right or wrong

A - Incorrect: Dirty read concerns uncommitted values from another transaction. (PDF pp. 68, 70.)

B - Correct: Lost update matches the stale-original-value overwrite. (PDF pp. 68, 70.)

C - Incorrect: Incorrect summary combines differently timed values in an aggregate. (PDF pp. 68, 70.)

D - Incorrect: Unrepeatable read concerns a later reread seeing a changed value. (PDF pp. 68, 70.)

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 68 | lines 4-7

> i. The Lost Update Problem
> ◦ This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

> ii. The Temporary Update (or Dirty Read) Problem
> ◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
> ◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
> iii. The Incorrect Summary Problem
> ◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
> Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Question sources

Computer Science model exam · 2015 · Question 114

Question 35 | Q00659

Which of the DBMS component ensures that the concurrent execution of multiple operations on the database results into consistent database state?

A. Log

B. Buffer manager

C. File manager

D. Transaction processing system

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Transaction processing system

The transaction processing system contains the scheduler and recovery managers. Its scheduling ensures concurrent individual actions have a net effect equivalent to one-at-a-time execution.

Why each choice is right or wrong

A - Incorrect: The log records changes but does not itself schedule consistent execution. (PDF pp. 79, 99.)

B - Incorrect: Buffers hold blocks and arrange writes, not the entire consistency protocol. (PDF pp. 79, 99.)

C - Incorrect: File management is not the scheduler responsible for concurrent effects. (PDF pp. 79, 99.)

D - Correct: Transaction processing contains the concurrency-control component implementing this guarantee. (PDF pp. 79, 99.)

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB09

> A serial schedule executes each transaction's operations consecutively.

Study-note section ADB09

> A serializable schedule is equivalent to some serial order; it may interleave transactions.

Related reading (context, not primary note proof)

Study-note section ADB07

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 79 | lines 4-15

- Transaction Processor is divided into:
  - A concurrency-control manager, or scheduler, responsible for assuring isolation of transactions
  - A logging and recovery manager, responsible for the durability of transactions.
- The scheduler (concurrency-control manager) must assure that the individual actions of multiple transactions are executed in such an order that the net effect is the same as if the transactions had in fact executed one-at-a-time.

1. Purpose of Concurrency Control
   - To enforce Isolation (through mutual exclusion) among conflicting transactions.
   - To preserve database consistency through consistency preserving execution of transactions.
   - To resolve read-write and write-write conflicts.

Fundamental of Database Systems module | PDF page 99 | lines 2-8

✓ Logging: In order to assure durability, every change in the database is logged separately on disk.
✓ Log manager initially writes the log in buffers and negotiates with the buffer manager to make sure that buffers are written to disk at appropriate times.
✓ Recovery Manager: will be able to examine the log of changes and restore the database to some consistent state.
Recovery techniques and facilities

Question sources

Computer Science model exam · 2015 · Question 128

Question 36 | Q00687

How do you prevent system privileges deletion and updating authority on PRICE and YEAR column of the CATALOGUE table from user Mohammed

A. GRANT DELETE, UPDATE (price,year) ON catalogue From Mohammed

B. ALLOW DELETE, UPDATE (price,year) ON catalogue From Mohammed

C. REVOKE DELETE, UPDATE (price,year) ON catalogue From Mohammed

D. DENY DELETE, UPDATE (price,year) ON catalogue From Mohammed

Hint

Track whether a privilege is being added or withdrawn and whether it applies to rows or columns.

Answer C: REVOKE DELETE, UPDATE (price,year) ON catalogue From Mohammed

To withdraw Mohammed's DELETE and column-specific UPDATE privileges, use REVOKE DELETE, UPDATE(price,year) ON catalogue FROM Mohammed. This concerns object privileges, despite the stem's 'system privileges' wording. Other independent grants may remain.

Why each choice is right or wrong

A - Incorrect: GRANT adds privileges and also uses TO, not FROM. (PDF pp. 54, 55.)

B - Incorrect: ALLOW is not the module's SQL privilege command. (PDF pp. 54, 55.)

C - Correct: REVOKE with ON and FROM matches the module syntax. (PDF pp. 54, 55.)

D - Incorrect: DENY is not the revocation command presented in this module. (PDF pp. 54, 55.)

Primary note proof

Study-note section ADB25

> GRANT SELECT ON Catalogue TO Mohammed; permits reads. REVOKE DELETE, UPDATE (Price,Year) ON Catalogue FROM Mohammed; removes the specified table deletion and column update grants in SQL dialects supporting that form. DELETE applies to rows/table access rather than particular columns; UPDATE can be column-scoped. REVOKE removes permission, whereas GRANT gives it.

Study-note section ADB25

> WITH GRANT OPTION permits a recipient to grant a privilege onward; removing an upstream grant can cascade or be restricted according to the chosen command and DBMS.

Study-note section ADB25

> Removing one direct grant may leave effective access through another role or grant path.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 54 | lines 12-20

> GRANT <privileges>
> ON <object>
> TO <users>
> [WITH GRANT OPTION]
> • <privileges> is a list of
> SELECT <columns>, INSERT <columns>, DELETE, and UPDATE <columns>, or
> simply ALL
> • <users> is a list of user names or PUBLIC
> • <object> is the name of a table or view

Fundamental of Database Systems module | PDF page 55 | lines 8-12

> REVOKE <privileges>
> ON <object>
> FROM <users>

- If a user has the same privilege from other users then they keep it
- All privileges dependent on the revoked one are also revoked

Question sources

Computer Science model exam 2 · Question 28

Question 37 | Q00890

Inclient-server database architecture, what is the primary role of the server?

A. Provide the GUI for end users

B. Initiate all network connections and poll clients for updates

C. Store and manage shared data, enforce ACID properties, and process client requests

D. Cache all user sessions locally for offline access

Hint

Identify which tier submits, mediates or executes a request.

Answer C: Store and manage shared data, enforce ACID properties, and process client requests

The database server provides database functionality and manages local data, while clients submit requests and combine results. DBMS components enforce transaction properties.

Why each choice is right or wrong

A - Incorrect: The client-facing GUI is not the server's primary data-management role. (PDF pp. 125, 126, 66.)

B - Incorrect: Clients initiate requests; polling clients is not the described workflow. (PDF pp. 125, 126, 66.)

C - Correct: Managing data and processing client requests matches server responsibilities. (PDF pp. 125, 126, 66.)

D - Incorrect: Offline local session caching is not the described database server function. (PDF pp. 125, 126, 66.)

Primary note proof

Study-note section ADB34

In client/server architecture, clients submit requests while servers manage shared data, execute operations and enforce supported integrity/transaction rules. In collaborating-server designs, servers cooperate on subqueries and transactions. Middleware can translate requests, locate data, coordinate operations and combine results across independent servers. These describe communication responsibilities, not fragmentation types.

Study-note section ADB34

State how much schema, transaction and administrative autonomy local sites retain.

Study-note section ADB34

A tightly integrated distributed system can present one global schema and shared control.

Related reading (context, not primary note proof)

Study-note section ADB06

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 125 | lines 11-16

6.6 Client-Server Database Architecture
• It consists of clients running client software, a set of servers which provide all database functionalities and a reliable communication infrastructure.
three-tier client/server architecture.
• Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server. This intermediate layer

Fundamental of Database Systems module | PDF page 126 | lines 8-16

• The server software is responsible for local data management at a site, much like centralized DBMS software.
• The client software is responsible for most of the distribution function.
• The communication software manages communication among clients and servers.
• The processing of a SQL queries goes as follows:
• Client parses a user query and decomposes it into a number of independent sub-queries. Each subquery is sent to appropriate site for execution.
• Each server processes its query and sends the result to the client.
• The client combines the results of subqueries and produces the final result.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Computer Science Exit Exam · 2018 · Question 3

Question 38 | Q00892

In two-phase locking (2PL), during which phase may a transaction acquire locks but not release any?

A. Growing phase

B. Commit phase

C. .

Validation phase

D. Shrinking phase

Hint

Track the first lock release and the transaction-completion boundary.

Answer A: Growing phase

The growing phase acquires locks without releasing any. Shrinking starts with release and prohibits new acquisitions.

Why each choice is right or wrong

A - Correct: Growing has the acquisition-without-release rule. (PDF pp. 86.)

B - Incorrect: Commit is not the phase name in basic 2PL. (PDF pp. 86.)

C - Incorrect: Validation belongs to optimistic concurrency control. (PDF pp. 86.)

D - Incorrect: Shrinking permits release, not acquisition. (PDF pp. 86.)

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 86 | lines 16-24

◦ Locking (Growing) Phase:
• A transaction applies locks (read or write) on desired data items one at a time.
• acquires all locks but cannot release any locks.
◦ Unlocking (Shrinking) Phase:
• A transaction unlocks its locked data items one at a time.
• Releases locks but cannot acquire any new locks.
● Requirement:
◦ For a transaction these two phases must be mutually exclusively, that is, during

Question sources

Computer Science Exit Exam · 2018 · Question 5

Question 39 | Q00915

Assume durable logs and serializable concurrency control, but no checkpoints or other valid log-truncation boundary. In a high-transaction system using deferred update, which recovery cost can grow after long uptime?

A. Recovery may require scanning a very long accumulated log

B. Increased disk I/O due to frequent page copying

C. Shared locks cannot be used

D. View serializability cannot be ensured

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer A: Recovery may require scanning a very long accumulated log

Checkpoints record recovery state so recovery need not inspect the entire accumulated history. Without a checkpoint or another valid truncation boundary, the log that must be searched can grow throughout a long-running workload. Deferred update does not remove that problem. It specifies when persistent changes occur, not a requirement to copy shadow pages or a prohibition on shared locks. Serializability is enforced by the assumed concurrency-control mechanism, rather than by the existence of checkpoints.

Why each choice is right or wrong

A - Correct: Correct: without a recovery boundary, a long-running system may need to inspect a very large accumulated log.

B - Incorrect: Incorrect: deferred update describes timing and does not require frequent shadow-page copying.

C - Incorrect: Incorrect: deferring persistent writes does not prohibit shared-lock concurrency control.

D - Incorrect: Incorrect: serializability is enforced by the assumed concurrency control; a checkpoint limits recovery work rather than determining schedule equivalence.

Primary note proof

Study-note section ADB30

A fuzzy checkpoint permits ongoing transactions and records active-transaction and dirty-page information without forcing all pages simultaneously. Recovery may start analysis at the checkpoint but redo from an earlier required log position. A checkpoint is neither a full backup nor an automatic commit of active transactions. Without checkpoints or another valid truncation boundary, long-running systems may need expensive log scans; deferred update does not avoid that problem.

Study-note section ADB29

Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO.

Study-note section ADB12

Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 108 | lines 24-29

• During recovery, all transactions of the commit table are redone and all transactions of active tables are ignored since none of their AFIMs reached the database.
• It is possible that a commit table transaction may be redone twice but this does not create any inconsistency because of a redone is —idempotent‖,
• that is, one redone for an AFIM is equivalent to multiple redone for the same AFIM.

Question sources

Computer Science Exit Exam · 2018 · Question 28

Question 40 | Q00933

Which listed techniques use transaction logs for recovery in the course model? Interpret deferred modification as the conventional logged, no-force deferred-update method. Select all that apply.

A. Recovery in Multi-database Systems

B. ARIES Recovery Algorithm

C. Shadow Paging Technique

D. Deferred Database Modification

Hint

Separate update timing from update location and follow the commit boundary.

Answer B, D: B. ARIES Recovery Algorithm; D. Deferred Database Modification

ARIES uses write-ahead logging and log-driven recovery metadata. Conventional logged deferred modification also uses recorded updates to redo committed effects that have not reached persistent pages. Both are log-based techniques under the stated model. Shadow paging instead preserves committed page mappings and installs a new map at commit. Recovery in multidatabase systems names an environment rather than one particular algorithm.

Why each choice is right or wrong

A - Incorrect: Multidatabase recovery names an environment rather than identifying a particular log-based technique.

B - Correct: ARIES explicitly relies on WAL and logged recovery metadata.

C - Incorrect: Shadow paging preserves page maps as an alternative to conventional log-based recovery.

D - Correct: Deferred modification can use a log to redo committed changes without undo.

Primary note proof

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB31

> ARIES uses WAL, page log sequence numbers (pageLSN), a transaction table and a dirty-page table.

Study-note section ADB29

Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 104 | lines 19-25

• For Undo: Before a data item‘s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
• For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
• Standard Recovery Terminology

Fundamental of Database Systems module | PDF page 107 | lines 16-28

• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:
• A set of transactions records their updates in the log.
• At commit point under WAL scheme these updates are saved on database disk.
• No undo is required because no AFIM is flushed to the disk before a transaction commits.

Fundamental of Database Systems module | PDF page 109 | lines 13-22

iii. Shadow Paging
• Maintain two page tables during life of a transaction: current page and shadow page table.
• When transaction starts, two pages are the same.
• Shadow page table is never changed thereafter and is used to restore database in event of failure.
• During transaction, current page table records all updates to database.
• When transaction completes, current page table becomes shadow page table.
X and Y: Shadow copies of data items
X' and Y': Current copies of data items

Question sources

Computer Science Exit Exam · 2018 · Question 46

Question 41 | Q00969

Which type of fragmentation in a distributed database divides a relation into subrelations defined by subsets of its columns, normally repeating a key to support reconstruction?

A. Replication

B. Location

C. Horizontal

D. Vertical

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer D: Vertical

Vertical fragmentation projects column subsets into separate fragments. Repeating an appropriate key or row identifier allows a join to reconstruct the original rows without spurious combinations. Horizontal fragmentation instead selects subsets of rows, reconstructed by union. Replication creates copies rather than dividing the relation by attributes. Location transparency hides where data resides; it does not name the column-splitting design operation. The corrected category therefore supports the existing vertical-fragmentation answer.

Why each choice is right or wrong

A - Incorrect: Replication creates copies of data rather than splitting a relation into column subsets.

B - Incorrect: Location transparency hides placement details; it is not a fragmentation method.

C - Incorrect: Horizontal fragmentation partitions rows, not columns.

D - Correct: Correct: vertical fragmentation splits columns and normally repeats a reconstruction key.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB35

> Location transparency hides which site holds a named object.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 116 | lines 11-28

ii. Replication transparency:
• It allows to store copies of a data at multiple sites as shown in the above diagram.
• This is done to minimize access time to the required data.
iii. Fragmentation transparency:
• Allows to fragment a relation horizontally (create a subset of rows of a relation) or vertically (create a subset of columns of a relation).
iv. Increased reliability and availability:
• Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
• A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
v. Improved performance:
• A distributed DBMS fragments the database to keep data closer to where it is needed most.
• This reduces data management (access and modification) time significantly.

Fundamental of Database Systems module | PDF page 118 | lines 3-21

• Horizontal fragmentation
• It is a horizontal subset of a relation which contain those of rows which satisfy selection conditions.
• Consider the Employee relation with selection condition (DNO = 5). All rows satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.
• A selection condition may be composed of several conditions connected by AND or OR.
• Vertical fragmentation
• It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns.
- Consider the Employee relation. A vertical fragment of can be created by keeping the values of Name, Bdate, Sex, and Address.
- Because there is no condition for creating a vertical fragment, each fragment must include the primary key attribute of the parent relation Employee. In this way all vertical fragments of a relation are connected.
• Representation
• There three rules that must be followed during fragmentation
• Completeness - if a relation r is decomposed into fragments r1, r2... rn ,

Question sources

Computer Science Exit Exam · 2018 · Question 82

Question 42 | Q00993

In two-phase locking, which phase permits releasing existing locks but forbids acquiring new locks?

A. Expanding phase

B. Waiting phase

C. Shrinking phase

D. All of the above

Hint

Track the first lock release and the transaction-completion boundary.

Answer C: Shrinking phase

Two-phase locking divides lock use into a growing phase and a shrinking phase. During growing, a transaction may acquire or upgrade locks; once it releases a lock, it enters shrinking and cannot acquire any new lock. Waiting describes delay on a conflicting request, not a separate 2PL phase defined by releasing locks. Because the growing phase allows acquisition instead of release, the all-of-the-above alternative also fails.

Why each choice is right or wrong

A - Incorrect: Incorrect: the growing/expanding phase acquires locks and does not release them.

B - Incorrect: Incorrect: waiting on a conflicting request is a delay condition, not the formal 2PL release phase.

C - Correct: Correct: shrinking permits release while forbidding new acquisitions.

D - Incorrect: Incorrect: the growing phase does not meet the release/no-new-lock rule.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 86 | lines 16-24

> ◦ Locking (Growing) Phase:
> • A transaction applies locks (read or write) on desired data items one at a time.
> • acquires all locks but cannot release any locks.

◦ Unlocking (Shrinking) Phase:
• A transaction unlocks its locked data items one at a time.
• Releases locks but cannot acquire any new locks.
● Requirement:
◦ For a transaction these two phases must be mutually exclusively, that is, during

Question sources

National Exit Exam practice questions · Question 7

Computer Science Exit Exam practice collection · Question 65

Question 43 | Q01002

Which one of the SQL keywords is used to provide access?

A. REVOKE

B. DISTINCT

C. GRANT

D. None of the above

Hint

Track whether a privilege is being added or withdrawn and whether it applies to rows or columns.

Answer C: GRANT

GRANT gives specified users privileges on an object. REVOKE removes privileges.

Why each choice is right or wrong

A - Incorrect: REVOKE withdraws access privileges. (PDF pp. 54, 55.)

B - Incorrect: DISTINCT is not the module's access-authorization command. (PDF pp. 54, 55.)

C - Correct: GRANT supplies the privileges on tables or views. (PDF pp. 54, 55.)

D - Incorrect: GRANT is listed, so None is false. (PDF pp. 54, 55.)

Primary note proof

Study-note section ADB25

GRANT SELECT ON Catalogue TO Mohammed; permits reads. REVOKE DELETE, UPDATE (Price,Year) ON Catalogue FROM Mohammed; removes the specified table deletion and column update grants in SQL dialects supporting that form. DELETE applies to rows/table access rather than particular columns; UPDATE can be column-scoped. REVOKE removes permission, whereas GRANT gives it.

Study-note section ADB25

WITH GRANT OPTION permits a recipient to grant a privilege onward; removing an upstream grant can cascade or be restricted according to the chosen command and DBMS.

Study-note section ADB25

Roles group privileges.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 54 | lines 12-20

GRANT <privileges>
ON <object>
TO <users>
[WITH GRANT OPTION]
• <privileges> is a list of
SELECT <columns>, INSERT <columns>, DELETE, and UPDATE <columns>, or
simply ALL
• <users> is a list of user names or PUBLIC
• <object> is the name of a table or view

Fundamental of Database Systems module | PDF page 55 | lines 8-12

REVOKE <privileges>
ON <object>
FROM <users>
- If a user has the same privilege from other users then they keep it
- All privileges dependent on the revoked one are also revoked

Question sources

National Exit Exam practice questions · Question 16

Computer Science Exit Exam practice collection · Question 48

Question 44 | Q01156

is a tree data structure that corresponds to a relational algebra expression.

A. a | Query graph x

B. i? Query tree WA

C. '@| Query optimizer x

D. | None of the above x

Hint

Separate query representation, strategy selection and execution.

Answer B: i? Query tree WA

A query tree corresponds to a relational algebra expression, with relations as leaves and operations as internal nodes.

Why each choice is right or wrong

A - Incorrect: Query graph corresponds to relational calculus. (PDF pp. 34.)

B - Correct: Query tree is the exact defined algebraic tree. (PDF pp. 34.)

C - Incorrect: The optimizer chooses an execution strategy; it is not this data structure. (PDF pp. 34.)

D - Incorrect: The matching tree is explicitly listed. (PDF pp. 34.)

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB-S01

> A query tree corresponds to a relational-algebra expression: base relations appear at leaves and operators at internal nodes.

Study-note section ADB17

> A query tree has base relations at leaves and operators at internal nodes; a query graph emphasizes relations and predicates.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 34 | lines 5-7

> 2. Apply heuristics rules to optimize the internal representation.
> 3. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

Question sources

Computer Science Exit Exam practice questions · Question 17

Question 45 | Q01184

Which listed approaches use both undo and redo? Interpret Immediate Update as the conventional steal/no-force logged model and Deferred Update as no-steal/no-force; use the course's ARIES definition. Select all that apply.

A. Immediate Update

B. Deferred Update

C. ARIES Algorithm

D. None

Hint

Determine whether uncommitted changes or unflushed committed changes may exist on persistent pages.

Answer A, C: A. Immediate Update; C. ARIES Algorithm

Steal allows uncommitted changes to reach persistent pages, creating a possible need for undo. No-force allows committed changes to remain buffered, creating a possible need for redo. Immediate update under the specified combined policy therefore needs both. ARIES also performs redo and then undoes losers. The specified deferred no-steal/no-force model needs redo but no undo, so it does not qualify.

Why each choice is right or wrong

A - Correct: The stated steal/no-force policy requires undo and redo.

B - Incorrect: The specified deferred policy prevents persistent uncommitted effects and needs redo only.

C - Correct: ARIES repeats history with redo and reverses losers with undo.

D - Incorrect: Two listed approaches use both actions.

Primary note proof

Study-note section ADB28

> Steal permits a page with uncommitted changes to be written to disk; hence recovery may need undo. No-steal prevents those uncommitted effects reaching the persistent database. Force writes a transaction's updated data pages at commit; no-force permits committed dirty pages to remain buffered, hence recovery may need redo.

Study-note section ADB28

> For the conventional page/log model: steal/no-force needs undo and redo; steal/force needs undo but no redo; no-steal/no-force needs redo but no undo; no-steal/force needs neither data-page undo nor redo after a system crash.

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO.

Related reading (context, not primary note proof)

Study-note section ADB31

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 109 | lines 13-22

iii. Shadow Paging
• Maintain two page tables during life of a transaction: current page and shadow page table.
• When transaction starts, two pages are the same.
• Shadow page table is never changed thereafter and is used to restore database in event of failure.
• During transaction, current page table records all updates to database.
• When transaction completes, current page table becomes shadow page table.
X and Y: Shadow copies of data items
X' and Y': Current copies of data items

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing,

before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Question sources

Computer Science model exam 2 · Question 5

Question 46 | Q01209

Which recovery mechanism preserves the committed page map while writing modified pages elsewhere, then atomically installs a new map at commit as an alternative to conventional log-based recovery?

A. Disk recovery

B. Shadow paging

C. Deferred update

D. Crash recovery

Hint

Separate update timing from update location and follow the commit boundary.

Answer B: Shadow paging

Shadow paging preserves the committed page map and writes modified pages to separate locations. Once the required pages are durable, commit atomically installs the new map; before that point, recovery can use the old committed map. Deferred update instead describes postponing persistent database changes until commit and does not itself specify this page-map mechanism. Disk recovery and crash recovery name recovery domains or events, not the particular algorithm requested.

Why each choice is right or wrong

A - Incorrect: Incorrect: disk recovery describes a recovery domain, not the specified preserved-page-map mechanism.

B - Correct: Correct: shadow paging preserves the committed map and installs the new map atomically at commit.

C - Incorrect: Incorrect: deferred update specifies commit timing, not this copy-on-write page-map algorithm.

D - Incorrect: Incorrect: crash recovery describes recovery after an event rather than the particular page-map mechanism.

Primary note proof

Study-note section ADB29

In-place update overwrites the original location. Shadow/copy-on-write update writes a new location while preserving the committed version. Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable. Before commit, a crash uses the old map. It is an alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Study-note section ADB29

Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO.

Related reading (context, not primary note proof)

Study-note section ADB26

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 102 | lines 25-28

- We study two approaches:
- log-based recovery, and
- shadow-paging
- We assume (initially) that transactions run serially, that is, one after the other.

Fundamental of Database Systems module | PDF page 109 | lines 13-22

iii. Shadow Paging
• Maintain two page tables during life of a transaction: current page and shadow page table.
• When transaction starts, two pages are the same.
• Shadow page table is never changed thereafter and is used to restore database in event of failure.
• During transaction, current page table records all updates to database.
• When transaction completes, current page table becomes shadow page table.
X and Y: Shadow copies of data items
X' and Y': Current copies of data items

Question sources

Computer Science model exam 2 · Question 30

Question 47 | Q01225

Which specific DBMS component directly handles undo of incomplete persistent changes after failure to support transaction atomicity? Choose the specialized component rather than its broader parent subsystem.

A. Transaction management component of DBMS

B. Application Programmer

C. Concurrency control component of DBMS

D. Recovery management component of DBMS

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: D. Recovery management component of DBMS

Atomicity requires all transaction effects or none. After failure, recovery can undo incomplete persistent changes to enforce that outcome. Transaction management is the broader subsystem containing these responsibilities, but the question asks for the specialized component. Concurrency control primarily addresses isolation, and application code alone cannot provide the DBMS's crash-recovery undo of partially persisted work.

Why each choice is right or wrong

A - Incorrect: This is the parent subsystem, not the specialized failure-undo component requested.

B - Incorrect: Application logic alone does not provide durable crash undo.

C - Incorrect: Concurrency control primarily enforces isolation.

D - Correct: Recovery directly supports atomicity by undoing incomplete effects.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Fundamental of Database Systems module | PDF page 97 | lines 4-13

i. Purpose of Database Recovery
◦ To bring the database into the last consistent state, which existed prior to the failure.
◦ To preserve transaction properties (Atomicity &and Durability).
• The recovery manager of a DBMS is responsible to ensure
• atomicity by undoing the action of transaction that do not commit
• Durability by making sure that all actions of committed transaction survive system crash
ii. Types of Failure
◦ The database may become unavailable for use due to :

Fundamental of Database Systems module | PDF page 66 | lines 22-31

- Atomicity- either all or none of the above operation will be done - this is materialized by transaction management component of DBMS
- Consistency-the sum of CA2090 and SB2359 be unchanged by the execution of Ti i.e 8500- this is the responsibility of application programmer who codes the transaction
- Isolation- when several transaction are being processed concurrently on a data item they may create many inconsistent problems. So handling such case is the responsibility of Concurrency control component of the DBMS
- Durability - once Ti writes its update this will remain there when the database restarted from failure . This is the responsibility of recovery management components of the DBMS

Question sources

Computer Science model exam 2 · Question 46

Question 48 | Q01238

If a database server is referenced in a distributed transaction, the value of its commit point strength determines which role it plays in the _____.

A. Two-phase commit

B. Transaction locking

C. Two-phase locking

D. Checkpoints

Hint

Distinguish coordinated transaction outcome from concurrent access and local storage.

Answer A: Two-phase commit.

ADB39 explicitly links Oracle COMMIT_POINT_STRENGTH to choosing the commit point site in distributed transactions. This is a role in two-phase commit, which coordinates an atomic outcome among participants. It is not a setting for acquiring or releasing locks, nor for checkpoints that bound recovery work. Higher eligible strength is preferred for the commit point.

Why each choice is right or wrong

A - Correct: Commit-point strength selects an Oracle distributed-commit role within the two-phase commit mechanism.

B - Incorrect: Transaction locking controls concurrent access rather than choosing the distributed commit point site.

C - Incorrect: Two-phase locking governs lock acquisition and release, not the distributed commit decision.

D - Incorrect: Checkpoints record recovery state and do not choose an Oracle commit point site.

Primary note proof

Study-note section ADB39

> Two-phase commit (2PC) first asks participants to prepare and durably promise they can commit. If all vote yes, the coordinator records commit durably and tells participants; otherwise it decides abort. A prepared participant must retain enough state to obey the final decision. Coordinator/communication failure can leave it in doubt and block progress until the decision is learned. Timeout after preparation is not permission to invent a commit or abort.

Study-note section ADB39

> Oracle's COMMIT_POINT_STRENGTH helps choose the commit point site in its distributed transaction mechanism, with higher eligible strength preferred.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Related reading (context, not primary note proof)

Study-note section ADB30

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 123 | lines 15-29

• Distributed commit:
• A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
• Distributed deadlock:
• Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.
6.5.1 Distributed Concurrency control
i. Primary site technique: A single site is designated as a primary site which serves as a coordinator for transaction management.
• Transaction management:
- Concurrency control and commit are managed by this site.
- In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed.
- Advantages:

Question sources

Computer Science model exam 2 · Question 59

Question 49 | Q01240

What is the distinctive purpose of query optimization within the broader query-processing pipeline? Select the most specific description.

A. Both are used to process SQL statements to return answers.

B. Both filter retrieved rows by a date/time condition.

C. Both improve the performance of the database by efficiently executing a query.

D. Both are important in executing queries to access information from the DB.

Hint

Separate query representation, strategy selection and execution.

Answer C: C. Both improve the performance of the database by efficiently executing a query.

Query optimization selects a suitable efficient execution strategy among equivalent ways of answering a query. C explicitly states that efficiency objective. Returning answers and accessing information are broader descriptions of processing and do not distinguish optimization's particular purpose. Filtering data by a date or time is a query condition rather than the general goal of choosing a plan.

Why each choice is right or wrong

A - Incorrect: Returning answers describes the broad pipeline but omits the distinctive efficiency objective.

B - Incorrect: Temporal filtering is query logic, not the general purpose of optimization.

C - Correct: Efficient execution is the specific objective of selecting a suitable plan.

D - Incorrect: Accessing information describes the broad pipeline rather than optimization's efficiency role.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB17

> DDL is not a mandatory stage before each SELECT, so authentication→DDL→DML permutations do not describe the processing sequence.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 27 | lines 4-15

> What is Query Processing?
> Steps required to transform high level SQL query into a correct and —efficient‖ strategy for execution and retrieval.
> What is Query Optimization?
> ✓ The activity of choosing a single —efficient‖ execution strategy (from hundreds) as determined by database catalog statistics.
> ✓ The aim is to answer which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?
> ✓ For each algebraic operator, what algorithm (of several available) do we use to

compute that operator?
✓ How do operations pass data (main memory buffer, disk buffer,...)?
✓ Will this plan minimize resource usage? (CPU/Response Time/Disk)

Fundamental of Database Systems module | PDF page 30 | lines 17-30

1.8 Query Optimization
• Everyone wants the performance of their database to be optimal. In particular, there is often a requirement for a specific query or object that is query based, to run faster.
• Problem of query optimization is to find the sequence of steps that produces the answer to user request in the most efficient manner, given the database structure.
• The performance of a query is affected by the tables or views that underlies the query and by the complexity of the query.
• Given a request for data manipulation or retrieval, an optimizer will choose an optimal plan for evaluating the request from among the manifold alternative strategies. i.e. there are many ways (access paths) for accessing desired file/record.
• Hence ,DBMS is responsible to pick the best execution strategy based on various considerations(Least amount of I/O and CPU resources.)
• Example: Consider relations r(AB) and s(CD). We require r X s.
• Method 1 :

Question sources

Computer Science model exam 2 · Question 61

Question 50 | Q01245

What is the objective of a Serializable schedule?

A. Find a Non-serial schedule that allows the interleaving of a transaction and produces the same result as a serial schedule.

B. Find a Serial schedule that allows the interleaving of a transaction and produces the same result as a serial schedule.

C. Find a Serial schedule that doesn't allow the interleaving of a transaction and produces the same result as a serial schedule.

D. Find a Non-serial schedule that doesn't allow the interleaving of a transaction and produces the same result as a serial schedule.

Hint

Distinguish transaction ordering, conflict conditions and full execution equivalence.

Answer A: Find a Non-serial schedule that allows the interleaving of a transaction and produces the same result as a serial schedule.

Serial schedules run each transaction consecutively. Serializability permits nonserial interleaving while preserving the effect of a complete serial schedule (p.72). A expresses this concurrency objective.

Why each choice is right or wrong

A - Correct: Interleaving plus serial-equivalent effect meets the objective.

B - Incorrect: A serial schedule cannot interleave transactions as claimed.

C - Incorrect: This describes serial execution, not the benefit of serializable interleaving.

D - Incorrect: Nonserial execution is distinguished by departure from consecutive transaction execution; this denies the intended interleaving.

Primary note proof

Study-note section ADB09

A serial schedule executes each transaction's operations consecutively. A serializable schedule is equivalent to some serial order; it may interleave transactions. Equality of one final value is insufficient evidence because different executions can accidentally coincide. Two operations conflict when they belong to different transactions, access the same item and at least one writes. Read/read pairs do not conflict.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 72 | lines 6-18

- Serial schedule:
  - A schedule S is serial if, for every transaction Ti participating in the schedule, all the operations of Ti are executed consecutively in the schedule. Otherwise, the schedule is called nonserial schedule.
  - For example, in the banking example suppose there are two transaction where one transaction calculate the interest on the account and another deposit some money into the account. hence the order of execution is important for the final result.
- Serializable schedule:
  - a schedule whose effect on any consistent database instance is identical to that of some complete serial schedule over the set of committed transactions in S.
  - A nonserial schedule S is serializable , is equivalent to say that it is correct to the result of one of the serial schedule .Example,

Question sources

Computer Science model exam 2 · Question 66

Question 51 | Q01249

Under the course data-model type-constructor convention, which statements are false? Select all that apply. Interpret type constructor as a data-model term, not an executable Java/C++ object constructor, and consider OID over one object's lifetime.

A. Object Identifier is immutable.

B. Type constructor is used to create a new object.

C. Tuple is a collection of records.

D. The atom constructor is a multi-value constructor.

Hint

Classify the shape of a value separately from an operation that creates an instance.

Answer B, C, D: B. Type constructor is used to create a new object.; C. Tuple is a collection of records.; D. The atom constructor is a multi-value constructor.

An OID remains stable over the object's lifetime, so A is true. A data-model type constructor specifies the shape of a value rather than executing creation of a new object. A tuple groups named components into one structured record, while an atom is one basic value. B, C and D therefore state incorrect properties under the specified convention.

Why each choice is right or wrong

A - Incorrect: OID stability over the object lifetime is true, so it does not answer the negative stem.

B - Correct: This is false under the note's convention: a type constructor specifies shape rather than executing object creation.

C - Correct: A tuple is one structured record of components, not a collection of records.

D - Correct: An atom represents one basic atomic value rather than a multivalued collection.

Primary note proof

Study-note section ADB04

> In a common object model, an object is described by (OID,type constructor,state). An atom is one basic value, such as an integer, real, Boolean or string. A tuple constructor groups named component values into one structured record, such as (Street:'A',City:'B'). It is not a collection of records.

Study-note section ADB04

> A type constructor defines the shape of a value; it is not the same as an executable constructor operation that creates a new object.

Study-note section ADB01

> The object identifier (OID) distinguishes objects independently of their attribute values and remains stable over that object's lifetime.

Question sources

Computer Science model exam 2 · Question 70

Question 52 | Q01264

Assume transaction A holds a shared lock R. If transaction B also requests for a shared lock on R

A. It will result in a deadlock situation

B. It will be granted as soon as it is released by A

C. It will immediately be granted

D. It will immediately be rejected

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer C: It will immediately be granted

The shared-mode rule permits more than one transaction to hold a read lock on X, while excluding write locks (p.82). With only A's shared lock specified, B's shared request is compatible.

Why each choice is right or wrong

A - Incorrect: Two compatible shared requests do not produce the stated deadlock.

B - Incorrect: Waiting for A is unnecessary for compatible reads.

C - Correct: The shared request is compatible and may be granted immediately under the stated conditions.

D - Incorrect: The shared rule supplies no rejection condition here.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB11

> One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Study-note section ADB11

> The lock manager decides requests; the lock table stores granted locks and waiting requests.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 82 | lines 20-26

• Shared mode: shared lock (X)
• More than one transaction can apply share lock on X for reading its value but no write lock can be applied on X by any other transaction.
• Exclusive mode: Write lock (X)
• Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X.
• Conflict matrix

Fundamental of Database Systems module | PDF page 83 | lines 23-25

• Reads cannot conflict, so more than one transaction can hold shared locks simultaneously on same item.
• Exclusive lock gives transaction exclusive access to that item.

Question sources

Computer Science model exam 2 · Question 85

Question 53 | Q01266

In the dirty read problem, one transaction reads:

A. an uncommitted value of another transaction

B. the committed value for another transaction

C. another transaction

D. one transaction commits another transaction

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer A: an uncommitted value of another transaction

The temporary-update/dirty-read definition involves another transaction accessing an updated item before the failed writer's value is restored. The hazard is observing the other transaction's uncommitted value.

Why each choice is right or wrong

A - Correct: This identifies the unsafe value from the other transaction.

B - Incorrect: A committed value lacks the defining uncommitted-update hazard.

C - Incorrect: Transactions read data items; this does not identify the dirty value.

D - Incorrect: One transaction committing another is not the read anomaly described.

Primary note proof

Study-note section ADB08

Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

ii. The Temporary Update (or Dirty Read) Problem
◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
iii. The Incorrect Summary Problem
◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Question sources

Computer Science model exam 2 · Question 87

Question 54 | Q01381

Which sequence matches the course's logical query-processing pipeline for a SELECT, from checking the request to returning its result?

A. Parsing and semantic analysis → logical translation → query optimization → evaluation → output

B. DDL->DML->query optimizer-> Authentication->output

C. DML->query optimizer-> Authentication-> DDL-> output

D. All of the mentioned

Hint

Separate query representation, strategy selection and execution.

Answer A: A. Parsing and semantic analysis → logical translation → query optimization → evaluation → output

The request is first checked for grammar and meaning, then translated into a logical representation. Optimization selects equivalent operations and physical access methods; evaluation executes that plan and returns the output. Option A follows those roles in order. The other sequences treat DDL and DML language categories as processing stages and place unrelated checks after optimization, so they do not fit.

Why each choice is right or wrong

A - Correct: This sequence follows checking, representation, strategy selection and execution.

B - Incorrect: DDL/DML categories do not form these phases, and checking is misplaced.

C - Incorrect: DDL is not a mandatory post-optimization phase of SELECT processing.

D - Incorrect: Only option A follows the pipeline, so all sequences are not correct.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> DDL is not a mandatory stage before each SELECT, so authentication→DDL→DML permutations do not describe the processing sequence.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 29 | lines 9-20

> 1.6 Query Processing Steps
> • Processing can be divided into 4 :Decomposition, Optimization , Execution and Code generation
> 1.7 Query Decomposition
> ● It is the process of transforming a high level query into a relational algebra query, and to check that the query is syntactically and semantically correct. It Consists of parsing and validation
> Typical stages in query decomposition are:
> i. Analysis: lexical and syntactical analysis of the query (correctness) based on attributes, data type.. ,. Query tree will be built for the query containing leaf node for base relations, one or many non-leaf nodes for relations produced by relational algebra operations and root node for the result of the query. Sequence of operation is from the leaves to the

Fundamental of Database Systems module | PDF page 30 | lines 17-30

> 1.8 Query Optimization
> • Everyone wants the performance of their database to be optimal. In particular, there is often a requirement for a specific query or object that is query based, to run faster.
> • Problem of query optimization is to find the sequence of steps that produces the answer to user request in the most efficient manner, given the database structure.

• The performance of a query is affected by the tables or views that underlies the query and by the complexity of the query.
• Given a request for data manipulation or retrieval, an optimizer will choose an optimal plan for evaluating the request from among the manifold alternative strategies. i.e. there are many ways (access paths) for accessing desired file/record.
• Hence ,DBMS is responsible to pick the best execution strategy based on various considerations(Least amount of I/O and CPU resources.)
• Example: Consider relations r(AB) and s(CD). We require r X s.
• Method 1 :

Question sources

Database Fundamentals practice questions · Question 56

Question 55 | Q01501

In object-oriented database, the keyword "inverse" is used in which of the following?

A. Relationship

B. All choices are correct

C. Class

D. Attribute

Hint

Distinguish schema definitions, queries and opposite navigation directions.

Answer A: Relationship.

An inverse declaration connects two navigation directions of the same association, such as Department.employees and Employee.department. ADB05 explains that it maintains their relationship consistency. It does not declare a class, establish inheritance or name an ordinary scalar attribute. Thus Relationship is the applicable category, and the inclusive option is false.

Why each choice is right or wrong

A - Correct: Inverse pairs the two directions of one relationship and expresses their consistency.

B - Incorrect: The other categories are not all valid uses of the inverse declaration.

C - Incorrect: Inverse is not the keyword that declares a class or establishes a class hierarchy.

D - Incorrect: The declaration links relationship directions rather than declaring an ordinary scalar attribute.

Primary note proof

Study-note section ADB05

> An inverse declaration pairs opposite directions of the same relationship. Conceptually, Department.employees is inverse to Employee.department: when an employee belongs to department D, the corresponding employee appears in D's collection. The declaration expresses relationship consistency; it is not an inheritance keyword or an ordinary scalar attribute. An MCQ calling ODL both a DDL and a language for developing logical schemas offers overlapping true descriptions, so its exact wording matters.

Question sources

Mock exam 1 · Question 17

Question 56 | Q01621

Which statements describe ODL? Select all that apply. In A, 'implement' specifically means choosing or carrying out physical storage implementation, rather than expressing a logical definition.

A. A method to implement a logical schema

B. All choices are correct

C. Used to develop logical schemas

D. A data definition language for OODB

Hint

Distinguish schema definitions, queries and opposite navigation directions.

Answer C, D: C. Used to develop logical schemas; D. A data definition language for OODB

ODL is the schema-definition language for object databases and describes classes or interfaces, attributes, operations and relationships. It is therefore both a data definition language and a language used to develop logical schemas. The question defines implementation in A as physical storage work, which is a different task. Because A is false in that sense, All choices are correct is also false.

Why each choice is right or wrong

A - Incorrect: Physical storage implementation is distinct from ODL logical definition.

B - Incorrect: A is false under the specified implementation meaning.

C - Correct: ODL develops logical object schemas.

D - Correct: ODL is a data definition language for object databases.

Primary note proof

Study-note section ADB05

> Object Definition Language (ODL) is the ODMG schema-definition language for object databases; it describes interfaces/classes, attributes, operations and relationships. Object Query Language (OQL) expresses queries. Host-language bindings connect these definitions and operations to languages such as Java or C++. Defining a logical schema is different from implementing its storage.

Question sources

Mock exam 2 · Question 10

Question 57 | Q01622

What are ACID properties of Transactions?

A. Automatically, Concurrency, Isolation, Durability

B. Atomicity, Consistency, Inconsistent, Durability

C. Atomicity, Consistency, Isolation, Database

D. Atomicity, Consistency, Isolation, Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Atomicity, Consistency, Isolation, Durability

Page 66 lists atomicity, consistency preservation, isolation, and durability/permanency. Only D preserves all four terms.

Why each choice is right or wrong

A - Incorrect: Automatically and concurrency replace atomicity and consistency.

B - Incorrect: Inconsistent replaces isolation.

C - Incorrect: Database replaces durability.

D - Correct: All four names match the listed properties.

Primary note proof

Study-note section ADB06

A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Mock exam 2 · Question 11

Question 58 | Q01623

Which of the following commands is used to save any transaction permanently into the database?

A. Rollback

B. None of the choices are correct

C. Savepoint

D. Commit

Hint

Distinguish completing a transaction from marking an intermediate recovery position.

Answer D: Commit

COMMIT successfully completes the transaction and requests a durable committed outcome. ROLLBACK instead cancels uncommitted changes. SAVEPOINT marks an intermediate position to which later work can be rolled back, retaining earlier work within the same transaction; it neither completes that transaction nor makes the retained changes independently durable. Because COMMIT is offered, the none-of-the-choices alternative is false. D therefore identifies the permanent transaction-completion command.

Why each choice is right or wrong

A - Incorrect: Incorrect: ROLLBACK cancels uncommitted work rather than accepting it permanently.

B - Incorrect: Incorrect: COMMIT is offered and is the requested completion command.

C - Incorrect: Incorrect: SAVEPOINT marks an intermediate rollback position inside the still-open transaction; it does not commit.

D - Correct: Correct: COMMIT completes the transaction successfully and requests a durable outcome.

Primary note proof

Study-note section ADB-S02

> In SQL transaction-block notation, BEGIN starts a block, COMMIT completes it successfully, and ROLLBACK cancels its uncommitted changes. Committing requests a durable transaction outcome; it is distinct from merely finishing the last read or write operation.

Study-note section ADB-S02

> SAVEPOINT marks an intermediate position inside the current transaction. ROLLBACK TO discards changes made after that position while retaining earlier work in the transaction. A savepoint is not an independent durable commit, and rolling back to it does not itself commit the retained changes.

Related reading (context, not primary note proof)

Study-note section ADB06

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 65 | lines 4-7

◦ Committed state -ensures that all the changes done on a record by a transition were done persistently
◦ Failed state happens when a transaction is aborted during its active state or if one of the rechecking is fails

Fundamental of Database Systems module | PDF page 77 | lines 11-13

● BEGIN TRANSACTION statement and end with the COMMIT or ROLLBACK statement.
● SQL transaction Examples:

Fundamental of Database Systems module | PDF page 65 | lines 17-19

• [commit,T]: Records that transaction T has completed successfully, and affirms that its effect can be committed (recorded permanently) to the database.
• [abort,T]: Records that transaction T has been aborted.

Question sources

Mock exam 2 · Question 12

Question 59 | SM0003

Which one of the following is NOT correct about transaction failure?

A. Transactions fail if logical errors are detected

B. Catastrophes can be a cause of transaction failure

C. Concurrency control enforcement avoids transaction failure

D. Transactions fail if disk blocks lose their data

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer C: Concurrency control enforcement avoids transaction failure

The failure list includes logical programming errors, disk blocks losing data, and physical catastrophes. It also explicitly says concurrency enforcement may abort a transaction for serializability or deadlock (p.98). Thus claiming it avoids transaction failure is the incorrect statement.

Why each choice is right or wrong

A - Incorrect: Logical errors are explicitly a cause.

B - Incorrect: Physical problems and catastrophes are explicitly listed.

C - Correct: Concurrency enforcement itself may abort transactions, contradicting the absolute claim.

D - Incorrect: Loss of disk-block data is explicitly listed.

Primary note proof

Study-note section ADB26

Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 98 | lines 4-31

- What causes a Transaction to fail

1. A computer failure (system crash):
A hardware or software error occurs in the computer system during transaction execution. If the hardware crashes, the contents of the computer‘s internal memory may be lost.
2. A transaction or system error:
Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error. In addition, the user may interrupt the transaction during its execution.
3. Exception conditions detected by the transaction:
◦ Certain conditions forces cancellation of the transaction.
◦ For example, data for the transaction may not be found. such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal from that account, to be canceled.
4. Concurrency control enforcement:
The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serializability or because several transactions are in a state of deadlock (see Chapter 2).
5. Disk failure:
Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.
6. Physical problems and catastrophes:
This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, overwriting disks or tapes by mistake
Transaction Manager : Accepts transaction commands from an application, which tell the transaction manager when transactions begin and end, as well as information about the

expectations of the application.
The transaction processor performs the following tasks:

Question sources

MOE Computer Science Exit Exam · Question 1-3

National Computer Science Exit Exam · Question 10

National Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 59

Question 60 | SM0004

Which one of the data updates best describes: 'All modified data items in the cache is written either after transaction ends its execution or after a pre-determined number of transactions have concluded their
execution'?

A. Immediate update

B. In-place update

C. Shadow update

D. Deferred update

Hint

Separate update timing from update location and follow the commit boundary.

Answer D: Deferred update

The deferred-update passage on p.110 almost exactly reproduces the stem: modified cache items are written after a transaction ends or a fixed number finish.

Why each choice is right or wrong

A - Incorrect: Immediate update writes as soon as an item changes in cache.

B - Incorrect: In-place describes overwriting the disk version, not the delay policy.

C - Incorrect: Shadow update writes a separate disk location.

D - Correct: Deferred update has precisely the stated timing.

Primary note proof

Study-note section ADB29

Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

Shadow/copy-on-write update writes a new location while preserving the committed version.

Study-note section ADB29

In-place update overwrites the original location.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Fundamental of Database Systems module | PDF page 111 | lines 2-9

◦ Shadow update:
• The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.
• Multiple version of the same data item can be maintained
• Thus the old value (before image BFIM) and the new value (AFIM) are kept in the disk
• No need of Log for recovery
◦ In-place update: The disk version of the data item is overwritten by the cache

Question sources

MOE Computer Science Exit Exam · Question 1-4

National Computer Science Exit Exam · Question 41

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 60

Question 61 | SM0015

T1 writes X but has not committed. T2 reads that value; T1 then fails and its write is rolled back. Which anomaly did T2 experience?

A. Nonrepeatable-read problem

B. It is dirty read

C. It is an incorrect summary problem

D. It is a lost update problem

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: B. It is dirty read

T2 consumes a value written by another transaction before that transaction commits. The writer's later failure makes this a dirty-read dependency requiring reconsideration of dependent work. A nonrepeatable read involves repeated reads seeing a committed change, while an incorrect summary mixes inconsistent versions in an aggregate. Lost update involves a stale overwrite, which this trace does not describe.

Why each choice is right or wrong

A - Incorrect: No repeated read observing a committed change is described.

B - Correct: T2 reads another transaction's uncommitted write.

C - Incorrect: No aggregate combines inconsistent versions.

D - Incorrect: No stale value is written over another update.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

> ii. The Temporary Update (or Dirty Read) Problem
> ◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
> ◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:

iii. The Incorrect Summary Problem
◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Question sources

MOE Computer Science Exit Exam · Question 4-3

National Computer Science Exit Exam · Question 58

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 61

Question 62 | SM0045

When two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect. Which option correctly identifies this?

A. It is a temporary read problem

B. It is a dirty read problem

C. It is an incorrect summary problem

D. It is a lost update problem

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer D: It is a lost update problem

The lost-update definition on p.68 is the wording in the stem. The other anomalies concern reading a temporary failed update or mixing summary values.

Why each choice is right or wrong

A - Incorrect: Temporary-update/dirty-read requires access to an uncommitted failed update, not merely the interleaving described.

B - Incorrect: Dirty read is the temporary-update anomaly.

C - Incorrect: Incorrect summary concerns aggregate evaluation over mixed versions.

D - Correct: Lost update matches the stated definition.

Primary note proof

Study-note section ADB08

Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 68 | lines 4-7

i. The Lost Update Problem
◦ This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

Fundamental of Database Systems module | PDF page 70 | lines 2-12

ii. The Temporary Update (or Dirty Read) Problem
◦ This occurs when one transaction updates a database item and then the transaction fails for some reason .
◦ The updated item is accessed by another transaction before it is changed back to its original value. Based on the above scenario:
iii. The Incorrect Summary Problem
◦ If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
Example: T1 would like to add the values of A=10, B=20 and C=30. after the values are read by

Question sources

MOE Computer Science Exit Exam · Question 13-3

National Computer Science Exit Exam · Question 40

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 62

Question 63 | SM0046

Which listed policy prevents lock-wait deadlock without comparing transaction-age timestamps? Here 'wait-wait' means allowing every conflicting request to wait with no ordering or cycle restriction.

A. Wait-wait

B. No-wait

C. Wound-wait

D. Wait-die

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: No-wait

No-wait aborts immediately when a lock request conflicts, so transactions cannot form a lock-wait cycle and no age comparison is needed. Wait-die and wound-wait use relative transaction ages to decide who waits or aborts. Under the explicitly defined wait-wait alternative, every conflicting request may wait without a cycle-prevention restriction, so a transaction can hold one item while waiting for another in a cycle. That alternative therefore fails the deadlock-prevention requirement.

Why each choice is right or wrong

A - Incorrect: Incorrect: unrestricted waiting permits a hold-and-wait cycle, so this defined policy does not prevent deadlock.

B - Correct: Correct: immediate abort on a conflicting request prevents lock waits without comparing ages.

C - Incorrect: Incorrect: wound-wait compares requester and holder ages to choose waiting or aborting a younger holder.

D - Incorrect: Incorrect: wait-die compares ages to choose whether the requester waits or aborts.

Primary note proof

Study-note section ADB13

> Fixed resource ordering and conservative 2PL prevent cycles. A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp. A timeout aborts long waits but may mistake slow progress for deadlock. Starvation is indefinite postponement while other work proceeds, without a required cycle. Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Study-note section ADB14

> A smaller timestamp means an older transaction. In wait-die, an older requester may wait for a younger holder; a younger requester needing an older holder aborts itself. In wound-wait, an

older requester aborts the younger holder; a younger requester waits for the older holder. Always identify requester versus holder before applying the names.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 88 | lines 6-28

• Transaction Timestamp(TS(T))
− We can prevent deadlocks by giving each transaction a priority and ensuring that lower priority transactions are not allowed to wait for higher priority transactions (or vice versa).
− One way to assign priorities is to give each transaction a timestamp when it starts up.
− it is a unique identifier given to each transaction based on time in which it is started. i.e if T1 starts before T2 , TS(T1)<TS(T2)
− The lower the timestamp, the higher the transaction's priority, that is, the oldest transaction has the highest priority.
− If a transaction Ti requests a lock and transaction Tj holds a conflicting lock, the lock manager can use one of the following two policies: Wait-die & Wound-wait
● Wait-die
◦ If Ti has higher priority, it is allowed to wait; otherwise it is aborted.
◦ An older transaction is allowed to wait on a younger transaction.
◦ A younger transaction requesting an item held by an older transaction is aborted
◦ If TS(Ti) < TS(Tj), then (Ti older than Tj)Ti is allowed to wait.
◦ Otherwise (Ti younger than Tj)Abort Ti (Ti dies) and restart it later with the same timestamp
T1(ts =10): M is locked , K is requested
T2 (ts =20): K is locked, Z is requested
88

Fundamental of Database Systems module | PDF page 89 | lines 4-13

• Wound-wait
- The opposite of wait-die
- If Ti has higher priority, abort Tj; otherwise Ti waits.
- A younger transaction is allowed to wait on an older one
- An older transaction requesting an item held by a younger transaction preempts the younger transaction by aborting it.
- If TS(Ti) < TS(Tj), then (Ti older than Tj),Abort Tj (Ti wounds Tj) and restart Tj later with the same timestamp
- Otherwise (Ti younger than Tj)Ti is allowed to wait
T2 (ts =20)

Question sources

MOE Computer Science Exit Exam · Question 14-1

National Computer Science Exit Exam · Question 100

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 63

Question 64 | SM0047

A database management system (DBMS) has _____ to control locks.

A. Lock manager

B. Lock table

C. Query optimizer

D. Query processor

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer A: Lock manager

Page 83 distinguishes the manager that manages locks from the table it uses to store lock identities and modes. The controlling component is the lock manager.

Why each choice is right or wrong

A - Correct: This component manages locks.

B - Incorrect: The table stores lock information for the manager.

C - Incorrect: Optimization selects execution strategies, not lock ownership.

D - Incorrect: The named lock-control component is the manager, not query processing.

Primary note proof

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks. S→X is an upgrade, possible only after conflicting other holders leave. X→S is a downgrade. The lock manager decides requests; the lock table stores granted locks and waiting requests. One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy. No double-locking is not a prohibition on shared readers.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB17

> A physical execution plan adds scan/index choices and algorithms.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 83 | lines 2-8

• Lock Manager:
• Managing locks on data
items.
• Lock table:
• Lock manager uses it to store the identify of transaction locking a data
item, the data item, lock mode and pointer to the next data item locked.
One simple way to implement a lock table is through linked list.

Question sources

MOE Computer Science Exit Exam · Question 14-2

National Computer Science Exit Exam · Question 75

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 64

Question 65 | SM0205

In a columnar database, data is stored:

A. Row-wise

B. In a hierarchical structure

C. Column-wise

D. In a network model

Hint

Distinguish operational transactions, analytical workloads and physical storage layout.

Answer C: Column-wise.

ADB41 defines columnar storage as grouping values by column, which avoids reading irrelevant columns and can improve compression for scans. Row-oriented storage instead groups each record's fields. Hierarchical and network models are logical representation alternatives rather than the physical column-oriented layout requested by the stem. Columnar layout alone does not determine the logical data model.

Why each choice is right or wrong

A - Incorrect: Row-wise grouping is row-oriented storage, the contrasting physical layout.

B - Incorrect: A hierarchy describes parent-child organization, not the columnar physical storage arrangement.

C - Correct: Columnar storage groups the values of each column together.

D - Incorrect: A network model concerns linked records rather than this column-oriented physical layout.

Primary note proof

Study-note section ADB41

> OLTP handles frequent operational transactions, such as account updates; OLAP scans and summarizes large datasets for analysis. A warehouse integrates data for historical analysis, often through extraction, transformation and loading. These are workload tendencies: OLTP can retain history and warehouses can be updated.

Study-note section ADB41

> Columnar storage groups values by column, reducing unnecessary column reads and aiding compression in scans; row-oriented storage groups each record's fields.

Study-note section ADB41

> Neither physical layout alone determines whether the logical model is relational.

Related reading (context, not primary note proof)

Study-note section ADB-S01

Question sources

Computer Science Exit Exam · Sene 2016 · Question 2-1

Computer Science Exit Exam practice collection · Question 69

Question 66 | SM0247

An online game wants extra read copies of player reference data near regional users while retaining the same logical data. Which concept specifically creates those additional copies, rather than dividing the only copy into fragments or choosing a model family?

A. Replication

B. NoSQL databases

C. Distributed databases

D. Partitioning

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer A: A. Replication

Replication creates additional copies of selected data and can improve read locality by placing them near users. That directly satisfies the specified regional read-copy requirement. Distribution is the broader arrangement across sites, partitioning divides data into fragments, and NoSQL names a family of data models. Neither the broader architecture nor a model label specifically says that extra copies are created.

Why each choice is right or wrong

A - Correct: Replication creates the required regional read copies.

B - Incorrect: NoSQL selects a model family and does not itself create copies.

C - Incorrect: Distribution is broader than the requested copy-creation policy.

D - Incorrect: Partitioning divides data; by itself it creates no spare copy.

Primary note proof

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once. Copies improve read locality and can improve availability but require storage and update propagation. They can also spread mistakes, so replication does not replace independent backups.

Study-note section ADB36

> Partitioning improves locality and parallelism but alone creates no spare copy.

Study-note section ADB42

> Flexible schema does not mean no schema rules, and NoSQL does not imply no transactions.

Related reading (context, not primary note proof)

Study-note section ADB33

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 3-13

> 6.1 Distributed Database Concepts
> • A transaction can be executed by multiple networked computers in a unified manner.
> • A distributed database (DDB) processes Unit of execution (a transaction) in a distributed manner.
> • A distributed database (DDB) can be defined as :
> - A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
> - The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Fundamental of Database Systems module | PDF page 117 | lines 15-31

> 6.2 Data Replication and Fragmentation: Distributed data storage
> • There are two approaches to store the relation in the distributed database : Replication and Fragmentation
> I. Data Replication
> • The system maintain several identical copies of the relation & store each copy at a

different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science Exit Exam · Sene 2016 · Question 15-3

Computer Science Exit Exam practice collection · Question 67

Question 67 | SM0298

In a generic DBMS component model applied to a MySQL deployment, which specialized role coordinates restoring valid data from stored backups and logs after data loss? Use generic functional roles, not claims about named MySQL engine internals.

A. Buffer Manager

B. Transaction Manager

C. Query Engine

D. Recovery Manager

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer D: D. Recovery Manager

Recovery is the specialized responsibility for restoring valid database state after failure or loss. Backups supply a separate stored copy, and available logs can restore later recoverable changes. The recovery-manager role therefore matches the question's generic component model. Buffer management caches working pages; transaction management is the broader parent responsibility, while the query engine processes requests. This makes no claim about MySQL's internal module names.

Why each choice is right or wrong

A - Incorrect: Buffers cache working data and are not durable backup copies.

B - Incorrect: Transaction management is broader than the specialized restoration role requested.

C - Incorrect: A query engine processes queries rather than coordinating restoration.

D - Correct: The generic recovery role coordinates restoration using durable recovery resources.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Generic recovery-manager terminology should not be mistaken for an exact MySQL component specification.

Study-note section ADB26

> A full backup copies the database to separate storage; scheduled backups plus archived logs support restoration to a chosen recoverable point.

Related reading (context, not primary note proof)

Study-note section ADB07

Study-note section ADB06

Study-note section ADB17

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 99 | lines 2-7

> ✓ Logging: In order to assure durability, every change in the database is logged separately on disk.
> ✓ Log manager initially writes the log in buffers and negotiates with the buffer manager to make sure that buffers are written to disk at appropriate times.
> ✓ Recovery Manager: will be able to examine the log of changes and restore the database to some consistent state.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 32-1

Computer Science Exit Exam practice collection · Question 70

Question 68 | SS102

Which listed sources can contain unstructured free text, images or audio? Select all that apply. Here Google search collections include free-text web pages; RDBMS systems may store text/BLOB attachments; the Apple Employee and Registrar student examples contain only fixed-field structured records.

A. Google search engine

B. RDBMS systems

C. Apple Employee database

D. Registrar student database

Hint

Classify the actual content or pattern being analyzed rather than the storage product.

Answer A, B: A. Google search engine; B. RDBMS systems

Unstructured classification depends on the actual content rather than the storage product. Search collections can contain free text and images, and an RDBMS can store text or binary attachments, so A and B qualify under the stated contents. The employee and registrar examples are explicitly restricted to fixed-field structured records, excluding the otherwise possible attachments that would change their classification.

Why each choice is right or wrong

A - Correct: The specified web collection includes unstructured free text.

B - Correct: The specified relational store may contain unstructured text/BLOB attachments.

C - Incorrect: This example is restricted to structured fixed-field records.

D - Incorrect: This example is restricted to structured fixed-field records.

Primary note proof

Study-note section ADB43

> Structured data follows explicit fields, such as a registrar's relational tables. Unstructured data includes free text, images and audio; web/search collections can include these alongside structured content. A DBMS can also store text or BLOBs, so a storage product does not prove all its contents are structured.

Study-note section ADB43

> Clustering groups similar cases without target labels; classification predicts categories and regression predicts numeric outcomes.

Study-note section ADB43

> If 75 of 100 shoe purchases include socks, confidence(shoes⇒socks)=0.75.

Question sources

Computer Science model exam collection · Question 9

Question 69 | SS123

Business people always remodel their business by closely studying their customers' purchasing habits over the course of their purchase history.
Accordingly, a salesperson at Markato shop noticed that people who always buy shoes will also buy socks with probability of 0.75 and hence decided
to put shoes and socks side by side for ease of accessibility. Which big data analytics technique could this salesperson could employ?

A. Association rule

B. Clustering algorithm

C. Predictive modeling

D. Regression modeling

Hint

Classify the actual content or pattern being analyzed rather than the storage product.

Answer A: Association-rule mining: shoes ⇒ socks has confidence 0.75.

The observation is a conditional co-purchase pattern: among shoe purchases, socks also appear with probability 0.75. ADB43 defines this as the confidence of an association rule and explicitly distinguishes co-occurrence mining from clustering, classification and numeric regression. The store-layout decision uses that rule; it does not change the underlying technique or prove causation.

Why each choice is right or wrong

A - Correct: The rule captures shoe/sock co-occurrence with conditional confidence 0.75.

B - Incorrect: Clustering groups similar cases without target labels rather than directly discovering this co-purchase rule.

C - Incorrect: The stated evidence is an association rule, not a separately specified predictive modeling task.

D - Incorrect: Regression predicts numeric outcomes rather than identifying this purchasing co-occurrence rule.

Primary note proof

Study-note section ADB43

> An association rule X⇒Y describes co-occurrence. Support=count(X and Y)/N; confidence=count(X and Y)/count(X)=P(Y|X). If 75 of 100 shoe purchases include socks, confidence(shoes⇒socks)=0.75. This does not prove causation or specify support without total transaction count. Clustering groups similar cases without target labels; classification predicts categories and regression predicts numeric outcomes. Purchasing co-occurrence is association mining even when its result informs store layout.

Question sources

Computer Science model exam collection · Question 30

Question 70 | SS130

Using this convention, Model Testing means measuring a trained model on held-out test data; Model Evaluation means the subsequent release decision using those results and business success criteria. Which sequence applies before deploying this newly trained model?

A. Model Training, Model Evaluation, Model Testing, Model Deployment

B. Model Training, Model Testing, Model Deployment, Model Evaluation

C. Model Training, Model Testing, Model Evaluation, Model Deployment

D. Model Evaluation, Model Training , Model Testing, Model Deployment

Hint

Identify the objective and the stated meanings of the workflow stages.

Answer C: C. Model Training, Model Testing, Model Evaluation, Model Deployment

Under the stated convention, training first produces the model. Testing then measures it on held-out data, and evaluation uses those results to decide whether business criteria justify deployment. C follows that sequence. Evaluation during model selection can use a different ordering, but this question explicitly assigns evaluation to the subsequent release review, so the other sequences misplace that review.

Why each choice is right or wrong

A - Incorrect: It puts the specified post-test release review before testing.

B - Incorrect: It deploys before the required release review.

C - Correct: Training, held-out testing, release evaluation and deployment follow the defined roles.

D - Incorrect: It evaluates the newly trained model before that model has been trained.

Primary note proof

Study-note section ADB44

> CRISP-DM starts with business understanding: define the decision and success criteria. Then understand data, prepare it, model, evaluate and deploy; the process is iterative. Cleaning, integration and feature selection occur during preparation, not before the business problem exists.

Study-note section ADB44

> If an MCQ asks training→testing→evaluation→deployment, state that convention; another valid workflow can evaluate during model selection before final testing.

Study-note section ADB44

> Within modeling, split data appropriately, train on training data, tune using validation and evaluate once on a held-out test set before deployment.

Question sources

Computer Science model exam collection · Question 37

Question 71 | SS188

Identify the one that comes first in data mining process

A. Business understanding

B. Data integration

C. Data cleaning

D. Data selection

Hint

Identify the objective and the stated meanings of the workflow stages.

Answer A: Business understanding.

ADB44 starts the data-mining workflow by defining the business decision and its success criteria. This establishes why data will be collected and modeled. Integration, cleaning and feature selection prepare data after the problem has been understood; they do not replace the initial business-understanding stage. Thus A precedes the other listed activities.

Why each choice is right or wrong

A - Correct: Business understanding first defines the decision to support and the criteria for success.

B - Incorrect: Data integration prepares sources after the business problem has been identified.

C - Incorrect: Cleaning improves data quality during preparation rather than defining the initial business objective.

D - Incorrect: Selecting relevant data depends on an already understood business problem and modeling objective.

Primary note proof

Study-note section ADB44

> CRISP-DM starts with business understanding: define the decision and success criteria. Then understand data, prepare it, model, evaluate and deploy; the process is iterative. Cleaning, integration and feature selection occur during preparation, not before the business problem exists.

Study-note section ADB44

> Deployment includes monitoring and may trigger another preparation/modeling cycle.

Study-note section ADB44

> If an MCQ asks training→testing→evaluation→deployment, state that convention; another valid workflow can evaluate during model selection before final testing.

Question sources

Computer Science model exam collection · Question 95

Question 72 | AM0162

What is the defining purpose of a transaction in database management?

A. To secure the database from unauthorized access

B. To optimize query performance

C. To group database operations into one logical unit of work

D. To define the structure and organization of a database

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: C. To group database operations into one logical unit of work

A transaction groups database operations into one logical unit of work. The DBMS supports its atomicity, consistency, isolation and durability requirements, including controlled interference with concurrent transactions. Grouping is the defining concept, rather than query tuning, access authorization or schema construction. Read-only work can also form a transaction, so the definition is not limited to operations that change stored data.

Why each choice is right or wrong

A - Incorrect: Authorization governs who may access data rather than defining transaction grouping.

B - Incorrect: Optimization chooses query execution strategies.

C - Correct: A transaction is one logical unit of database work.

D - Incorrect: Schema definition describes data structures rather than a unit of work.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks.

Study-note section ADB11

One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 63 | lines 3-9

3.1 Introduction to Transaction
- A Transaction:
  - Logical unit of database processing that includes one or more access operations (read -retrieval, write - insert or update, delete).
- Examples include ATM transactions, credit card approvals, flight reservations, hotel check-in, phone calls, supermarket scanning, academic registration and billing.
- Transaction boundaries:

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Computer Science MCQ practice collection · Question 162

Question 73 | AM0177

A correctly implemented financial transaction must preserve a declared total-balance invariant when it completes. Which ACID property names preservation of such integrity rules? Focus on the invariant rather than partial execution, concurrent interference or survival after a crash.

A. Atomicity

B. Consistency

C. Isolation

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: B. Consistency

Consistency means that correct transactional work preserves declared integrity rules. Maintaining the total-balance invariant is therefore the consistency requirement. Atomicity addresses whether the effects happen together, isolation controls interference among concurrent transactions and durability preserves committed effects after failure. These properties cooperate, but the revised question identifies invariant preservation as the particular guarantee to name.

Why each choice is right or wrong

A - Incorrect: Atomicity names all-or-none effects rather than invariant preservation itself.

B - Correct: Consistency names preservation of declared integrity rules by correct work.

C - Incorrect: Isolation controls concurrent interference.

D - Incorrect: Durability preserves committed effects after failure.

Primary note proof

Study-note section ADB06

A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 66 | lines 4-13

- Atomicity: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.
- Consistency preservation: A correct execution of the transaction must take the database from one consistent state to another.
- Isolation: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary
- Durability or permanency: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

Example

Question sources

Computer Science MCQ practice collection · Question 177

Question 74 | AM0185

In the object-oriented model, an object is defined as:

A. An instance of a class

B. A collection of attributes

C. A primary key in a database

D. A relationship between tables

Hint

Distinguish an object instance from its class, attribute values and persistent identity.

Answer A: An instance of a class.

ADB01 explicitly distinguishes a class specification from each object instantiated from it. A class defines available attributes and operations, while an object is one individual instance with identity, state and behavior. A collection of attributes omits identity and behavior; a key or an intertable relationship belongs to a different relational description.

Why each choice is right or wrong

A - Correct: An object is an individual instance of a class with its own identity and state.

B - Incorrect: Attributes describe state but omit the identity and behavior that also characterize an object.

C - Incorrect: A database primary key is an identifier constraint, not the definition of an object instance.

D - Incorrect: A relationship between tables is a relational association rather than an object instance.

Primary note proof

Study-note section ADB01

> An object combines identity, state and behavior. A class specifies the attributes and operations of its instances; an object is one instance. The object identifier (OID) distinguishes objects independently of their attribute values and remains stable over that object's lifetime. Two objects can have equal values and different identities; changing a name does not create a new identity.

Study-note section ADB01

> Complex CAD or multimedia structures can suit an object model, but many-to-many relationships alone do not require abandoning relational design.

Study-note section ADB01

> An object-relational DBMS extends a relational system with richer types; ORM maps programming objects into relational tables.

Question sources

Computer Science MCQ practice collection · Question 185

Question 75 | AM0186

In the object-oriented model, encapsulation refers to:

A. Combining multiple classes into a single entity

B. Hiding the internal details of an object

C. Establishing relationships between objects

D. Defining the inheritance hierarchy

Hint

Separate the object boundary, the exposed contract and links between objects.

Answer B: Hiding and controlling access to internal object details.

ADB02 describes encapsulation as grouping state with operations and controlling access to implementation details. A withdrawal method can enforce rules while hiding storage. B states the relevant information-hiding aspect. Combining classes, associating objects and defining subtype hierarchies are distinct activities; none describes the encapsulation boundary around an object's state and operations.

Why each choice is right or wrong

A - Incorrect: Encapsulation groups an object's state and operations, not multiple classes into one entity.

B - Correct: Encapsulation controls access to the internal representation behind the object's operations.

C - Incorrect: Links between objects are associations rather than encapsulation itself.

D - Incorrect: Superclass/subclass structure concerns inheritance rather than information-hiding boundaries.

Primary note proof

Study-note section ADB02

> Encapsulation groups state with operations and controls access to implementation details. Account.withdraw(amount) can enforce a balance rule while hiding its storage. Abstraction presents the relevant contract-what withdrawal does-without exposing how it is implemented. These ideas overlap in information hiding, but combining state and methods specifically describes encapsulation.

Study-note section ADB02

> A class defines structure and behavior; an attribute describes state; a method implements an operation.

Study-note section ADB02

It is different from inheritance, which means an is-a relationship between types.

Question sources

Computer Science MCQ practice collection · Question 186

Question 76 | AM0187

In the object-oriented model, inheritance allows:

A. Objects to have multiple parents

B. Objects to inherit attributes and behaviors from a superclass

C. Objects to be created dynamically at runtime

D. Objects to establish relationships with other objects

Hint

Distinguish the subtype relationship from use through a common interface.

Answer B: Inherit applicable attributes and behavior from a superclass.

ADB03 defines inheritance as deriving a subtype from a superclass and reusing its applicable attributes and behavior. B captures that idea. It does not inherently require multiple parents, dynamically create objects, or merely link unrelated objects. Those alternatives concern particular inheritance forms, instantiation or association rather than the general meaning of inheritance.

Why each choice is right or wrong

A - Incorrect: Multiple parents are not a general requirement or guarantee of inheritance.

B - Correct: A subtype inherits applicable superclass attributes and behavior.

C - Incorrect: Dynamic object creation is instantiation, not the capability defined by inheritance.

D - Incorrect: Relationships between separate objects are associations rather than the is-a subtype relationship.

Primary note proof

Study-note section ADB03

Inheritance derives a subtype from a superclass, reusing applicable attributes and behavior. GraduateStudent is a Student; association instead says Student belongs to Department. EER teaches inherited attributes and relationships, while a programming object model also specifies behavior.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 13 | lines 5-9

Subclasses and super classes can be thought as useful ways to describe similarities in the entities represented. Because subclasses inherit their attributes and relationships from the superclass, the
database designer can represent complex, hierarchical interdependencies (inheritance) within the
system being described.
Specialization is the process of dividing a parent-level entity (super-class) into narrower

Question sources

Computer Science MCQ practice collection · Question 187

Question 77 | AM0188

Which concept in the object-oriented model allows different classes to share the same method name but with different implementations?

A. Polymorphism

B. Abstraction

C. Encapsulation

D. Inheritance

Hint

Distinguish the subtype relationship from use through a common interface.

Answer A: Polymorphism.

ADB03 illustrates Shape.area dispatching to Circle.area or Rectangle.area according to the actual object. The common operation name has different compatible implementations, which is polymorphism. Inheritance can provide the subtype structure used by that mechanism, but is not itself the runtime dispatch behavior. Abstraction and encapsulation concern contracts and implementation access instead.

Why each choice is right or wrong

A - Correct: Polymorphism lets the same interface operation invoke different implementations for compatible object types.

B - Incorrect: Abstraction exposes a relevant contract rather than selecting among differing method implementations.

C - Incorrect: Encapsulation controls internal access rather than providing type-dependent method dispatch.

D - Incorrect: Inheritance supplies a structural subtype relation; polymorphism describes the differing implementations through a common interface.

Primary note proof

Study-note section ADB03

> Subtype polymorphism lets a superclass/interface reference denote objects of different compatible subclasses. The same operation can dispatch to different implementations: Shape.area() calls Circle.area() or Rectangle.area() according to the actual object. This is not arbitrary conversion of an object into an unrelated class, nor dynamic class mutation. Inheritance establishes a subtype relationship; polymorphism uses that relationship through a common interface. Distinguish inheritance as a structural relationship from polymorphism as a use of that relationship at runtime.

Question sources

Computer Science MCQ practice collection · Question 188

Question 78 | AM0189

The process of combining data and methods into a single entity is known as:

A. Inheritance

B. Polymorphism

C. Abstraction

D. Encapsulation

Hint

Separate the object boundary, the exposed contract and links between objects.

Answer D: Encapsulation.

The question specifically asks about placing data and methods inside one entity. ADB02 identifies that grouping as encapsulation, along with controlled access to implementation details. Abstraction concerns what the contract exposes; inheritance derives compatible subtypes; polymorphism invokes different implementations through a common interface. None of those replaces the state-and-method grouping named in the stem.

Why each choice is right or wrong

A - Incorrect: Inheritance derives a subtype and reuses superclass features rather than naming state/method grouping.

B - Incorrect: Polymorphism varies implementations behind a compatible common interface rather than grouping data and methods.

C - Incorrect: Abstraction presents a relevant contract; the specific grouping asked for is encapsulation.

D - Correct: Encapsulation packages state and operations together within the object boundary.

Primary note proof

Study-note section ADB02

> Encapsulation groups state with operations and controls access to implementation details. Account.withdraw(amount) can enforce a balance rule while hiding its storage. Abstraction presents the relevant contract-what withdrawal does-without exposing how it is implemented. These ideas overlap in information hiding, but combining state and methods specifically describes encapsulation.

Study-note section ADB02

> It is different from inheritance, which means an is-a relationship between types.

Study-note section ADB02

> A class defines structure and behavior; an attribute describes state; a method implements an operation.

Question sources

Computer Science MCQ practice collection · Question 189

Question 79 | AM0190

The object-oriented model supports the concept of association, which represents:

A. Inheritance relationships between classes

B. Shared attributes among objects

C. The interactions between objects

D. Data duplication in the database

Hint

Separate the object boundary, the exposed contract and links between objects.

Answer C: Links and interactions between objects.

ADB02 defines association as links between objects, commonly implemented with references, and describes methods invoking operations on other objects. C is the closest description of that collaboration. An association is not an is-a inheritance hierarchy, shared attribute definitions or duplication. For example, Student belonging to Department links objects without making either a subtype of the other.

Why each choice is right or wrong

A - Incorrect: Inheritance is an is-a relationship between types, distinct from association links between objects.

B - Incorrect: Having similar attributes does not itself define an association between particular objects.

C - Correct: Associations link objects and support their interactions or collaboration through references.

D - Incorrect: Association represents relationships rather than duplicated copies of the same database facts.

Primary note proof

Study-note section ADB02

> An association links objects, often through references; Department.employees can reference Employee objects. It is different from inheritance, which means an is-a relationship between types. Methods can invoke operations on other objects, supporting collaboration. A class defines structure and behavior; an attribute describes state; a method implements an operation. Do not treat each of these as an interchangeable name for the whole object.

Question sources

Computer Science MCQ practice collection · Question 190

Question 80 | AM0191

Circle and Rectangle already inherit from Shape. A Shape reference can hold either object and dispatch area() to its actual implementation at runtime. Which concept names this common-interface use, rather than the structural subtype relationship?

A. Polymorphism

B. Inheritance

C. Abstraction

D. Encapsulation

Hint

Distinguish the subtype relationship from use through a common interface.

Answer A: A. Polymorphism

Subtype polymorphism allows a common superclass or interface reference to denote different compatible subclass objects. Dispatching area() according to whether the actual object is a Circle or Rectangle is that runtime use. Inheritance establishes the subtype relationship already given in the stem. Abstraction exposes a contract and encapsulation controls representation, but neither names this particular substitutable runtime dispatch.

Why each choice is right or wrong

A - Correct: Using one superclass interface with subclass-specific dispatch is subtype polymorphism.

B - Incorrect: Inheritance is the established structural prerequisite, not the runtime use requested.

C - Incorrect: Abstraction describes a contract rather than the dispatch behavior.

D - Incorrect: Encapsulation controls internal representation and access.

Primary note proof

Study-note section ADB03

> Subtype polymorphism lets a superclass/interface reference denote objects of different compatible subclasses. The same operation can dispatch to different implementations: Shape.area() calls Circle.area() or Rectangle.area() according to the actual object. This is not arbitrary conversion of an object into an unrelated class, nor dynamic class mutation. Inheritance establishes a subtype relationship; polymorphism uses that relationship through a common interface. Distinguish inheritance as a structural relationship from polymorphism as a use of that relationship at runtime.

Study-note section ADB02

> Encapsulation groups state with operations and controls access to implementation details.

Study-note section ADB02

> These ideas overlap in information hiding, but combining state and methods specifically describes encapsulation.

Question sources

Computer Science MCQ practice collection · Question 191

Question 81 | AM0192

What is the purpose of abstraction in the object-oriented model?

A. To combine multiple classes into a single entity

B. To hide the internal details of an object

C. To establish relationships between objects

D. To define the attributes and behaviors of a class

Hint

Separate the object boundary, the exposed contract and links between objects.

Answer B: Hide implementation details while exposing a relevant contract.

ADB02 explains abstraction through an operation's relevant contract: what withdrawal does is exposed without showing how storage implements it. B captures hiding those irrelevant internal details. This overlaps encapsulation, but abstraction emphasizes the view or contract. Combining classes, creating associations or listing class members is not its defining purpose.

Why each choice is right or wrong

A - Incorrect: Abstraction does not mean merging several classes into one entity.

B - Correct: Abstraction exposes relevant behavior while hiding implementation details from the client view.

C - Incorrect: Establishing links between objects is association rather than abstraction's defining purpose.

D - Incorrect: A class specifies attributes and behavior; abstraction chooses the relevant contract exposed to users.

Primary note proof

Study-note section ADB02

> Encapsulation groups state with operations and controls access to implementation details. Account.withdraw(amount) can enforce a balance rule while hiding its storage. Abstraction presents the relevant contract-what withdrawal does-without exposing how it is implemented. These ideas overlap in information hiding, but combining state and methods specifically describes encapsulation.

Study-note section ADB03

EER teaches inherited attributes and relationships, while a programming object model also specifies behavior.

Study-note section ADB02

An association links objects, often through references; Department.employees can reference Employee objects.

Question sources

Computer Science MCQ practice collection · Question 192

Question 82 | AM0193

The concept of polymorphism allows objects to:

A. Inherit attributes from multiple classes

B. Change their class dynamically at runtime

C. Access private methods of other objects

D. Be treated as objects of different classes

Hint

Distinguish the subtype relationship from use through a common interface.

Answer D: Use an object through a compatible superclass or interface type.

ADB03 allows a superclass or interface reference to denote compatible subclass objects. D is correct in that sense: the same object can be treated through its actual type and a compatible supertype. This does not permit arbitrary conversion to unrelated classes. Multiple inheritance, runtime class mutation and access to another object's private methods are different concepts and are not granted by polymorphism.

Why each choice is right or wrong

A - Incorrect: Inheriting from multiple classes is multiple inheritance, not the operation described by polymorphism.

B - Incorrect: Polymorphism does not change the actual class of an object dynamically.

C - Incorrect: Polymorphism does not bypass private-access restrictions on other objects.

D - Correct: An object can be used through compatible superclass/interface types; this does not mean arbitrary unrelated classes.

Primary note proof

Study-note section ADB03

> Subtype polymorphism lets a superclass/interface reference denote objects of different compatible subclasses. The same operation can dispatch to different implementations: Shape.area() calls Circle.area() or Rectangle.area() according to the actual object. This is not arbitrary conversion of an object into an unrelated class, nor dynamic class mutation. Inheritance establishes a subtype relationship; polymorphism uses that relationship through a common interface. Distinguish inheritance as a structural relationship from polymorphism as a use of that relationship at runtime.

Question sources

Computer Science MCQ practice collection · Question 193

Question 83 | AM0194

Which concept in the object-oriented model defines the structure and behavior of objects?

A. Class

B. Object

C. Attribute

D. Method

Hint

Distinguish an object instance from its class, attribute values and persistent identity.

Answer A: Class.

A class is the specification that defines attributes and operations for its instances, as ADB01 states. An object is an instance conforming to that specification. An attribute accounts for part of state and a method for one operation, but neither individually defines the whole structure and behavior of the object type.

Why each choice is right or wrong

A - Correct: A class specifies the structure and available operations for its object instances.

B - Incorrect: An object is one instance of the specification rather than the specification itself.

C - Incorrect: An attribute represents part of an object's state rather than its whole structure and behavior.

D - Incorrect: A method implements an operation but does not alone define the complete object type.

Primary note proof

Study-note section ADB01

> An object combines identity, state and behavior. A class specifies the attributes and operations of its instances; an object is one instance. The object identifier (OID) distinguishes objects independently of their attribute values and remains stable over that object's lifetime. Two objects can have equal values and different identities; changing a name does not create a new identity.

Study-note section ADB01

> Complex CAD or multimedia structures can suit an object model, but many-to-many relationships alone do not require abandoning relational design.

Study-note section ADB01

> An object-relational DBMS extends a relational system with richer types; ORM maps programming objects into relational tables.

Question sources

Computer Science MCQ practice collection · Question 194

Question 84 | AM0195

Which recovery method aims to restore the database to a previous consistent state by undoing the effects of incomplete or failed transactions?

A. Rollback recovery

B. Redo recovery

C. Checkpoint recovery

D. Restart recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A: Rollback recovery

Undo restores before images and removes after images; failed transactions that updated the database must roll back (p.107). This directly answers the specific action, although undo may be one phase of broader crash recovery.

Why each choice is right or wrong

A - Correct: Rollback is the direct undo operation.

B - Incorrect: Redo restores after images instead.

C - Incorrect: Checkpoints synchronize database and log; they do not name the undo action.

D - Incorrect: A restart can include undo, but is broader than the requested failed-transaction reversal.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 195

Question 85 | AM0196

Which recovery method involves reapplying the changes made by committed transactions from a log file to restore the database?

A. Redo recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A: Redo recovery

If a transaction committed but its buffers did not reach disk, recovery redoes its updates for durability (p.103). Redo restores after images (p.107), whose log records are required before commit (p.104).

Why each choice is right or wrong

A - Correct: Redo is exactly reapplying logged committed updates.

B - Incorrect: A checkpoint reduces the amount of recovery work; it is not the replay operation.

C - Incorrect: Restart recovery can invoke redo but names a larger process.

D - Incorrect: A backup restores a stored database copy, not this specific logged replay.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB30

> A checkpoint is neither a full backup nor an automatic commit of active transactions.

Related reading (context, not primary note proof)

Study-note section ADB26

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 104 | lines 19-25

• For Undo: Before a data item‘s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
• For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
• Standard Recovery Terminology

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 196

Question 86 | AM0197

Which recovery method involves creating checkpoints at regular intervals to save the current state of the database and transaction logs?

A. Checkpoint recovery

B. Redo recovery

C. Rollback recovery

D. Restart recovery

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer A: Checkpoint recovery

Checkpointing synchronizes database and log (p.105); its listed steps force modified buffers to disk, write a checkpoint record, and save the log (p.107). This is not necessarily a full database backup.

Why each choice is right or wrong

A - Correct: The named synchronization procedure matches.

B - Incorrect: Redo reapplies after images, not periodic synchronization.

C - Incorrect: Rollback restores before images.

D - Incorrect: Restart follows failure, whereas checkpoints are established during operation.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB27

> To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage.

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 197

Question 87 | AM0198

Which recovery method involves restarting the system and recovering the database by applying the necessary redo and undo operations?

A. Restart recovery

B. Checkpoint recovery

C. Redo recovery

D. Rollback recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer A: Restart recovery

Recovery from a non-catastrophic crash may combine undo and redo (pp.103,107). A is the best descriptive name for restarting and performing that combined recovery, although the module does not define four mutually exclusive method classes under these exact option names.

Why each choice is right or wrong

A - Correct: Restart recovery describes the complete post-crash recovery process stated in the stem.

B - Incorrect: Checkpointing supplies a saved starting boundary rather than the whole restart process.

C - Incorrect: Redo supplies only the forward part of the stated combination.

D - Incorrect: Rollback supplies only the undo part.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> These are cooperating mechanisms, not four mutually exclusive complete alternatives.

Study-note section ADB27

> Redo installs logged after-images when necessary, usually forward.

Related reading (context, not primary note proof)

Study-note section ADB30

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Fundamental of Database Systems module | PDF page 110 | lines 5-13

◦ Repeating history during redo:
• ARIES will retrace all actions of the database system prior to the crash to reconstruct the database state when the crash occurred.
• Transaction that were un committed at the time of crash are undone
◦ Logging changes during undo:

• It will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.
Data Update : Four types

Question sources

Computer Science MCQ practice collection · Question 198

Question 88 | AM0199

Which recovery method involves restoring the database from a backup copy taken prior to the failure?

A. Backup recovery

B. Rollback recovery

C. Redo recovery

D. Restart recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer A: Backup recovery

For catastrophic damage the module restores a past database copy from backup storage and reconstructs committed operations from the backup log (p.103). A names restoration of that backup copy; a backup alone may need subsequent log replay.

Why each choice is right or wrong

A - Correct: Restoration from a stored copy is the stated backup operation.

B - Incorrect: Rollback reverses updates using before images, not a backup copy.

C - Incorrect: Redo replays logged changes, which may supplement backup restoration.

D - Incorrect: Restart names an overall recovery process rather than its backup-copy source.

Primary note proof

Study-note section ADB26

Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB27

Redo installs logged after-images when necessary, usually forward.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Question sources

Computer Science MCQ practice collection · Question 199

Question 89 | AM0200

Which recovery method is used to recover the database after a media failure, such as a disk crash?

A. Backup recovery

B. Redo recovery

C. Checkpoint recovery

D. Rollback recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer A: Backup recovery

Disk-head crash is media failure (p.97). Where that destroys database contents, the catastrophic-recovery strategy restores a past backup and then reconstructs committed operations from the backup log (p.103).

Why each choice is right or wrong

A - Correct: A backup supplies contents lost in the media failure.

B - Incorrect: Redo can advance a restored copy but alone assumes data pages available to update.

C - Incorrect: A checkpoint is not a replacement copy of a destroyed database.

D - Incorrect: Undo cannot recreate arbitrary destroyed pages by reversing incomplete transactions alone.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 97 | lines 4-13

> i. Purpose of Database Recovery
> ◦ To bring the database into the last consistent state, which existed prior to the failure.
> ◦ To preserve transaction properties (Atomicity &and Durability).
> • The recovery manager of a DBMS is responsible to ensure
> • atomicity by undoing the action of transaction that do not commit
> • Durability by making sure that all actions of committed transaction survive system crash

ii. Types of Failure
◦ The database may become unavailable for use due to :

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Question sources

Computer Science MCQ practice collection · Question 200

Question 90 | AM0201

A crash-recovery system has a long log history and wants to reduce the log-search work compared with scanning all history. Which method adds durable recovery-state records for that purpose?

A. Redo recovery

B. Checkpoint recovery

C. Backup recovery

D. Rollback recovery

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer B: B. Checkpoint recovery

Checkpoints record durable recovery state so recovery need not inspect all old history. That is the requested mechanism for reducing log-search work. Redo reinstalls missing changes, rollback removes incomplete effects and backup recovery starts from a separate copy. They may cooperate with checkpoints, but their defining actions do not name the added recovery-state boundary specified in this question.

Why each choice is right or wrong

A - Incorrect: Redo applies logged after-images rather than defining the recovery-state boundary.

B - Correct: Checkpoint information reduces how much old history must be searched.

C - Incorrect: A backup is a separate stored copy rather than the stated history-bounding record.

D - Incorrect: Rollback reverses incomplete effects rather than adding this recovery-state boundary.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

> • Recovery from catastrophic
> • If there is extensive damage to the wide portion of the database
> • This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
> • Recovery from non-catastrophic failure
> • When the database is not physically damaged but has be come in

consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 201

Computer Science MCQ practice collection · Question 211

Question 91 | AM0202

Which recovery method is focused on ensuring that all committed changes are durable and can survive system failures?

A. Redo recovery

B. Restart recovery

C. Rollback recovery

D. Checkpoint recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A: Redo recovery

Durability requires committed effects to survive crashes (p.97). The module specifically redoes committed updates lost between commit and buffer flushing (p.103), making redo the direct mechanism asked for.

Why each choice is right or wrong

A - Correct: Redo reapplies the missing committed updates.

B - Incorrect: Restart is a broader process that can invoke redo and undo.

C - Incorrect: Rollback preserves atomicity by removing uncommitted effects.

D - Incorrect: Checkpoints reduce replay work but are not the committed-update replay operation.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Related reading (context, not primary note proof)

Study-note section ADB30

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 97 | lines 4-13

i. Purpose of Database Recovery
◦ To bring the database into the last consistent state, which existed prior to the failure.
◦ To preserve transaction properties (Atomicity &and Durability).
• The recovery manager of a DBMS is responsible to ensure
• atomicity by undoing the action of transaction that do not commit
• Durability by making sure that all actions of committed transaction survive system crash
ii. Types of Failure
◦ The database may become unavailable for use due to :

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Question sources

Computer Science MCQ practice collection · Question 202

Question 92 | AM0203

Which recovery method uses the transaction log to identify the changes made by transactions and undo the effects of incomplete or failed transactions?

A. Rollback recovery

B. Redo recovery

C. Checkpoint recovery

D. Restart recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A: Rollback recovery

Before images logged under WAL permit undo of failed transaction effects (p.104); rollback restores them (p.107). The stem specifies this particular operation rather than all crash-recovery phases.

Why each choice is right or wrong

A - Correct: Rollback is the direct log-based undo operation.

B - Incorrect: Redo restores new values instead of removing failed updates.

C - Incorrect: Checkpointing records a synchronization boundary.

D - Incorrect: Restart can include rollback, but is broader than the named reversal.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 104 | lines 19-25

• For Undo: Before a data item‘s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
• For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
• Standard Recovery Terminology

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 203

Question 93 | AM0204

Which recovery method is used to recover the database after a system crash or power failure?

A. Restart recovery

B. Backup recovery

C. Redo recovery

D. Checkpoint recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer A: Restart recovery

For a recoverable system crash, recovery uses undo/redo to restore consistency (pp.103,107). A is the broad descriptive answer for restarting the failed system. If power failure also destroys media, backup restoration may additionally be required; the question supplies no such destruction.

Why each choice is right or wrong

A - Correct: Restart recovery denotes the overall post-crash process.

B - Incorrect: Backup is required for damaged/lost storage, which is not specified here.

C - Incorrect: Redo alone may be insufficient when incomplete transactions reached disk.

D - Incorrect: A checkpoint assists recovery but does not itself execute the full crash recovery.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 98 | lines 4-31

- What causes a Transaction to fail

1. A computer failure (system crash):
A hardware or software error occurs in the computer system during transaction execution. If the hardware crashes, the contents of the computer‘s internal memory may be lost.
2. A transaction or system error:
Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error. In addition, the user may interrupt the transaction during its execution.
3. Exception conditions detected by the transaction:
◦ Certain conditions forces cancellation of the transaction.
◦ For example, data for the transaction may not be found. such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal from that account, to be canceled.
4. Concurrency control enforcement:
The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serializability or because several transactions are in a state of deadlock (see Chapter 2).
5. Disk failure:
Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.
6. Physical problems and catastrophes:
This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, overwriting disks or tapes by mistake
Transaction Manager : Accepts transaction commands from an application, which tell the transaction manager when transactions begin and end, as well as information about the expectations of the application.
The transaction processor performs the following tasks:

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore

to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 204

Question 94 | AM0205

What is the primary goal of database recovery after a failure?

A. Restoring the database to a previous state

B. Preventing future failures from occurring

C. Optimizing database performance

D. Enforcing data integrity constraints

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer A: Restoring the database to a previous state

The explicit purpose is bringing the database into its last consistent state before failure (p.97). A is correct only with this consistency qualification; restoring an arbitrary earlier corrupt state is not the goal.

Why each choice is right or wrong

A - Correct: Read previous state as the last consistent state, as stated.

B - Incorrect: Recovery handles failures rather than preventing all future ones.

C - Incorrect: Performance is not the stated recovery purpose.

D - Incorrect: Integrity is relevant, but constraint enforcement is not the direct restoration objective stated here.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB26

> Generic recovery-manager terminology should not be mistaken for an exact MySQL component specification.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 97 | lines 4-13

> i. Purpose of Database Recovery
> ◦ To bring the database into the last consistent state, which existed prior to the failure.
> ◦ To preserve transaction properties (Atomicity &and Durability).
> • The recovery manager of a DBMS is responsible to ensure
> • atomicity by undoing the action of transaction that do not commit
> • Durability by making sure that all actions of committed transaction survive system crash
> ii. Types of Failure
> ◦ The database may become unavailable for use due to :

Question sources

Computer Science MCQ practice collection · Question 205

Question 95 | AM0206

Which recovery method involves periodically making a copy of the entire database onto a separate storage medium?

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer D: Backup recovery

A full backup copies the database to separate storage so it can later be restored, often followed by replay of archived logs. Rollback reverses failed transaction work; restart recovery restores a valid state after a crash. A checkpoint records recovery state to reduce log-search and recovery work and is not a full independent database copy. The question's periodic separate copy is therefore the backup mechanism, represented by backup recovery.

Why each choice is right or wrong

A - Incorrect: Incorrect: rollback reverses transaction effects rather than making an independent database copy.

B - Incorrect: Incorrect: checkpointing records recovery state and limits recovery work; it is not a full backup.

C - Incorrect: Incorrect: restart recovery is the crash-recovery process, not the periodic creation of a separate copy.

D - Correct: Correct: backup recovery relies on a stored independent database copy.

Primary note proof

Study-note section ADB26

> A full backup copies the database to separate storage; scheduled backups plus archived logs support restoration to a chosen recoverable point.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB30

> A checkpoint is neither a full backup nor an automatic commit of active transactions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Question sources

Computer Science MCQ practice collection · Question 206

Question 96 | AM0207

Which recovery method involves undoing the effects of incomplete or failed transactions?

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A: Rollback recovery

The module calls undoing a failed transaction rollback and specifies restoring before images (p.107). A therefore directly names removal of incomplete effects.

Why each choice is right or wrong

A - Correct: Rollback is the named undo operation.

B - Incorrect: Checkpoint is synchronization, not transaction reversal.

C - Incorrect: Restart can include undo but is the wider process.

D - Incorrect: Backup restores a past copy instead of directly reversing this transaction.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB30

> A checkpoint is neither a full backup nor an automatic commit of active transactions.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

> • Recovery from catastrophic
> • If there is extensive damage to the wide portion of the database

• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 207

Question 97 | AM0208

What is the purpose of a checkpoint in the recovery process?

A. To save the current state of the database

B. To recover from a system crash

C. To log all transaction operations

D. To enforce data integrity constraints

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer A: To save the current state of the database

A checkpoint synchronizes database and log by forcing modified buffers and recording the checkpoint (pp.105,107), saving a durable recovery boundary and reducing later work. It is not necessarily a full snapshot copy.

Why each choice is right or wrong

A - Correct: Saving synchronized current state best matches the listed procedure.

B - Incorrect: Crash recovery uses checkpoints, but this is a later use rather than checkpoint creation itself.

C - Incorrect: Logging every operation is the transaction log's role, not checkpoint's distinguishing purpose.

D - Incorrect: Constraint enforcement is not the checkpoint procedure.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB07

> Insert and delete also modify data; sorting is a query-processing operation rather than a primitive item update.

Study-note section ADB07

> Real systems may use physical or logical records, and need not log every read.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 208

Question 98 | AM0209

In a log-based crash restart that uses durable checkpoint state plus relevant transaction logs, which listed labels describe the checkpoint-assisted technique and its overall restoration process? Select all that apply.

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer B, C: B. Checkpoint recovery; C. Restart recovery

Checkpoint-assisted recovery uses recorded recovery state to limit the necessary log history. Restart recovery is the overall process that restores a valid state after the crash while preserving durable commits and removing incomplete effects. Both labels describe the specified procedure at different levels. Rollback names only undo work, and backup recovery would require starting from a separate stored copy.

Why each choice is right or wrong

A - Incorrect: Rollback specifically undoes incomplete effects rather than naming the checkpoint-plus-log restart procedure.

B - Correct: Checkpoint-assisted recovery uses durable checkpoint information with the relevant log history.

C - Correct: Restart recovery can use checkpoints and transaction logs to reconstruct valid state.

D - Incorrect: Backup recovery starts from a separate database copy, which the description does not mention.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

> Check pointing
> • Log file is used to recover failed DB but we may not know how far back in the

log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 209

Question 99 | AM0210

Which overall crash-recovery process preserves durable committed work and removes incomplete effects after a system failure, leaving any business-level retry of formerly active transactions to a new controlled attempt?

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer C: C. Restart recovery

Restart recovery restores a valid database state after a crash. It preserves durable committed effects and removes incomplete effects using the required recovery mechanisms. Transactions active at failure are not automatically rerun and committed; a business retry is a separate controlled attempt. Rollback and checkpoint processing can be components, while backup recovery starts from another stored copy when that is needed.

Why each choice is right or wrong

A - Incorrect: Rollback names undo of incomplete work rather than the entire restart process.

B - Incorrect: Checkpoint information guides recovery rather than naming its whole process.

C - Correct: Restart recovery is the overall crash-restoration process described.

D - Incorrect: Backup restoration starts from a separate copy, which this scenario does not require.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Related reading (context, not primary note proof)

Study-note section ADB32

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 105 | lines 25-36

> Check pointing
> • Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
> • Checkpoint is a Point of synchronization between database and log file.
> • Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.

• When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 108 | lines 24-29

• During recovery, all transactions of the commit table are redone and all transactions of active tables are ignored since none of their AFIMs reached the database.
• It is possible that a commit table transaction may be redone twice but this does not create any inconsistency because of a redone is —idempotent‖,
• that is, one redone for an AFIM is equivalent to multiple redone for the same AFIM.

Fundamental of Database Systems module | PDF page 110 | lines 14-30

◦ Deferred Update:
• All transaction updates are recorded in the local workspace (cache)
• All modified data items in the cache is then written after a transaction ends its execution or after a fixed number of transactions have completed their execution.
• During commit the updates are first recorded on the log and then on the database
• If a transaction fails before reaching its commit point undo is not needed because it didn‘t change the database yet
• If a transaction fails after commit (writing on the log) but before finishing saving to the data base redoing is needed from the log
◦ Immediate Update:
• As soon as a data item is modified in cache, the disk copy is updated.
• These update are first recorded on the log and on the disk by force writing, before the database is updated
• If a transaction fails after recording some change to the database but before reaching its commit point , this will be rolled back

Fundamental of Database Systems module | PDF page 110 | lines 5-13

◦ Repeating history during redo:
• ARIES will retrace all actions of the database system prior to the crash to reconstruct the database state when the crash occurred.
• Transaction that were un committed at the time of crash are undone
◦ Logging changes during undo:
• It will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.
Data Update : Four types

Question sources

Computer Science MCQ practice collection · Question 210

Question 100 | AM0212

After a power failure loses volatile memory but leaves persistent storage intact, which term names the overall process that restores a valid database while preserving durable commits and removing incomplete effects? Choose the umbrella process rather than one component.

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer C: C. Restart recovery

Restart recovery is the overall restoration process following a system crash or power loss when persistent storage survives. It preserves durable commits and removes incomplete effects using the appropriate logged recovery work. Rollback may supply undo and checkpoints can shorten log searches, but these are cooperating mechanisms within the process. Backup restoration addresses a separate stored copy and is not required by this stated failure.

Why each choice is right or wrong

A - Incorrect: Rollback is an undo component rather than the overall process.

B - Incorrect: Checkpoint-assisted work guides restart rather than naming its umbrella process.

C - Correct: Restart recovery names the complete restoration after this system crash.

D - Incorrect: The stated intact persistent storage does not call for restoring a separate backup.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB30

Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Related reading (context, not primary note proof)

Study-note section ADB27

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 212

Question 101 | AM0213

In a conventional transaction-log system, which listed mechanisms use logged updates or checkpoint/log state for undo or restart? Select all that apply. Backup recovery in D means restoring only a standalone full backup, without replaying archived logs.

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer A, B, C: A. Rollback recovery; B. Checkpoint recovery; C. Restart recovery

Logged rollback uses before-images to remove incomplete effects. Checkpoint-assisted recovery combines durable recovery state with relevant log history, and log-based restart uses that history to preserve commits and remove incomplete work. All three fit the specified model. A standalone backup restore instead copies a separately stored database image; the question excludes archived-log replay from D so that this alternative does not overlap.

Why each choice is right or wrong

A - Correct: The logged rollback uses before-images for undo.

B - Correct: Checkpoint-assisted recovery uses recovery-state records and relevant log history.

C - Correct: The specified log-based restart uses durable transaction history.

D - Incorrect: A standalone image restore without log replay does not use the specified logged recovery mechanism.

Primary note proof

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk. To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage. A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Study-note section ADB30

Active transactions can span the checkpoint, so their relevant earlier updates can still need undo.

Related reading (context, not primary note proof)

Study-note section ADB26

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 103 | lines 6-23

• Recovery from catastrophic
• If there is extensive damage to the wide portion of the database
• This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 104 | lines 19-25

• For Undo: Before a data item‘s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
• For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
• Standard Recovery Terminology

Fundamental of Database Systems module | PDF page 105 | lines 25-36

Check pointing
• Log file is used to recover failed DB but we may not know how far back in the log to search. Thus
• Checkpoint is a Point of synchronization between database and log file.
• Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery.
• When failure occurs, redo all transactions that committed since the checkpoint

and undo all transactions active at time of crash.
• In previous example ,Figure 2, on slide18 , with checkpoint at time tc, changes made by T2 and T3 have been written to secondary storage. Thus: only redo T4 and T5, undo transactions T1 and T6.
• At what interval to take a check pointing may be measured :

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science MCQ practice collection · Question 213

Question 102 | AM0214

Which recovery method is used to recover from media failures, such as disk crashes?

A. Rollback recovery

B. Checkpoint recovery

C. Restart recovery

D. Backup recovery

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer D: Backup recovery

For disk media destruction, restore a backup copy and reconstruct committed operations from the backup log (p.103). D supplies the missing database contents.

Why each choice is right or wrong

A - Incorrect: Undo alone does not recreate a destroyed database copy.

B - Incorrect: Checkpoint synchronization alone is not an independent media backup.

C - Incorrect: A restart without restoring lost storage is insufficient.

D - Correct: Backup recovery is the required catastrophic/media restoration approach.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 97 | lines 4-13

> i. Purpose of Database Recovery
> ◦ To bring the database into the last consistent state, which existed prior to the failure.
> ◦ To preserve transaction properties (Atomicity &and Durability).
> • The recovery manager of a DBMS is responsible to ensure
> • atomicity by undoing the action of transaction that do not commit
> • Durability by making sure that all actions of committed transaction survive system crash
> ii. Types of Failure
> ◦ The database may become unavailable for use due to :

Fundamental of Database Systems module | PDF page 103 | lines 6-23

> • Recovery from catastrophic
> • If there is extensive damage to the wide portion of the database
> • This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure

• Recovery from non-catastrophic failure
• When the database is not physically damaged but has be come in consistent
• The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
• For instance,
• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Question sources

Computer Science MCQ practice collection · Question 214

Question 103 | AM0215

What is a distributed database system?

A. A database system that is spread across multiple locations

B. A database system that uses multiple database management systems

C. A database system that supports multiple database schemas

D. A database system that is used for storing distributed data

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer A: A database system that is spread across multiple locations

A DDB is multiple logically related databases distributed over a computer network, managed transparently (p.114). A captures the physical distribution; logical relatedness and coordinated management are implicit qualifications.

Why each choice is right or wrong

A - Correct: Multiple locations captures the defining distribution.

B - Incorrect: Different DBMS products are not required; homogeneous DDBs exist.

C - Incorrect: Multiple schemas are not the physical distribution definition.

D - Incorrect: Storing distributed data is circular and omits the network/location criterion.

Primary note proof

Study-note section ADB33

A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB33

Distribution adds communication, coordination and security costs; a site outage can still make needed data unavailable if no usable replica exists.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 3-13

6.1 Distributed Database Concepts
• A transaction can be executed by multiple networked computers in a unified manner.
• A distributed database (DDB) processes Unit of execution (a transaction) in a distributed manner.
• A distributed database (DDB) can be defined as :
- A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
- The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Variant 1 | Alternative version 1

What is a distributed database system?

A. A system that uses multiple database management systems

B. A system that supports multiple database schemas

C. A system that spans multiple locations or sites

D. A system that replicates data across multiple databases

Answer C: A system that spans multiple locations or sites

Apply the same study-note rule to this alternative source set; the choice contrasts below show the derivation.

Why each choice is right or wrong

A - Incorrect: Multiple DBMS products are not required for distribution.

B - Incorrect: Multiple schemas are not the spatial/network definition.

C - Correct: Spanning sites matches the distributed placement definition on p.114.

D - Incorrect: Replication is optional and narrower than a distributed database.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB33

> Distribution adds communication, coordination and security costs; a site outage can still make needed data unavailable if no usable replica exists.

Question sources

Computer Science MCQ practice collection · Question 215

Computer Science MCQ practice collection · Question 225

Question 104 | AM0216

What is the key advantage of a distributed database system?

A. Improved data security

B. Reduced data redundancy

C. Enhanced data availability

D. Simplified data querying

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer C: Enhanced data availability

Multiple nodes allow others to do the job if one fails (p.116), improving availability. The module lists integrity/security difficulty and complexity among disadvantages (p.117).

Why each choice is right or wrong

A - Incorrect: Improved security is not guaranteed; distributed access complicates control.

B - Incorrect: Replication stores multiple copies rather than necessarily reducing redundancy.

C - Correct: Availability is explicitly improved through multiple nodes.

D - Incorrect: Querying across sites adds network and optimization complexity rather than inherently simplifying it.

Primary note proof

Study-note section ADB33

A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB37

Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB33

Distribution adds communication, coordination and security costs; a site outage can still make needed data unavailable if no usable replica exists.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 116 | lines 2-28

• Advantages of DDB :
i. Distribution and Network transparency:
• Users do not have to worry about operational details of the network.
» There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
» Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.
ii. Replication transparency:
• It allows to store copies of a data at multiple sites as shown in the above diagram.
• This is done to minimize access time to the required data.
iii. Fragmentation transparency:
• Allows to fragment a relation horizontally (create a subset of rows of a relation) or vertically (create a subset of columns of a relation).
iv. Increased reliability and availability:
• Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
• A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
v. Improved performance:
• A distributed DBMS fragments the database to keep data closer to where it is needed most.
• This reduces data management (access and modification) time significantly.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science MCQ practice collection · Question 216

Question 105 | AM0217

What is a homogenous distributed database system?

A. A system that uses the same database management system across all sites

B. A system that stores the same type of data across all sites

C. A system that uses a single database schema across all sites

D. A system that shares the same network infrastructure across all sites

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer A: A system that uses the same database management system across all sites

A homogeneous distributed database uses the same or compatible DBMS technology at its sites. Sharing data types, a schema or network infrastructure does not by itself establish that DBMS property. In particular, identical DBMS technology does not force all local schemas to be identical, and compatible structures alone do not prove identical DBMSs. The definition concerns the database-management technology, so A identifies the relevant criterion without adding unnecessary sameness requirements.

Why each choice is right or wrong

A - Correct: Correct: the same DBMS technology across sites meets the homogeneous-system criterion.

B - Incorrect: Incorrect: matching data types alone does not establish that the sites use the same DBMS technology.

C - Incorrect: Incorrect: schema agreement alone does not establish DBMS homogeneity, and same DBMS technology need not imply identical local schemas.

D - Incorrect: Incorrect: shared networking does not establish the database-management technology used at the sites.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Related reading (context, not primary note proof)

Study-note section ADB34

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 118 | lines 27-32

> 6.3 Types of Distributed Database Systems
> • Homogeneous
> - All sites of the database system have identical setup, i.e., same database system software.
> - The system may have little or no local autonomy
> - The underlying operating systems can be a mixture of Linux, Window, Unix, etc.

Variant 1 | Alternative version 1

What is a homogenous distributed database system?

A. A system that uses the same database management system across all sites

B. A system that stores the same type of data across all sites

C. A system that uses a single database schema across all sites

D. A system that uses the same network infrastructure across all sites

Answer A: A system that uses the same database management system across all sites

Apply the same study-note rule to this alternative source set; the choice contrasts below show the derivation.

Why each choice is right or wrong

A - Correct: Identical DBMS software matches p.118.

B - Incorrect: Identical data types alone do not establish software homogeneity.

C - Incorrect: A schema is not the software identity criterion.

D - Incorrect: Network infrastructure is not the DBMS software criterion.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB34

> A tightly integrated distributed system can present one global schema and shared control.

Study-note section ADB34

> State how much schema, transaction and administrative autonomy local sites retain.

Question sources

Computer Science MCQ practice collection · Question 217

Computer Science MCQ practice collection · Question 226

Question 106 | AM0218

What is a heterogeneous distributed database system?

A. A system that uses different database management systems across all sites

B. A system that stores different types of data across all sites

C. A system that uses different database schemas across all sites

D. A system that uses different network infrastructures across all sites

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer A: A system that uses different database management systems across all sites

Heterogeneous means at least one database is from a different vendor (p.119), supporting different DBMS software. It does not require every pair of sites to differ; A’s all sites must be read as across the participating sites, not pairwise uniqueness.

Why each choice is right or wrong

A - Correct: Different database-system vendors/software captures the defining distinction with the stated qualification.

B - Incorrect: Different content/types alone do not establish vendor heterogeneity.

C - Incorrect: Federated systems may have one conceptual schema despite heterogeneity.

D - Incorrect: Network differences are not the database-vendor definition.

Primary note proof

Study-note section ADB33

A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB33

Heterogeneous design must reconcile naming, types, units, identifiers, query capabilities and transaction semantics.

Study-note section ADB33

ODBC and JDBC provide connectivity interfaces; they do not automatically solve semantic differences or create one distributed transaction.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 119 | lines 11-20

• Heterogeneous
- At least one of the database must be from different vendor : two variants
- Federated: Each site may run different database system but the data access is managed through a single conceptual schema.
• This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.
- Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.
Site 2
Site 3

Question sources

Computer Science MCQ practice collection · Question 218

Computer Science MCQ practice collection · Question 227

Question 107 | AM0219

What is data fragmentation in a distributed database system?

A. Storing data in multiple copies at different sites

B. Dividing data into smaller subsets and storing them at different sites

C. Merging data from different sites into a single copy

D. Replicating data across all sites in the system

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Dividing data into smaller subsets and storing them at different sites

Fragmentation splits a relation into logically related correct parts (p.117); horizontal fragments select rows and vertical fragments select columns (p.118). B describes storing these subsets across sites.

Why each choice is right or wrong

A - Incorrect: Multiple identical copies describe replication.

B - Correct: Smaller subsets describe fragmentation.

C - Incorrect: Merging is reconstruction/integration, the opposite direction from splitting.

D - Incorrect: Copying data to all sites is replication rather than fragmentation.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB24

> Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Fundamental of Database Systems module | PDF page 118 | lines 27-32

6.3 Types of Distributed Database Systems
• Homogeneous
- All sites of the database system have identical setup, i.e., same database system software.
- The system may have little or no local autonomy
- The underlying operating systems can be a mixture of Linux, Window, Unix, etc.

Variant 1 | Alternative version 1

What is data fragmentation in a distributed database system?

A. Dividing data into smaller subsets and storing them at different sites

B. Replicating data across all sites for fault tolerance

C. Merging data from different sites into a single database

D. Encrypting data for secure transmission between sites

Answer A: Dividing data into smaller subsets and storing them at different sites

Apply the same study-note rule to this alternative source set; the choice contrasts below show the derivation.

Why each choice is right or wrong

A - Correct: Splitting into subsets matches fragmentation on pp.117-118.

B - Incorrect: Copies for availability describe replication.

C - Incorrect: Merging combines rather than splits the relations.

D - Incorrect: Encryption concerns protection rather than fragmentation.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB24

> Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions.

Question sources

Computer Science MCQ practice collection · Question 219

Computer Science MCQ practice collection · Question 228

Question 108 | AM0220

What is data replication in a distributed database system?

A. Storing data in multiple copies at different sites

B. Dividing data into smaller subsets and storing them at different sites

C. Merging data from different sites into a single copy

D. Deleting data that is no longer needed from the database

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer A: Storing data in multiple copies at different sites

Replication maintains several identical copies of a relation at different sites (p.117). This differs from dividing it into fragments.

Why each choice is right or wrong

A - Correct: Multiple copies at different sites exactly matches.

B - Incorrect: Subsets describe fragmentation.

C - Incorrect: Merging into one copy is not maintaining multiple copies.

D - Incorrect: Deletion removes data rather than replicating it.

Primary note proof

Study-note section ADB36

Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB37

Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB24

Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Variant 1 | Alternative version 1

What is data replication in a distributed database system?

A. Storing multiple copies of data at different sites

B. Dividing data into smaller subsets for efficient storage

C. Merging data from different sites into a single database

D. Encrypting data to ensure security during transmission

Answer A: Storing multiple copies of data at different sites

Apply the same study-note rule to this alternative source set; the choice contrasts below show the derivation.

Why each choice is right or wrong

A - Correct: Identical copies at different sites match p.117.

B - Incorrect: Subsets describe fragmentation, not replication.

C - Incorrect: Merging to one database does not retain distributed copies.

D - Incorrect: Encryption does not create replicated copies.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB24

> Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions.

Question sources

Computer Science MCQ practice collection · Question 220

Computer Science MCQ practice collection · Question 229

Question 109 | AM0221

Which transparency hides the physical site storing a named database object, so users can refer to that object without specifying its site?

A. Local-only data access

B. Fragmentation transparency

C. Location transparency

D. Replication transparency

Hint

Identify exactly which detail is hidden from the user.

Answer C: Location transparency

Location transparency hides the physical site holding a named logical object. Fragmentation transparency instead hides the existence and reconstruction of fragments, while replication transparency hides which copy is used and how copies are maintained. Those properties can coexist in a distributed database, so a vague single-logical-database experience would not identify only one. The revised stem isolates hidden site placement. Local-only access restricts what can be accessed rather than making remote placement transparent.

Why each choice is right or wrong

A - Incorrect: Incorrect: restricting access to local data does not hide the placement of a named distributed object.

B - Incorrect: Incorrect: fragmentation transparency hides splitting and reconstruction, not specifically the object's physical site.

C - Correct: Correct: location transparency hides which physical site stores the named object.

D - Incorrect: Incorrect: replication transparency hides copy selection and maintenance, not specifically site placement.

Primary note proof

Study-note section ADB35

> Location transparency hides which site holds a named object. Fragmentation transparency hides the existence and reconstruction of fragments: users query the whole logical relation. Replication transparency hides which copy is used and how copies are maintained. The broad single-logical-database experience may involve all three.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 3-13

> 6.1 Distributed Database Concepts
> • A transaction can be executed by multiple networked computers in a unified manner.
> • A distributed database (DDB) processes Unit of execution (a transaction) in a distributed manner.
> • A distributed database (DDB) can be defined as :
> - A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
> - The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Fundamental of Database Systems module | PDF page 116 | lines 2-28

> • Advantages of DDB :
> i. Distribution and Network transparency:
> • Users do not have to worry about operational details of the

network.
» There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
» Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.
ii. Replication transparency:
• It allows to store copies of a data at multiple sites as shown in the above diagram.
• This is done to minimize access time to the required data.
iii. Fragmentation transparency:
• Allows to fragment a relation horizontally (create a subset of rows of a relation) or vertically (create a subset of columns of a relation).
iv. Increased reliability and availability:
• Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
• A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
v. Improved performance:
• A distributed DBMS fragments the database to keep data closer to where it is needed most.
• This reduces data management (access and modification) time significantly.

Question sources

Computer Science MCQ practice collection · Question 221

Question 110 | AM0222

A user explicitly names a fragment and can query it without specifying or knowing the physical site that stores it. Which transparency is being provided? The existence of the fragment is visible.

A. Local-only data access

B. Fragmentation transparency

C. Location transparency

D. Replication transparency

Hint

Identify exactly which detail is hidden from the user.

Answer C: Location transparency

Location transparency hides the physical site storing a named object. The user in this scenario still names the fragment, so fragmentation remains visible; fragmentation transparency would hide the split and reconstruction. Replication transparency hides which copy is selected or maintained. Restricting users to local data is an access limitation rather than concealment of a remote fragment's placement. C therefore identifies the particular hidden detail without treating all transparency properties as interchangeable.

Why each choice is right or wrong

A - Incorrect: Incorrect: local-only access restricts accessible data rather than hiding a named remote fragment's physical site.

B - Incorrect: Incorrect: the user explicitly names the fragment, so its existence is not hidden.

C - Correct: Correct: the fragment's physical site is hidden while its logical name remains usable.

D - Incorrect: Incorrect: replica selection and maintenance are different hidden details from the specified site placement.

Primary note proof

Study-note section ADB35

> Location transparency hides which site holds a named object. Fragmentation transparency hides the existence and reconstruction of fragments: users query the whole logical relation. Replication transparency hides which copy is used and how copies are maintained. The broad single-logical-database experience may involve all three.

Study-note section ADB35

> If users explicitly name a fragment but need not specify its site, fragmentation is visible while location can remain transparent. Distinguish whether fragment names, storage sites or both are hidden from the user.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 3-13

> 6.1 Distributed Database Concepts
> • A transaction can be executed by multiple networked computers in a unified manner.
> • A distributed database (DDB) processes Unit of execution (a transaction) in a distributed manner.
> • A distributed database (DDB) can be defined as :
> - A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
> - The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Fundamental of Database Systems module | PDF page 116 | lines 2-28

• Advantages of DDB :
i. Distribution and Network transparency:
• Users do not have to worry about operational details of the network.
» There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
» Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.
ii. Replication transparency:
• It allows to store copies of a data at multiple sites as shown in the above diagram.
• This is done to minimize access time to the required data.
iii. Fragmentation transparency:
• Allows to fragment a relation horizontally (create a subset of rows of a relation) or vertically (create a subset of columns of a relation).
iv. Increased reliability and availability:
• Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
• A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
v. Improved performance:
• A distributed DBMS fragments the database to keep data closer to where it is needed most.
• This reduces data management (access and modification) time significantly.

Fundamental of Database Systems module | PDF page 118 | lines 27-32

6.3 Types of Distributed Database Systems
• Homogeneous
- All sites of the database system have identical setup, i.e., same database system software.
- The system may have little or no local autonomy
- The underlying operating systems can be a mixture of Linux, Window, Unix, etc.

Question sources

Computer Science MCQ practice collection · Question 222

Question 111 | AM0223

Which data transparency level ensures that users are not aware of the presence of multiple copies of data in different sites?

A. Local data transparency

B. Fragmentation transparency

C. Location transparency

D. Replication transparency

Hint

Identify exactly which detail is hidden from the user.

Answer D: Replication transparency

The module combines transparent distribution (p.114) with replication as storing copies at multiple sites (pp.116-117). Hiding this copying is replication transparency.

Why each choice is right or wrong

A - Incorrect: Local transparency does not identify multiple copies.

B - Incorrect: Fragmentation concerns subsets rather than duplicates.

C - Incorrect: Location hides placement but does not specifically name the copying aspect.

D - Correct: Replication transparency is the copy-specific aspect.

Primary note proof

Study-note section ADB35

> Location transparency hides which site holds a named object. Fragmentation transparency hides the existence and reconstruction of fragments: users query the whole logical relation. Replication transparency hides which copy is used and how copies are maintained. The broad single-logical-database experience may involve all three.

Study-note section ADB35

> If users explicitly name a fragment but need not specify its site, fragmentation is visible while location can remain transparent.

Study-note section ADB35

> Column subsets define vertical fragmentation, not a type of transparency.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 114 | lines 3-13

> 6.1 Distributed Database Concepts
> • A transaction can be executed by multiple networked computers in a unified manner.
> • A distributed database (DDB) processes Unit of execution (a transaction) in a

distributed manner.
• A distributed database (DDB) can be defined as :
- A collection of multiple logically related database distributed over a computer network, and a distributed database management system as a software system that manages a distributed database while making the distribution transparent to the user.
- The physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

Fundamental of Database Systems module | PDF page 116 | lines 2-28

• Advantages of DDB :
i. Distribution and Network transparency:
• Users do not have to worry about operational details of the network.
» There is Location transparency, which refers to freedom of issuing command from any location without affecting its working.
» Then there is Naming transparency, which allows access to any names object (files, relations, etc.) from any location.
ii. Replication transparency:
• It allows to store copies of a data at multiple sites as shown in the above diagram.
• This is done to minimize access time to the required data.
iii. Fragmentation transparency:
• Allows to fragment a relation horizontally (create a subset of rows of a relation) or vertically (create a subset of columns of a relation).
iv. Increased reliability and availability:
• Reliability refers to system live time, that is, system is running efficiently most of the time. Availability is the probability that the system is continuously available (usable or accessible) during a time interval.
• A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.
v. Improved performance:
• A distributed DBMS fragments the database to keep data closer to where it is needed most.
• This reduces data management (access and modification) time significantly.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the

availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science MCQ practice collection · Question 223

Question 112 | AM0224

What is the role of a distributed query processor in a distributed database system?

A. It ensures the consistency of data across all sites.

B. It manages the distribution and retrieval of data across multiple sites.

C. It controls the access permissions of users in the system.

D. It handles the physical storage of data at each site.

Hint

Compare local processing with the bytes and messages moved between sites.

Answer B: It manages the distribution and retrieval of data across multiple sites.

Distributed query processing compares moving source relations and results among sites, selecting the lower-transfer strategy (pp.120-121). B best captures distributed retrieval, understood as planning transfers for a query rather than deciding the physical database design.

Why each choice is right or wrong

A - Incorrect: Global consistency is assigned to concurrency control/recovery rather than query transfer planning.

B - Correct: Managing query data transfer and retrieval matches the worked example.

C - Incorrect: Access permissions are authorization, not the worked query role.

D - Incorrect: Physical local storage management is not choosing distributed query strategies.

Primary note proof

Study-note section ADB38

Distributed processing decomposes a logical query, localizes referenced relations to fragments/replicas, chooses global placement/order and optimizes local subplans. Include bytes

moved, message latency, bandwidth and local work. Apply filters and projections before shipping when equivalent. A semijoin retains only rows of one input that have a match in the other, returning that input's attributes.

Study-note section ADB24

Authentication establishes identity; authorization/access control determines permitted actions.

Study-note section ADB34

A tightly integrated distributed system can present one global schema and shared control.

Related reading (context, not primary note proof)

Study-note section ADB39

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 120 | lines 16-33

6.4 Query Processing in Distributed Databases
• Issues
- Cost of transferring data (files and results) over the network.
• This cost is usually high, so some optimization is necessary.
• Example: suppose there are three sites. Where the relation Employee at site 1, Department at Site 2 and no relation at site 3
- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 106
bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.
- And a query is initiated from S3 to retrieve employees [First Name (15 byte long), Last name (15 byte long) and Department name (10 byte long) total of 40 bytes]
• Q: For each employee, retrieve employee Fname, Lname, and department name
• Q: δFname,Lname,Dname (Employee Dno = Dnumber Department)
• Assumption
Site 3

Fundamental of Database Systems module | PDF page 121 | lines 8-24

• Strategies : Minimizing data transfer.
- Transfer Employee and Department to site 3.
• Total transfer bytes = 1,000,000 + 3500 = 1,003,500 bytes.
- Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
• Transferring employees data from site 1 to site 2: 1,000,000 bytes
• Query result size = 40 * 10,000 = 400,000 bytes.
• Total transfer size = 1,000,000 + 400,000 = 1,400,000 bytes.
- Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
• Data Transfer from site 2 to site 1: 3500 bytes

• Query result size = 40 * 10,000 = 400,000 bytes
• Total bytes transferred = 3500+ 400,000 = 403,500 bytes.
• Preferred approach: strategy 3.
Example 2 : Consider the query
- Q‟: For each department, retrieve the department name ,Fname and LName of the department manager
• Relational Algebra expression:

Question sources

Computer Science MCQ practice collection · Question 224

Question 113 | AM0230

Which condition specifically makes a distributed database environment homogeneous, rather than merely being useful for distributed operation?

A. Consistency of database schemas across all sites

B. Compatibility of network infrastructures across all sites

C. Standardization of database management systems across all sites

D. Replication of data at each site for fault tolerance

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer C: Standardization of database management systems across all sites

Homogeneity is defined by using the same or compatible DBMS technology across sites. Schema compatibility, functioning network connections and replication can be useful design concerns, but none alone establishes that common DBMS technology. A homogeneous system may have different local schemas, and replication is not required merely because data is distributed. Narrowing the question to the defining condition preserves C while acknowledging that the other design concerns can matter in practice.

Why each choice is right or wrong

A - Incorrect: Incorrect: schema compatibility can be useful but does not establish common DBMS technology.

B - Incorrect: Incorrect: compatible networking supports communication but does not establish DBMS homogeneity.

C - Correct: Correct: DBMS standardization supplies the defining common technology across sites.

D - Incorrect: Incorrect: replication can improve availability but is optional and does not define homogeneity.

Primary note proof

Study-note section ADB33

A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB37

Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 118 | lines 27-32

6.3 Types of Distributed Database Systems
• Homogeneous
- All sites of the database system have identical setup, i.e., same database system software.
- The system may have little or no local autonomy
- The underlying operating systems can be a mixture of Linux, Window, Unix, etc.

Question sources

Computer Science MCQ practice collection · Question 230

Question 114 | AM0231

A heterogeneous distributed database retains different local DBMSs and schemas and must integrate them without requiring full replication. Which integration requirements apply? Select all that apply. D means coherent schema semantics through mappings, not identical schemas.

A. Replication of data at each site for fault tolerance

B. Standardization of database management systems across all sites

C. Compatibility of network infrastructures across all sites

D. Consistency of database schemas across all sites

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer C, D: C. Compatibility of network infrastructures across all sites; D. Consistency of database schemas across all sites

Integration requires interoperable communication so requests and results can move between sites. It also requires coherent mappings for names, types, identifiers and other schema meanings. These support C and D. Replacing every DBMS with one product would remove the retained heterogeneity, and full replication is an optional placement decision rather than a requirement for integrating different local systems.

Why each choice is right or wrong

A - Incorrect: Copying all data everywhere is not required for the stated integration.

B - Incorrect: The scenario retains differing DBMSs rather than replacing all with one standard.

C - Correct: Interoperable communication is needed to exchange requests and results.

D - Correct: Mappings must reconcile schema meanings without requiring identical local schemas.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB33

> Heterogeneous design must reconcile naming, types, units, identifiers, query capabilities and transaction semantics.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Related reading (context, not primary note proof)

Study-note section ADB40

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 119 | lines 11-20

> • Heterogeneous
> - At least one of the database must be from different vendor : two variants
> - Federated: Each site may run different database system but the data access is managed through a single conceptual schema.
> • This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.
> - Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.
> Site 2
> Site 3

Fundamental of Database Systems module | PDF page 117 | lines 15-31

> 6.2 Data Replication and Fragmentation: Distributed data storage
> • There are two approaches to store the relation in the distributed database : Replication

and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science MCQ practice collection · Question 231

Question 115 | AM0232

Which approach is commonly used for query processing in a distributed database system?

A. Centralized query processing

B. Distributed query processing

C. Parallel query processing

D. Sequential query processing

Hint

Compare local processing with the bytes and messages moved between sites.

Answer B: Distributed query processing

The module's distributed-query section compares execution and data-transfer locations among multiple sites (pp.120-121). B names the overall approach. A chosen plan may execute a particular join at one site; that does not make the whole system centralized.

Why each choice is right or wrong

A - Incorrect: A local execution site can be part of a distributed plan, but centralized processing does not name the chapter's overall approach.

B - Correct: Distributed query processing matches the section and cross-site strategies.

C - Incorrect: Parallelism may be used, but simultaneous execution is not required by these examples.

D - Incorrect: A plan may contain sequential operations; this does not identify cross-site processing.

Primary note proof

Study-note section ADB38

Distributed processing decomposes a logical query, localizes referenced relations to fragments/replicas, chooses global placement/order and optimizes local subplans. Include bytes moved, message latency, bandwidth and local work. Apply filters and projections before shipping when equivalent. A semijoin retains only rows of one input that have a match in the other, returning that input's attributes.

Study-note section ADB38

If nearly all S matches, key shipping may not help.

Study-note section ADB22

Plan caching reuses an execution strategy, not the rows returned.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 120 | lines 16-33

6.4 Query Processing in Distributed Databases
• Issues
- Cost of transferring data (files and results) over the network.
• This cost is usually high, so some optimization is necessary.
• Example: suppose there are three sites. Where the relation Employee at
site 1, Department at Site 2 and no relation at site 3
- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size
= 106
bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size =
3,500 bytes.
- And a query is initiated from S3 to retrieve employees [First Name (15 byte long),
Last name (15 byte long) and Department name (10 byte long) total of 40 bytes]
• Q: For each employee, retrieve employee Fname, Lname, and
department name
• Q: δFname,Lname,Dname (Employee Dno = Dnumber Department)
• Assumption
Site 3

Fundamental of Database Systems module | PDF page 121 | lines 8-24

• Strategies : Minimizing data transfer.
- Transfer Employee and Department to site 3.
• Total transfer bytes = 1,000,000 + 3500 = 1,003,500 bytes.
- Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
• Transferring employees data from site 1 to site 2: 1,000,000 bytes

• Query result size = 40 * 10,000 = 400,000 bytes.
• Total transfer size = 1,000,000 + 400,000 = 1,400,000 bytes.
- Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
• Data Transfer from site 2 to site 1: 3500 bytes
• Query result size = 40 * 10,000 = 400,000 bytes
• Total bytes transferred = 3500+ 400,000 = 403,500 bytes.
• Preferred approach: strategy 3.
Example 2 : Consider the query
- Q‟: For each department, retrieve the department name ,Fname and LName of the department manager
• Relational Algebra expression:

Question sources

Computer Science MCQ practice collection · Question 232

Question 116 | AM0233

What is the role of a distributed transaction coordinator in a distributed database system?

A. It ensures the consistency of data across all sites.

B. It manages the distribution and retrieval of data across multiple sites.

C. It controls the access permissions of users in the system.

D. It handles the physical storage of data at each site.

Hint

Distinguish coordinated transaction outcome from concurrent access and local storage.

Answer A: It ensures the consistency of data across all sites.

Global consistency is required of concurrency/recovery (p.122). Under the primary-site technique a coordinator manages transaction concurrency and commit (p.123). A is supported in that transaction-coordination sense, rather than as an unconditional guarantee against every error.

Why each choice is right or wrong

A - Correct: Coordination of concurrency and commit maintains transaction consistency across sites.

B - Incorrect: Query transfer/retrieval is a separate distributed-query task.

C - Incorrect: User permissions are authorization, not commit coordination.

D - Incorrect: Local physical storage is not the primary coordinator’s stated function.

Primary note proof

Study-note section ADB39

Two-phase commit (2PC) first asks participants to prepare and durably promise they can commit. If all vote yes, the coordinator records commit durably and tells participants; otherwise it decides abort. A prepared participant must retain enough state to obey the final decision. Coordinator/communication failure can leave it in doubt and block progress until the decision is learned. Timeout after preparation is not permission to invent a commit or abort.

Study-note section ADB39

The transaction coordinator coordinates decisions rather than physically storing every site's rows.

Study-note section ADB38

Distributed processing decomposes a logical query, localizes referenced relations to fragments/replicas, chooses global placement/order and optimizes local subplans.

Related reading (context, not primary note proof)

Study-note section ADB06

Study-note section ADB24

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 122 | lines 22-29

6.5 Concurrency Control and Recovery
• Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases. Some of them are listed below.
• Dealing with multiple copies of data items:
• The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.
• Failure of individual sites:

Fundamental of Database Systems module | PDF page 123 | lines 15-29

• Distributed commit:
• A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit.
• Distributed deadlock:
• Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.
6.5.1 Distributed Concurrency control
i. Primary site technique: A single site is designated as a primary site which serves as a coordinator for transaction management.
• Transaction management:
- Concurrency control and commit are managed by this site.
- In two phase locking, this site manages locking and releasing data items. If all

transactions follow two-phase policy at all sites, then serializability is guaranteed.
- Advantages:

Question sources

Computer Science MCQ practice collection · Question 233

Question 117 | AM0234

Which listed approaches are replication policies or transfer methods in distributed databases, allowing dimensions to coexist? Select all that apply.

A. Eager replication

B. Lazy replication

C. Snapshot replication

D. Incremental replication

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer A, B, C, D: A. Eager replication; B. Lazy replication; C. Snapshot replication; D. Incremental replication

Eager and lazy replication differ in when propagation occurs relative to the acknowledgment or commit point. Snapshot and incremental replication differ in whether a copy or changes are transferred. These are legitimate replication dimensions and can coexist, so all four qualify. The question asks for recognized approaches rather than claiming that one policy is most prevalent or universally preferable.

Why each choice is right or wrong

A - Correct: Eager replication coordinates propagation before the specified acknowledgment or commit point.

B - Correct: Lazy replication propagates changes later and can expose stale reads or require conflict handling.

C - Correct: Snapshot replication distributes a copy at intervals rather than necessarily sending each individual update.

D - Correct: Incremental replication transfers changes and can coexist with a propagation-timing policy.

Primary note proof

Study-note section ADB37

Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once. Copies

improve read locality and can improve availability but require storage and update propagation. They can also spread mistakes, so replication does not replace independent backups.

Study-note section ADB37

Eager/synchronous replication coordinates propagation before the specified commit/acknowledgment point, increasing update latency.

Study-note section ADB37

Snapshot replication transfers a copy at intervals; incremental replication transfers changes.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science MCQ practice collection · Question 234

Question 118 | AM0235

When comparing distributed query strategies, which factors can affect estimated execution cost or replica placement decisions? Select all that apply.

A. Cost of query execution

B. Size of the database

C. Network bandwidth

D. Data replication

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer A, B, C, D: A. Cost of query execution; B. Size of the database; C. Network bandwidth; D. Data replication

Execution cost summarizes the resources consumed by a candidate plan. Relation sizes affect scans and intermediate work, network bandwidth affects data-transfer time, and replica placement changes which copies are available near the computation. All these inputs can alter a distributed strategy's cost. They are complementary considerations rather than competing names for one exclusively important factor.

Why each choice is right or wrong

A - Correct: Estimated execution cost is the explicit basis for comparing candidate strategies.

B - Correct: Database and relation sizes affect scan work, intermediate sizes and algorithm feasibility.

C - Correct: Bandwidth affects transfer time when data or results move between sites.

D - Correct: Replica locations affect which copy can serve a query and its communication cost.

Primary note proof

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic. Catalog statistics include row/page counts, distinct values and distributions. Selection with fraction s predicts s×N rows. For non-NULL equijoin keys, a common estimate is |R|×|S|/max(V(R,k),V(S,k)), assuming uniform values and containment of the smaller distinct-value domain within the larger. Adjust counts for NULLs; if either non-NULL input is empty, the result is zero. Skew, correlations or incomplete domain overlap can invalidate this estimate; known disjoint domains give zero matches.

Study-note section ADB38

> Distributed processing decomposes a logical query, localizes referenced relations to fragments/replicas, chooses global placement/order and optimizes local subplans.

Study-note section ADB38

> Include bytes moved, message latency, bandwidth and local work.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 41 | lines 25-29

> B. Cost Estimation Approach to Query Optimization
> - The main idea is to minimize he cost of processing a query. The cost function is comprised of:
> - I/O cost + CPU processing cost + communication cost + Storage cost
> - These components might have different weights in different processing environments

Fundamental of Database Systems module | PDF page 42 | lines 10-25

Cost Estimation for RA Operations
- Many different ways of implementing RA operations.
- Aim of QO is to choose most efficient one.
- Use formulae that estimate costs for a number of options, and select one with lowest cost.
- Consider only cost of disk access, which is usually dominant cost in QP.
- Many estimates are based on cardinality of the relation, so need to be able to estimate this.

Using Selectivity and Cost Estimates in Query Optimization
- Cost-based query optimization:
  - Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate. (Compare to heuristic query optimization)
- Issues
  - Cost function
  - Number of execution strategies to be considered
- Cost Components for Query Execution

Fundamental of Database Systems module | PDF page 120 | lines 16-33

6.4 Query Processing in Distributed Databases
• Issues
- Cost of transferring data (files and results) over the network.
• This cost is usually high, so some optimization is necessary.
• Example: suppose there are three sites. Where the relation Employee at site 1, Department at Site 2 and no relation at site 3
- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 106
bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.
- And a query is initiated from S3 to retrieve employees [First Name (15 byte long), Last name (15 byte long) and Department name (10 byte long) total of 40 bytes]
• Q: For each employee, retrieve employee Fname, Lname, and department name
• Q: δFname,Lname,Dname (Employee Dno = Dnumber Department)
• Assumption
Site 3

Fundamental of Database Systems module | PDF page 42 | lines 15-17

- Many estimates are based on cardinality of the relation, so need to be able to estimate this.

Using Selectivity and Cost Estimates in Query Optimization

Question sources

Computer Science MCQ practice collection · Question 235

Question 119 | AM0236

What is the purpose of query optimization in a database system?

A. To ensure data consistency

B. To minimize query execution time

C. To maximize data replication

D. To manage database security

Hint

Separate query representation, strategy selection and execution.

Answer B: To minimize query execution time

Query optimization chooses an efficient execution strategy and minimizes resource usage including response time (p.27). Page 30 explicitly seeks faster queries and the most efficient sequence of steps.

Why each choice is right or wrong

A - Incorrect: Consistency is a transaction correctness concern, not the stated optimization objective.

B - Correct: Minimizing execution/response time captures the efficiency objective.

C - Incorrect: Maximizing replication is not the execution-plan goal.

D - Incorrect: Authorization/security is not query optimization's stated purpose.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work.

Related reading (context, not primary note proof)

Study-note section ADB19

Study-note section ADB24

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 27 | lines 4-15

What is Query Processing?
Steps required to transform high level SQL query into a correct and —efficient‖ strategy for execution and retrieval.
What is Query Optimization?
✓ The activity of choosing a single —efficient‖ execution strategy (from hundreds) as determined by database catalog statistics.
✓ The aim is to answer which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?
✓ For each algebraic operator, what algorithm (of several available) do we use to compute that operator?
✓ How do operations pass data (main memory buffer, disk buffer,...)?
✓ Will this plan minimize resource usage? (CPU/Response Time/Disk)

Fundamental of Database Systems module | PDF page 30 | lines 17-30

1.8 Query Optimization
• Everyone wants the performance of their database to be optimal. In particular, there is often a requirement for a specific query or object that is query based, to run faster.
• Problem of query optimization is to find the sequence of steps that produces the answer to user request in the most efficient manner, given the database structure.
• The performance of a query is affected by the tables or views that underlies the query and by the complexity of the query.
• Given a request for data manipulation or retrieval, an optimizer will choose an optimal plan for evaluating the request from among the manifold alternative strategies. i.e. there are many ways (access paths) for accessing desired file/record.
• Hence ,DBMS is responsible to pick the best execution strategy based on various considerations(Least amount of I/O and CPU resources.)
• Example: Consider relations r(AB) and s(CD). We require r X s.
• Method 1 :

Question sources

Computer Science MCQ practice collection · Question 236

Question 120 | AM0237

What is a query plan in a database system?

A. A set of database schemas

B. A representation of the query result

C. A sequence of operations to execute a query

D. A collection of database indices

Hint

Separate query representation, strategy selection and execution.

Answer C: A sequence of operations to execute a query

An execution plan includes a relational-algebra query tree, access methods and operator implementations (p.44). Its operations run from leaves toward the root (p.29), so C captures the execution procedure, which can have a tree structure rather than a single linear list.

Why each choice is right or wrong

A - Incorrect: Schemas describe data structure rather than the executable operations.

B - Incorrect: The result is the output, whereas the plan specifies how to obtain it.

C - Correct: An operation sequence/tree with access methods is the stated plan concept.

D - Incorrect: Indices may be access methods within a plan but do not alone constitute it.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> DDL is not a mandatory stage before each SELECT, so authentication→DDL→DML permutations do not describe the processing sequence.

Study-note section ADB-S01

> A query tree corresponds to a relational-algebra expression: base relations appear at leaves and operators at internal nodes.

Related reading (context, not primary note proof)

Study-note section ADB23

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 29 | lines 9-20

> 1.6 Query Processing Steps
> • Processing can be divided into 4 :Decomposition, Optimization , Execution and Code generation
> 1.7 Query Decomposition
> • It is the process of transforming a high level query into a relational algebra query, and to check that the query is syntactically and semantically correct. It Consists of parsing and validation

Typical stages in query decomposition are:
i. Analysis: lexical and syntactical analysis of the query (correctness) based on attributes, data type.. ,. Query tree will be built for the query containing leaf node for base relations, one or many non-leaf nodes for relations produced by relational algebra operations and root node for the result of the query. Sequence of operation is from the leaves to the

Fundamental of Database Systems module | PDF page 44 | lines 6-13

3. Query Execution Plans
◦ An execution plan for a relational algebra query consists of :
◦ Relational algebra query tree
◦ Information about the access methods to be used for each relation
◦ The methods to be used in computing the relational operators stored in the tree.
1.9 Pipelining
● Pipelined evaluation: evaluate several operations simultaneously, passing the results of one operation on to the next.

Question sources

Computer Science MCQ practice collection · Question 237

Question 121 | AM0238

What is a cost model in query optimization?

A. A statistical model of query execution times

B. A model for estimating the cost of query plans

C. A model for analyzing database schemas

D. A statistical model for data distribution

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: A model for estimating the cost of query plans

The cost function combines I/O, CPU, communication and storage (p.41); optimization estimates and compares alternative strategy costs (p.42). A cost model therefore estimates costs of query plans.

Why each choice is right or wrong

A - Incorrect: Execution time can contribute, but this is narrower than the listed resource-cost model.

B - Correct: Estimating plan costs is the defining function.

C - Incorrect: Schema analysis is not evaluation of competing execution strategies.

D - Incorrect: Data distribution statistics are possible inputs, not the whole cost model.

Primary note proof

Study-note section ADB19

A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic. Catalog statistics include row/page counts, distinct values and distributions. Selection with fraction s predicts s×N rows. For non-NULL equijoin keys, a common estimate is |R|×|S|/max(V(R,k),V(S,k)), assuming uniform values and containment of the smaller distinct-value domain within the larger. Adjust counts for NULLs; if either non-NULL input is empty, the result is zero. Skew, correlations or incomplete domain overlap can invalidate this estimate; known disjoint domains give zero matches.

Study-note section ADB19

State the assumed cost objective before ranking strategies.

Study-note section ADB19

Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 41 | lines 25-29

B. Cost Estimation Approach to Query Optimization
- The main idea is to minimize he cost of processing a query. The cost function is comprised of:
- I/O cost + CPU processing cost + communication cost + Storage cost
- These components might have different weights in different processing environments

Fundamental of Database Systems module | PDF page 42 | lines 10-25

Cost Estimation for RA Operations
- Many different ways of implementing RA operations.
- Aim of QO is to choose most efficient one.
- Use formulae that estimate costs for a number of options, and select one with lowest cost.
- Consider only cost of disk access, which is usually dominant cost in QP.
- Many estimates are based on cardinality of the relation, so need to be able to estimate this.

Using Selectivity and Cost Estimates in Query Optimization
- Cost-based query optimization:
  - Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate. (Compare to heuristic query optimization)
- Issues
  - Cost function
  - Number of execution strategies to be considered
- Cost Components for Query Execution

Question sources

Computer Science MCQ practice collection · Question 238

Question 122 | AM0239

What is a join operation in query processing?

A. Combining tuples from multiple tables based on a common attribute

B. Sorting the query result in ascending order

C. Grouping the query result based on a specific attribute

D. Filtering the query result based on a condition

Hint

Separate the query's representation from the operations used to calculate its result.

Answer A: Combining tuples from multiple tables based on a common attribute

The query joins Staff and Branch by matching branchNo (p.27). The heuristic combines a Cartesian product and selection on the join condition into a join (p.41); the distributed example similarly matches Employee.Dno and Department.Dnumber (p.120). A expresses this equijoin case; joins in general need not always use equality or identical attribute names.

Why each choice is right or wrong

A - Correct: Combining tuples that satisfy the matching attribute condition describes these joins.

B - Incorrect: Sorting changes order rather than combining source tuples.

C - Incorrect: Grouping builds groups/aggregates, not cross-relation matching.

D - Incorrect: Filtering alone selects rows rather than combining related tuples from tables.

Primary note proof

Study-note section ADB-S01

> A relation is represented by a table of tuples. Selection filters its rows by a predicate; projection retains chosen columns. A Cartesian product constructs every pair of input tuples. A matching join combines tuples satisfying its predicate, and can be represented as a selection over a product. Sorting orders results, while grouping collects rows sharing grouping values; these are different operations.

Study-note section ADB18

> Heuristics reduce intermediate work: push selective row filters toward base tables; replace product followed by a join predicate with a join; project unneeded columns away while retaining join/filter columns; reorder inner joins when equivalence permits; reuse common expressions when worthwhile.

Study-note section ADB18

> Cost-based optimization estimates candidate plans instead of relying on rules alone.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 27 | lines 4-15

What is Query Processing?
Steps required to transform high level SQL query into a correct and —efficient‖ strategy for execution and retrieval.
What is Query Optimization?
✓ The activity of choosing a single —efficient‖ execution strategy (from hundreds) as determined by database catalog statistics.
✓ The aim is to answer which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?
✓ For each algebraic operator, what algorithm (of several available) do we use to compute that operator?
✓ How do operations pass data (main memory buffer, disk buffer,...)?
✓ Will this plan minimize resource usage? (CPU/Response Time/Disk)

Fundamental of Database Systems module | PDF page 41 | lines 25-29

B. Cost Estimation Approach to Query Optimization
- The main idea is to minimize he cost of processing a query. The cost function is comprised of:
- I/O cost + CPU processing cost + communication cost + Storage cost
- These components might have different weights in different processing environments

Fundamental of Database Systems module | PDF page 120 | lines 16-33

6.4 Query Processing in Distributed Databases
• Issues
- Cost of transferring data (files and results) over the network.
• This cost is usually high, so some optimization is necessary.
• Example: suppose there are three sites. Where the relation Employee at site 1, Department at Site 2 and no relation at site 3
- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 106
bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.
- And a query is initiated from S3 to retrieve employees [First Name (15 byte long), Last name (15 byte long) and Department name (10 byte long) total of 40 bytes]
• Q: For each employee, retrieve employee Fname, Lname, and department name
• Q: δFname,Lname,Dname (Employee Dno = Dnumber Department)
• Assumption
Site 3

Fundamental of Database Systems module | PDF page 41 | lines 5-7

- Combine Cartesian product with subsequent Selection whose predicate represents join condition into a Join operation.
- Use associativity of binary operations to rearrange leaf nodes so leaf nodes with most

Question sources

Computer Science MCQ practice collection · Question 239

Question 123 | AM0240

Which listed index or clustered-storage approaches can speed a suitable query workload, without a claim that any always wins? Select all that apply.

A. B-tree index

B. Hash index

C. Bitmap index

D. Clustered index

Hint

Match the predicate and workload to the access structure or reasons to choose a plan again.

Answer A, B, C, D: A. B-tree index; B. Hash index; C. Bitmap index; D. Clustered index

B-tree indexes support ordered and range access, hash indexes target equality, and bitmap indexes can support analytic selections. Clustered storage places related rows close together and can reduce access work for suitable queries. Each approach can therefore help an appropriate workload. Space, maintenance and selectivity still matter, so accepting all four does not claim that every query benefits or that one design is universally best.

Why each choice is right or wrong

A - Correct: B-tree indexes support selective ordered and range access for suitable queries.

B - Correct: Hash indexes can accelerate equality lookups on the indexed key.

C - Correct: Bitmap indexes can accelerate suitable analytic selection workloads.

D - Correct: Clustered storage keeps related rows close and can reduce reads for appropriate access patterns.

Primary note proof

Study-note section ADB23

B/B+ tree indexes support ordered and range predicates; hash indexes target equality; bitmap indexes can help some analytic selections. Clustered storage keeps related rows close. An index may reduce reads but adds space and update work; scanning can be faster when most rows qualify. Commonly useful is not a unique property of one index type.

Study-note section ADB23

Avoid universal rules that joins always beat subqueries, functions can never use indexes or more indexes always improve performance.

Study-note section ADB23

Updating statistics does not itself execute the query.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 21 | lines 5-11

5.5 Index
Indexes are additional auxiliary access structures which are used to speed up the retrieval of records in response to certain search conditions. Index structures are additional files, which provide alternative ways to access the records without affecting the physical placement of records on disk.
Types of Indexes
Dense Index

Fundamental of Database Systems module | PDF page 22 | lines 11-20

Clustering Indexes
Here files are assumed as ordered with a Non-key field. If file records are physically ordered on a non-key field which doesn‘t have a distinct value for each record that field is called clustering field and the data file is called a clustered file.
We can create a different type of index, called a clustering index, to speedup retrieval of all records that have the same value for the clustering field. If the ordering field is not a key field i.e; if numerous records in the file can have the same value for the ordering field clustering index can be used.
A clustering index is also an ordered file with two fields. The first field is of the same type as the clustering field of the data file, and the second field is a disk block/cluster pointer.

Question sources

Computer Science MCQ practice collection · Question 240

Question 124 | AM0241

What is query re-optimization in a database system?

A. Re-executing the query with different parameters

B. Re-evaluating the query cost based on updated statistics

C. Re-writing the query in a different query language

D. Re-organizing the database schema to improve performance

Hint

Match the predicate and workload to the access structure or reasons to choose a plan again.

Answer B: Reconsider the plan using updated cost information or statistics.

ADB23 defines re-optimization as choosing a plan again when statistics, parameters or observed cardinalities suggest that the existing plan is unsuitable. B is the listed description of this cost reconsideration. Merely executing again with other values does not necessarily re-optimize, and changing the query language or redesigning the schema is not the definition.

Why each choice is right or wrong

A - Incorrect: Re-execution with new parameters can reuse the same plan and is not inherently re-optimization.

B - Correct: Updated statistics can change cost estimates and lead to choosing a new execution plan.

C - Incorrect: Changing query language is not required to reconsider an execution strategy.

D - Incorrect: Schema reorganization is a design change rather than the definition of query re-optimization.

Primary note proof

Study-note section ADB23

> B/B+ tree indexes support ordered and range predicates; hash indexes target equality; bitmap indexes can help some analytic selections. Clustered storage keeps related rows close. An index may reduce reads but adds space and update work; scanning can be faster when most rows qualify. Commonly useful is not a unique property of one index type.

Study-note section ADB23

> Re-optimization chooses a plan again when statistics, parameters or observed cardinalities suggest the old plan is unsuitable.

Study-note section ADB23

> Updating statistics does not itself execute the query.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 27 | lines 4-15

> What is Query Processing?
> Steps required to transform high level SQL query into a correct and —efficient‖ strategy for execution and retrieval.
> What is Query Optimization?
> ✓ The activity of choosing a single —efficient‖ execution strategy (from hundreds) as determined by database catalog statistics.
> ✓ The aim is to answer which relational algebra expression, equivalent to the given query, will lead to the most efficient solution plan?
> ✓ For each algebraic operator, what algorithm (of several available) do we use to compute that operator?

✓ How do operations pass data (main memory buffer, disk buffer,...)?
✓ Will this plan minimize resource usage? (CPU/Response Time/Disk)

Fundamental of Database Systems module | PDF page 42 | lines 10-25

Cost Estimation for RA Operations
- Many different ways of implementing RA operations.
- Aim of QO is to choose most efficient one.
- Use formulae that estimate costs for a number of options, and select one with lowest cost.
- Consider only cost of disk access, which is usually dominant cost in QP.
- Many estimates are based on cardinality of the relation, so need to be able to estimate this.

Using Selectivity and Cost Estimates in Query Optimization
- Cost-based query optimization:
  - Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate. (Compare to heuristic query optimization)
- Issues
  - Cost function
  - Number of execution strategies to be considered
- Cost Components for Query Execution

Question sources

Computer Science MCQ practice collection · Question 241

Question 125 | AM0242

What is query pipelining in query processing?

A. Dividing a query into subqueries and executing them in parallel

B. Dividing a query into stages and executing them sequentially

C. Dividing a query into multiple queries and executing them in parallel

D. Dividing a query into multiple stages and executing them concurrently

Hint

Distinguish overlapping dependent operators, independent workers and reused cached objects.

Answer D: Dividing a query into multiple stages and executing them concurrently

Pipelined evaluation evaluates several operations simultaneously and passes results directly from one to the next (p.44). D means overlapping producer/consumer stages, not necessarily executing independent whole queries.

Why each choice is right or wrong

A - Incorrect: Parallel independent subqueries are not the defining inter-operator tuple stream.

B - Incorrect: Pure sequential completion of each whole stage loses the stated simultaneous overlap.

C - Incorrect: Multiple independent queries are not the defining pipeline within an expression tree.

D - Correct: Concurrent overlapping stages passing tuples match the explicit definition.

Primary note proof

Study-note section ADB22

> Materialization stores an intermediate result before the next operator consumes it. Pipelining passes tuples as they become available, overlapping producer and consumer work and reducing temporary I/O. A sort is normally blocking; a hash join must build its hash table before probing can output, though later probe/output can stream. Pipelining does not require separate CPUs.

Study-note section ADB22

> Parallel execution uses multiple workers, for example partitioned scans/joins, independent subplans or operator pipelines.

Study-note section ADB22

> Specify the kind of query caching being discussed.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 44 | lines 6-13

> 3. Query Execution Plans
> ◦ An execution plan for a relational algebra query consists of :
> ◦ Relational algebra query tree
> ◦ Information about the access methods to be used for each relation
> ◦ The methods to be used in computing the relational operators stored in the tree.
> 1.9 Pipelining
> • Pipelined evaluation: evaluate several operations simultaneously, passing the results of one operation on to the next.

Fundamental of Database Systems module | PDF page 44 | lines 12-20

> • Pipelined evaluation: evaluate several operations simultaneously, passing the results of one operation on to the next.
> • E.g., in expression tree, don‘t store result of
> ◦ Instead, pass tuples directly to the join.. Similarly, don‘t store result of join, pass tuples directly to projection.
> • Much cheaper than materialization: no need to store a temporary relation to disk.
> • Pipelining may not always be possible - e.g., sort, hash-join.
> • For pipelining to be effective, use evaluation algorithms that generate output tuples even as tuples are received for inputs to the operation.

Question sources

Computer Science MCQ practice collection · Question 242

Question 126 | AM0243

What is query result caching, specifically caching returned answers rather than execution plans or intermediate rows?

A. Storing query results for future reuse

B. Storing intermediate results during query execution

C. Storing database metadata for query optimization

D. Storing query plans for efficient execution

Hint

Distinguish overlapping dependent operators, independent workers and reused cached objects.

Answer A: A. Storing query results for future reuse

Result caching stores a query's returned rows or answer for future reuse. It needs suitable invalidation when underlying data changes. Plan caching instead stores the execution strategy without its returned rows. Intermediate rows retained during one execution are materialization, while database metadata supports optimization. Naming the cached object makes result caching distinct from these other useful storage mechanisms.

Why each choice is right or wrong

A - Correct: This stores final query answers for future reuse.

B - Incorrect: Intermediate-row storage is materialization rather than final-answer caching.

C - Incorrect: Metadata supports optimization but is not the query's returned answer.

D - Incorrect: Plan caching stores strategy rather than returned rows.

Primary note proof

Study-note section ADB22

> Parallel execution uses multiple workers, for example partitioned scans/joins, independent subplans or operator pipelines. Interleaving on one CPU is concurrency; simultaneous cores provide parallelism. Result caching reuses answers and needs invalidation when underlying data changes. Plan caching reuses an execution strategy, not the rows returned. Buffer caching keeps data pages; materialization retains intermediate rows. Specify the kind of query caching being discussed.

Study-note section ADB22

> Materialization stores an intermediate result before the next operator consumes it.

Study-note section ADB19

> Catalog statistics include row/page counts, distinct values and distributions.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 41 | lines 25-29

B. Cost Estimation Approach to Query Optimization
- The main idea is to minimize he cost of processing a query. The cost function is comprised of:
- I/O cost + CPU processing cost + communication cost + Storage cost
- These components might have different weights in different processing environments

Fundamental of Database Systems module | PDF page 104 | lines 19-25

• For Undo: Before a data item‘s AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
• For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
• Standard Recovery Terminology

Fundamental of Database Systems module | PDF page 41 | lines 11-15

- Compute common expressions once.
- If common expression appears more than once, and result not too large, store result and reuse it when required.
- Useful when querying views, as same expression is used to construct view each time.

Summary of Heuristics for Algebraic Optimization:

Fundamental of Database Systems module | PDF page 104 | lines 2-13

Data Caching
• Data items to be modified are first stored into database cache by the Cache Manager (CM) and after modification they are flushed (written) to the disk
• When DBMS request for read /write operation on some item
• It check the requested data item is in the cache or not
• If it is not, the appropriate disk block are copied to the cache
• If the cache is already full, some buffer replacement policy can be used .
Like
• Least recent used (LRU)
• FIFO
• While replacing buffers , first of all the updated value on that buffer should be saved on the appropriate block in the data base

Question sources

Computer Science MCQ practice collection · Question 243

Question 127 | AM0244

What is query parallelization in query processing?

A. Dividing a query into subqueries and executing them in parallel

B. Dividing a query into stages and executing them sequentially

C. Dividing a query into multiple queries and executing them in parallel

D. Dividing a query into multiple stages and executing them concurrently

Hint

Distinguish overlapping dependent operators, independent workers and reused cached objects.

Answer A: Dividing a query into subqueries and executing them in parallel

The fragmentation discussion says a transaction can be divided into several subqueries operating on fragments, increasing concurrency (p.117). A matches this parallel decomposition; it is distinct from pipelining a dependent stream of stages.

Why each choice is right or wrong

A - Correct: Independent fragment subqueries can run in parallel as described.

B - Incorrect: Sequential stages do not supply parallelism.

C - Incorrect: Replacing one query with multiple separate whole queries is not the stated subquery decomposition.

D - Incorrect: Concurrent stages describe pipelining when results flow between dependent operators, not the specific fragment-subquery approach.

Primary note proof

Study-note section ADB22

> Materialization stores an intermediate result before the next operator consumes it. Pipelining passes tuples as they become available, overlapping producer and consumer work and reducing temporary I/O. A sort is normally blocking; a hash join must build its hash table before probing can output, though later probe/output can stream. Pipelining does not require separate CPUs.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB22

> Parallel execution uses multiple workers, for example partitioned scans/joins, independent subplans or operator pipelines.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 117 | lines 15-31

6.2 Data Replication and Fragmentation: Distributed data storage
• There are two approaches to store the relation in the distributed database : Replication and Fragmentation
I. Data Replication
• The system maintain several identical copies of the relation & store each copy at a different site
• In general it enhance the performance of read operation and increase the availability of data to read only transaction. However, update transactions incur greater overhead
II. Data Fragmentation
- Split a relation into logically related and correct parts.
- The main reasons for fragmenting a relation are
• Efficiency- data that is not needed by the local applications is not stored
• Parallelism - a transaction can be divided into several subqueries that operate on fragments which will increase the degree of concurrency
- but reconstruction of the whole relation will require accessing data from all sites containing part of the relation

Question sources

Computer Science MCQ practice collection · Question 244

Question 128 | CS0066

If power went off while processing a transaction, what should the DBMS do to reverse any changes made to the database?

A. Read_transaction

B. Fail transaction

C. Commit transaction

D. Rollback

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer D: Rollback

To reverse changes of an incomplete failed transaction, recovery undoes its effects (p.103) by restoring before images (p.107). This is rollback. A committed transaction instead requires durability, so the answer assumes the interrupted transaction had not committed, as implied by reverse any changes.

Why each choice is right or wrong

A - Incorrect: Reading a transaction is not an undo operation.

B - Incorrect: Marking failure does not itself restore old values.

C - Incorrect: Committing would preserve rather than reverse the changes.

D - Correct: Rollback/undo is the direct reversal mechanism.

Primary note proof

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk. To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage. A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Study-note section ADB06

> Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure.

Module passages - related reading

These source passages provide context; the primary note proof is identified separately above.

Fundamental of Database Systems module | PDF page 97 | lines 4-13

> i. Purpose of Database Recovery
> ◦ To bring the database into the last consistent state, which existed prior to the failure.
> ◦ To preserve transaction properties (Atomicity &and Durability).
> • The recovery manager of a DBMS is responsible to ensure
> • atomicity by undoing the action of transaction that do not commit
> • Durability by making sure that all actions of committed transaction survive system crash
> ii. Types of Failure
> ◦ The database may become unavailable for use due to :

Fundamental of Database Systems module | PDF page 103 | lines 6-23

> • Recovery from catastrophic
> • If there is extensive damage to the wide portion of the database
> • This method restore a past copy of the database from the backup storage and reconstructs operation of a committed transaction from the back up log up to the time of failure
> • Recovery from non-catastrophic failure
> • When the database is not physically damaged but has be come in consistent
> • The strategy uses undoing and redoing some operations in order to restore to a consistent state : example
> • For instance,

• If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to redo (roll forward) transaction‘s updates.
• If transaction had not committed at failure time, recovery manager has to undo (rollback) any effects of that transaction for atomicity. Example,
Transaction Log
◦ For recovery from any type of failure data values prior to modification (BFIM -

Fundamental of Database Systems module | PDF page 107 | lines 8-23

Transaction Roll-back (Undo) and Roll-Forward (Redo)
• If transaction fails for what so ever reason after updating the data base it must be necessary to roll back
• If a transaction T is roll back , any transaction that has read the value of some data item X written by T must be also be roll back
• Similarly once S is roll back, any transaction R that has read the values of some data item Y written by S must aslo be roll back, and so on..
• To maintain such cases, a transaction‘s operations are redone or undone.
• Undo: Restore all BFIMs on to disk (Remove all AFIMs).
• Redo: Restore all AFIMs on to disk.
• Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen.
Data Update and Recovery Scheme : Three types
i. Deferred Update (No Undo/Redo)
◦ The data update goes as follows:

Question sources

Computer Science Exit Exam practice collection · Question 66

Question 129 | NS0051

What is the main difference between homogeneous and heterogeneous distributed databases?

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer: Homogeneous systems use compatible DBMS technology; heterogeneous systems integrate differing DBMSs, models, schemas or interfaces.

ADB33 defines the distinction through compatibility of database technology and representation. Operating-system differences can occur but do not define heterogeneity. Homogeneous sites need not have identical schemas or automatic replication. Heterogeneous integration requires explicit mappings for names, types and semantics, rather than assuming that common connectivity alone reconciles their meanings.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB33

> ODBC and JDBC provide connectivity interfaces; they do not automatically solve semantic differences or create one distributed transaction.

Study-note section ADB33

> Heterogeneous design must reconcile naming, types, units, identifiers, query capabilities and transaction semantics.

Question sources

Advanced Database Systems review questions · Question 1

Question 130 | NS0052

Why is data fragmentation useful in distributed databases?

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer: Fragmentation improves locality, manageable placement and parallelism by splitting data across useful site boundaries.

ADB36 splits rows with predicates and columns with projections while preserving reconstruction through unions or key joins. This can place frequently used data near its consumers and allow independent work on pieces. Correct fragments must preserve completeness and required disjointness. Fragmentation alone creates no spare copy, so losing the only site containing a fragment still makes it unavailable.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Partitioning improves locality and parallelism but alone creates no spare copy.

Study-note section ADB36

> Losing the only site containing a fragment makes that fragment unavailable, even while other sites continue.

Question sources

Advanced Database Systems review questions · Question 2

Question 131 | NS0053

What is the purpose of two-phase locking (2PL) in concurrency control?

Hint

Track the first lock release and the transaction-completion boundary.

Answer: 2PL guarantees conflict serializability by separating lock acquisition from release.

During the growing phase a transaction acquires or upgrades locks; after it releases any lock, the shrinking phase forbids acquiring new ones. For well-formed transactions this guarantees conflict serializability. Basic 2PL can still deadlock and permit cascading aborts, so strict or conservative variants add different protections. It is distinct from distributed two-phase commit.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB12

> Strict 2PL retains exclusive locks until commit/abort.

Question sources

Advanced Database Systems review questions · Question 3

Question 132 | NS0054

Why is write-ahead logging important for database recovery?

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer: WAL makes recovery information durable before its changed data pages, enabling safe undo/redo after crashes.

ADB27 requires needed update logs to reach stable storage before corresponding changed pages. A durable commit also forces required log records, including commit. Recovery can then use before-images for undo and after-images for redo if buffered data was lost. WAL orders durable writes; modifying an in-memory buffer before the log flush does not itself violate it.

Primary note proof

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk. To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage. A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> Redo installs logged after-images when necessary, usually forward.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Question sources

Advanced Database Systems review questions · Question 4

Question 133 | NS0055

How does a data warehouse differ from an operational database?

Hint

Distinguish operational transactions, analytical workloads and physical storage layout.

Answer: A warehouse integrates historical data for analysis; an operational database serves frequent business transactions.

OLTP handles account updates and other frequent operational transactions, while OLAP scans and summarizes large datasets. A warehouse commonly integrates sources through extraction, transformation and loading for historical analysis. These are workload tendencies rather than absolute restrictions: operational databases may retain history, and warehouses may receive updates. Schema and storage choices should follow the actual workload.

Primary note proof

Study-note section ADB41

> OLTP handles frequent operational transactions, such as account updates; OLAP scans and summarizes large datasets for analysis. A warehouse integrates data for historical analysis, often through extraction, transformation and loading. These are workload tendencies: OLTP can retain history and warehouses can be updated.

Question sources

Advanced Database Systems review questions · Question 5

Question 134 | NS0056

What does ACID stand for in the context of database transactions?

A. Atomicity, Consistency, Isolation, Durability

B. Accuracy, Consistency, Integrity, Durability

C. Atomicity, Concurrency, Isolation, Data

D. Availability, Consistency, Isolation, Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer A: Atomicity, Consistency, Isolation, Durability.

ADB06 names all four ACID properties and assigns distinct roles: all-or-none effects, preserved invariants, controlled interference and durable commits. Only A lists those names in the correct order. Accuracy, integrity, concurrency, data and availability are related database concepts but do not replace the letters of this established transaction acronym.

Why each choice is right or wrong

A - Correct: It lists the four ACID properties in the proper acronym order.

B - Incorrect: Accuracy and Integrity replace Atomicity and Isolation, changing the defined acronym.

C - Incorrect: Concurrency and Data incorrectly replace Consistency and Durability.

D - Incorrect: Availability is not the A in the ACID transaction acronym.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 1

Question 135 | NS0057

Which ACID property ensures that a transaction is treated as an indivisible unit - either all steps succeed or none are applied?

A. Consistency

B. Isolation

C. Atomicity

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: Atomicity.

The question specifies indivisibility of the transaction effects: partial success must not remain if the transaction fails. ADB06 assigns that guarantee to atomicity. Consistency concerns valid invariants, isolation controls interference, and durability protects already committed work. Those properties support correctness too, but none directly means all-or-none effects.

Why each choice is right or wrong

A - Incorrect: Consistency concerns preserving integrity rules rather than specifically making effects indivisible.

B - Incorrect: Isolation controls interactions between concurrent transactions rather than partial transaction application.

C - Correct: Atomicity applies every transaction effect together or discards the incomplete work.

D - Incorrect: Durability protects committed effects after failure rather than defining all-or-none completion.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 2

Question 136 | NS0058

Account A = 5,000 Birr, Account B = 8,000 Birr. After transferring 750 Birr from A to B, which values maintain CONSISTENCY?

A. A = 4,000, B = 9,000

B. A = 4,250, B = 8,750

C. A = 5,750, B = 7,250

D. A = 4,250, B = 8,000

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: A = 4,250 Birr and B = 8,750 Birr.

The requested transfer subtracts 750 from 5000 and adds 750 to 8000, giving 4250 and 8750. The invariant total remains 13000. A and C also sum to 13000, but implement a different amount or the reverse direction. Preserving the total alone does not implement the specified successful transfer; D loses the credit.

Why each choice is right or wrong

A - Incorrect: The sum remains 13000, but the changes transfer 1000 rather than the requested 750.

B - Correct: It debits 750 from A, credits 750 to B and preserves the 13000 total.

C - Incorrect: It preserves the total but transfers 750 in the reverse direction.

D - Incorrect: It debits A without crediting B, reducing the total to 12250.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB06

> Unchanged or reverse-transfer balances can also preserve that sum; only the requested debit/credit implements the specified successful transfer.

Study-note section ADB06

> A=5,000 and B=8,000 become 4,250 and 8,750 after transferring 750 from A to B.

Question sources

Advanced Database Systems review questions · Question 3

Question 137 | NS0059

Which ACID property guarantees that committed transaction changes persist even after a system crash?

A. Atomicity

B. Consistency

C. Isolation

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Durability.

ADB06 defines durability as preserving committed effects despite failure. A system may leave data pages buffered at commit provided durable recovery information can restore the changes after a crash. Atomicity controls partial outcomes, consistency protects declared invariants and isolation controls concurrent interference; they do not name this persistence guarantee.

Why each choice is right or wrong

A - Incorrect: Atomicity means all-or-none effects, not specifically persistence after a successful commit.

B - Incorrect: Consistency concerns valid integrity rules before and after correct work.

C - Incorrect: Isolation controls concurrent interference rather than persistence across a crash.

D - Correct: Durability ensures committed changes survive failures through the supported recovery mechanisms.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 4

Question 138 | NS0060

In the Read_item(X) operation, what is the THIRD and final step?

A. Store the updated block back to disk

B. Find the disk block address containing X

C. Copy the disk block into main memory buffer

D. Copy item X from the buffer to the program variable named x

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer D: Copy X from the buffer into the program variable x.

ADB07 first locates the page containing X, brings it into a buffer if absent and then copies X into a program variable. Thus D is the final conceptual read step. Locating and loading precede it, and storing an updated block to disk belongs to writing/flushing rather than completing read_item. A buffer hit can avoid physical I/O.

Why each choice is right or wrong

A - Incorrect: A read does not update and write the block back to disk.

B - Incorrect: Locating the containing page is the first conceptual step, not the final one.

C - Incorrect: Loading the page into a buffer precedes copying the item into the program variable.

D - Correct: The final read step copies the requested item from its buffer into x.

Primary note proof

Study-note section ADB07

> read_item(X) locates X's page, brings it into a buffer if absent, and copies X into a program variable. write_item(X) copies the new program value into the buffered page, marking it dirty. Writing that page to disk can occur later according to the buffer policy; write_item is not inherently an immediate disk flush. Insert and delete also modify data; sorting is a query-processing operation rather than a primitive item update.

Question sources

Advanced Database Systems review questions · Question 5

Question 139 | NS0061

A transaction that has started and is currently executing its operations is in which state?

A. Committed

B. Partially Committed

C. Active

D. Failed

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: Active.

ADB06 defines Active as the state while transaction operations are executing. Partially committed follows completion of the last operation but precedes assured durable commitment. Committed means commitment succeeded, while Failed means execution encountered a failure. Merely starting and continuing its operations therefore places this transaction in Active.

Why each choice is right or wrong

A - Incorrect: Committed requires successful commitment, which has not occurred while operations are still executing.

B - Incorrect: Partially committed follows the last operation rather than ordinary ongoing execution.

C - Correct: Active is the state during execution of the transaction operations.

D - Incorrect: Failed requires an execution failure, which the stem does not report.

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Question sources

Advanced Database Systems review questions · Question 6

Question 140 | NS0062

Which transaction state means all operations executed successfully but the commit has NOT been finalized yet?

A. Active

B. Partially Committed

C. Committed

D. Aborted

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Partially committed.

Finishing all operations does not yet establish durable commitment. ADB06 names this intermediate state Partially committed. The transaction becomes Committed only after commitment succeeds, or can fail before that boundary. Active describes operations still running, while Aborted means rollback completed, so neither matches successful operations awaiting finalized commit.

Why each choice is right or wrong

A - Incorrect: Active means operations are still executing, whereas the last operation has finished here.

B - Correct: Partially committed means execution finished but durable commitment is not yet assured.

C - Incorrect: Committed requires the commit itself to have succeeded, which the stem excludes.

D - Incorrect: Aborted follows rollback rather than successful execution awaiting a commit decision.

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Question sources

Advanced Database Systems review questions · Question 7

Question 141 | NS0063

What does the log record [abort, T] indicate?

A. Transaction T has just started execution

B. Transaction T committed successfully

C. Transaction T was aborted and ALL its changes must be discarded

D. Transaction T read a data item

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: T is aborting/aborted and its transaction effects must be discarded.

ADB07 includes ABORT among conceptual log records and warns that it can mark the start of rollback rather than prove every undo step is already complete. C is correct because it says the changes must be discarded, not that all undo has finished. START, COMMIT and ordinary read activity have different meanings.

Why each choice is right or wrong

A - Incorrect: A START record, rather than ABORT, marks beginning transaction execution.

B - Incorrect: COMMIT records successful completion; ABORT means the opposite outcome.

C - Correct: Abort requires discarding transaction effects, though this record need not prove all undo is already complete.

D - Incorrect: The record denotes abort handling rather than a read_item operation.

Primary note proof

Study-note section ADB07

> A conceptual log can contain START, UPDATE(T,X,old,new), COMMIT, ABORT and END records. Before-images support undo; after-images support redo. Real systems may use physical or logical records, and need not log every read. COMMIT records successful completion; an ABORT record can mark the start of rollback rather than prove every undo step is complete.

Question sources

Advanced Database Systems review questions · Question 8

Question 142 | NS0064

Which of the following is NOT one of the four basic transaction operations on a database?

A. Read

B. Write

C. Sort

D. Delete

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: Sort.

ADB07 distinguishes read_item and write_item and notes that inserts and deletes also modify data. Sorting is instead an operation used in query processing, not a primitive item update in this classification. The exact count four is a teaching convention, but the note directly supplies the relevant distinction that singles out Sort.

Why each choice is right or wrong

A - Incorrect: Reading is an explicitly defined primitive database item operation.

B - Incorrect: Writing is an explicitly defined primitive database item modification operation.

C - Correct: Sorting is a query-processing operation rather than the primitive item access/update category.

D - Incorrect: Delete modifies stored data and belongs with the listed primitive data operations.

Primary note proof

Study-note section ADB07

> read_item(X) locates X's page, brings it into a buffer if absent, and copies X into a program variable. write_item(X) copies the new program value into the buffered page, marking it dirty. Writing that page to disk can occur later according to the buffer policy; write_item is not inherently an immediate disk flush. Insert and delete also modify data; sorting is a query-processing operation rather than a primitive item update.

Question sources

Advanced Database Systems review questions · Question 9

Question 143 | NS0065

The Isolation property of ACID ensures that concurrent transactions:

A. Always read the same data at the same time

B. Do not interfere with each other; each executes independently

C. Commit in the same order they started

D. Always complete successfully without any errors

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Concurrent work is isolated from improper interference.

ADB06 defines isolation as controlling interference between concurrent transactions. B captures logical independence under the specified isolation guarantee; it does not mean transactions literally use separate hardware or never wait. Isolation does not require identical reads, start-order commits or universal successful completion. Strong serializable isolation makes behavior equivalent to some serial order.

Why each choice is right or wrong

A - Incorrect: Isolation does not require all transactions to read identical values at the same time.

B - Correct: Isolation controls interference so concurrent transactions retain the promised logical execution semantics.

C - Incorrect: Isolation does not require commit order to match transaction start order.

D - Incorrect: Conflicts and failures can still cause aborts; isolation is not a success guarantee.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB09

> A serializable schedule is equivalent to some serial order; it may interleave transactions.

Study-note section ADB09

> A serial schedule executes each transaction's operations consecutively.

Question sources

Advanced Database Systems review questions · Question 10

Question 144 | NS0066

In a simplified teaching sequence, write_item(X) updates a buffered page and then the buffer manager immediately flushes that page, after required WAL records are durable. What is the final physical step of this specified sequence?

A. Copy item X to program variable

B. Find the disk block address

C. Store the updated buffer block back to disk

D. Read the old value of X first

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: C. Store the updated buffer block back to disk

After X's program value has been copied into the buffered page, the specified sequence completes by writing the changed page back to disk. That is C. In ordinary buffering this flush may be deferred, but the question explicitly includes it. Locating a page or reading its old value occurs before or separately from the final physical output, and copying into a program variable describes reading.

Why each choice is right or wrong

A - Incorrect: Copying into a program variable is a read operation.

B - Incorrect: The page address is located before the final output.

C - Correct: The explicitly included flush stores the updated buffered page on disk.

D - Incorrect: Reading the old value is not the final physical output step.

Primary note proof

Study-note section ADB07

> read_item(X) locates X's page, brings it into a buffer if absent, and copies X into a program variable. write_item(X) copies the new program value into the buffered page, marking it dirty. Writing that page to disk can occur later according to the buffer policy; write_item is not inherently an immediate disk flush. Insert and delete also modify data; sorting is a query-processing operation rather than a primitive item update.

Study-note section ADB27

> A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage.

Question sources

Advanced Database Systems review questions · Question 11

Question 145 | NS0067

A transaction that encountered an error during execution and could not complete is in which state?

A. Aborted

B. Failed

C. Terminated

D. Partially Committed

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Failed.

ADB06 places a transaction in Failed when an error prevents successful execution. After rollback removes its effects it reaches Aborted. The stem reports the execution error but does not say rollback has completed, so Failed is the precise state. Partially committed follows successful final operation execution, not an error that prevents completion.

Why each choice is right or wrong

A - Incorrect: Aborted requires completed rollback, which is not stated after the execution error.

B - Correct: Failed is the immediate state when an error prevents completing the transaction.

C - Incorrect: Termination is later lifecycle handling rather than the immediate error state.

D - Incorrect: Partially committed follows completion of the last operation rather than failed execution.

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Question sources

Advanced Database Systems review questions · Question 12

Question 146 | NS0068

Transaction log records serve primarily for which purpose?

A. Speeding up query execution

B. Encrypting sensitive data

C. Enabling database recovery after failures by recording all changes

D. Enforcing access control permissions

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: Enable recovery by recording the information needed to undo or redo changes.

ADB07 records starts, updates, commits and aborts, with before-images for undo and after-images for redo. That makes recovery the primary purpose among the choices. Actual log formats may be physical or logical and need not record every read, so all changes should be understood as the change information required by the recovery design, not every operation indiscriminately.

Why each choice is right or wrong

A - Incorrect: The primary log purpose here is recovery rather than speeding query execution.

B - Incorrect: Transaction logging does not inherently encrypt the stored sensitive data.

C - Correct: Logged update and transaction-state information supports recovery of valid committed state after failures.

D - Incorrect: Access privileges control authorization; transaction log records are not themselves that enforcement mechanism.

Primary note proof

Study-note section ADB07

> A conceptual log can contain START, UPDATE(T,X,old,new), COMMIT, ABORT and END records. Before-images support undo; after-images support redo. Real systems may use physical or logical records, and need not log every read. COMMIT records successful completion; an ABORT record can mark the start of rollback rather than prove every undo step is complete.

Study-note section ADB24

> A user who can view but cannot change a record has restricted access privileges.

Study-note section ADB24

> Authentication establishes identity; authorization/access control determines permitted actions.

Question sources

Advanced Database Systems review questions · Question 13

Question 147 | NS0069

Which processing type uses multiple CPUs to execute different transactions simultaneously, achieving higher throughput?

A. Sequential processing

B. Interleaved processing

C. Parallel processing

D. Batch processing

Hint

Distinguish overlapping dependent operators, independent workers and reused cached objects.

Answer C: Parallel processing.

ADB22 distinguishes concurrency by interleaving on one CPU from actual simultaneous work on multiple cores. The stem explicitly uses multiple CPUs to execute different transactions at the same time, making Parallel the correct classification. Batch describes grouping work, and sequential processing performs work in order; neither guarantees the stated simultaneous execution.

Why each choice is right or wrong

A - Incorrect: Sequential processing does not execute different transactions simultaneously on multiple CPUs.

B - Incorrect: Interleaving alternates execution and need not provide simultaneous CPU work.

C - Correct: Multiple CPUs working simultaneously provide parallel execution.

D - Incorrect: Batch describes grouped submission or handling, not necessarily simultaneous multi-CPU execution.

Primary note proof

Study-note section ADB22

> Materialization stores an intermediate result before the next operator consumes it. Pipelining passes tuples as they become available, overlapping producer and consumer work and reducing temporary I/O. A sort is normally blocking; a hash join must build its hash table before probing can output, though later probe/output can stream. Pipelining does not require separate CPUs.

Study-note section ADB22

> Interleaving on one CPU is concurrency; simultaneous cores provide parallelism.

Study-note section ADB22

> Parallel execution uses multiple workers, for example partitioned scans/joins, independent subplans or operator pipelines.

Question sources

Advanced Database Systems review questions · Question 14

Question 148 | NS0070

Why are transactions important in database systems?

A. They make queries run faster

B. They ensure the database remains consistent and reliable through ACID properties

C. They allow the database to store more data

D. They simplify the SQL syntax for users

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Transactions support consistent, reliable work through ACID guarantees.

ADB06 groups operations into a logical unit with all-or-none effects, invariant preservation, isolation and durable commits. These guarantees support consistent and reliable database behavior when application logic is correct. They do not inherently speed queries, expand storage capacity or simplify syntax. In particular a DBMS cannot make an incorrect business operation consistent merely by wrapping it in a transaction.

Why each choice is right or wrong

A - Incorrect: Transactions can add coordination costs and do not inherently make queries faster.

B - Correct: ACID guarantees support valid and recoverable logical work, provided business logic is correct.

C - Incorrect: Transaction semantics do not increase the database's physical storage capacity.

D - Incorrect: Grouping work atomically does not inherently simplify the SQL language for users.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 15

Question 149 | NS0071

T1 reads balance=$500 and adds $100, planning to write $600. T2 reads the same $500, adds $200, and writes $700. T1 then writes $600. What problem occurred?

A. Dirty Read

B. Lost Update (Write-Write Problem)

C. Incorrect Summary Problem

D. Deadlock

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Lost update: T1 overwrites T2’s increment with its stale calculation.

Both transactions read 500. T2 writes 700 after adding 200, but T1 then writes its previously computed 600, erasing T2’s increment. A serial execution would produce 800. This matches the lost-update trace in ADB08; no waiting cycle or aggregate calculation is described, and reading an uncommitted value is not the stated cause.

Why each choice is right or wrong

A - Incorrect: Neither transaction is described as reading the other's uncommitted write.

B - Correct: T1's stale write of 600 overwrites T2's added 200; the correct combined result is 800.

C - Incorrect: No aggregate over mutually inconsistent versions is being calculated.

D - Incorrect: No circular waiting or blocked resource requests are present.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X.

Study-note section ADB08

> Incorrect summary mixes mutually inconsistent versions: a transfer moves 10 from A=100 to B=100; a sum reading old A=100 and new B=110 gets 210 instead of the invariant 200.

Question sources

Advanced Database Systems review questions · Question 16

Question 150 | NS0072

A dirty read occurs when transaction T1 reads data that:

A. T1 itself wrote earlier in the same transaction

B. T2 modified but has NOT yet committed, and T2 may later abort

C. Was permanently deleted from the database

D. Was last read more than one hour ago

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Read another transaction’s uncommitted modification.

ADB08 defines dirty read as consuming a value written by an uncommitted other transaction. If that writer aborts, the dependent read used a value that should not survive. Reading one’s own changes is allowed and is not this anomaly; the age of a value or a completed permanent deletion does not define dirty reading.

Why each choice is right or wrong

A - Incorrect: Reading changes made earlier by the same transaction is not a dirty read.

B - Correct: The read depends on T2's uncommitted update, which may later need undo.

C - Incorrect: Permanent deletion is not the defining uncommitted-value dependency.

D - Incorrect: Elapsed time since a read does not determine whether the writer committed.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB08

> A phantom means repeating a predicate query finds a changed qualifying set after another transaction inserts, deletes or changes matching rows.

Study-note section ADB08

> A single committed update from B=20 to B=50 does not alone prove an invalid summary of 90.

Question sources

Advanced Database Systems review questions · Question 17

Question 151 | NS0073

Which lock type allows MULTIPLE transactions to simultaneously READ a data item?

A. Exclusive Lock

B. Write Lock

C. Shared Lock

D. Binary Lock

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer C: Shared lock.

ADB11 states that a shared lock permits reading and multiple shared holders can coexist on the same item. Exclusive/write locks conflict with another transaction’s shared or exclusive lock on that item. Thus shared locking provides the requested concurrent-reader behavior. A generic binary locked/unlocked label does not establish shared-reader compatibility in the supplied model.

Why each choice is right or wrong

A - Incorrect: An exclusive lock conflicts with other transactions' read locks on the same item.

B - Incorrect: A write/exclusive lock does not grant multiple independent readers the requested shared ownership.

C - Correct: Shared locks are compatible with other shared locks, permitting multiple readers.

D - Incorrect: The binary label does not provide the explicit S/S compatibility defined in the note.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB-S04

> Shared/exclusive locking adds modes so multiple shared readers can coexist while incompatible exclusive access is excluded.

Study-note section ADB11

> One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Question sources

Advanced Database Systems review questions · Question 18

Question 152 | NS0074

In Two-Phase Locking (2PL), a transaction can acquire new locks ONLY during which phase?

A. Shrinking Phase

B. Termination Phase

C. Growing Phase

D. Validation Phase

Hint

Track the first lock release and the transaction-completion boundary.

Answer C: Growing phase.

Basic 2PL separates acquiring/upgrading locks from releasing/downgrading them. ADB12 permits acquisition only in the growing phase; after any release, no new lock may be acquired. Termination is a transaction boundary and validation belongs to a different control approach. The growing/shrinking rule is what distinguishes 2PL from unrestricted use of locks.

Why each choice is right or wrong

A - Incorrect: The shrinking phase releases or downgrades locks and forbids new acquisitions.

B - Incorrect: Termination is not the 2PL phase in which new locks are acquired.

C - Correct: Growing permits acquiring and upgrading locks before any release begins.

D - Incorrect: Validation is an optimistic-control stage, not the lock-acquisition phase of 2PL.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries.

Question sources

Advanced Database Systems review questions · Question 19

Question 153 | NS0075

What does Two-Phase Locking (2PL) guarantee that locking alone does NOT guarantee?

A. That no deadlocks will occur

B. Serializability - results equivalent to some serial execution

C. That all transactions complete without aborting

D. That every transaction reads the most current data

Hint

Track the first lock release and the transaction-completion boundary.

Answer B: Conflict serializability for well-formed transactions.

ADB12 explicitly guarantees conflict serializability for well-formed 2PL transactions. Locking without its phase discipline need not provide that ordering guarantee. Basic 2PL can still deadlock and may permit cascading aborts, so it does not guarantee successful completion. Nor does serializability require every read to observe the latest wall-clock update.

Why each choice is right or wrong

A - Incorrect: Basic 2PL can deadlock, so deadlock freedom is not its general guarantee.

B - Correct: The phase discipline guarantees conflict-serializable schedules for well-formed transactions.

C - Incorrect: Deadlock resolution or failures can still abort transactions.

D - Incorrect: Serializability does not require every read to see the newest wall-clock version.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB12

> Strict 2PL retains exclusive locks until commit/abort.

Question sources

Advanced Database Systems review questions · Question 20

Question 154 | NS0076

What is a deadlock in database systems?

A. A transaction that executes too slowly

B. A circular wait where each transaction holds a resource another needs, and none can proceed

C. A type of hardware failure causing data loss

D. When a transaction violates the ACID consistency property

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: A circular wait among transactions holding resources needed by others.

ADB13 defines deadlock through a waiting cycle: each participant holds a resource another needs, so none can advance without an intervention. Slow work alone can still make progress and is not deadlock. Hardware loss and consistency violations are distinct problems that do not require a resource-wait cycle.

Why each choice is right or wrong

A - Incorrect: A slow transaction may still make progress and need not participate in a wait cycle.

B - Correct: Circular hold-and-wait prevents the involved transactions from progressing without intervention.

C - Incorrect: Hardware failure is a different failure category and does not define a lock wait cycle.

D - Incorrect: A consistency violation concerns invalid data states rather than necessarily circular waiting.

Primary note proof

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

> A timeout aborts long waits but may mistake slow progress for deadlock.

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts.

Question sources

Advanced Database Systems review questions · Question 21

Question 155 | NS0077

In the Wait-Die deadlock prevention strategy, when a YOUNGER transaction needs a resource held by an OLDER transaction, the younger transaction:

A. Wounds (aborts) the older transaction

B. Waits for the older transaction to release the resource

C. Is immediately aborted and restarted

D. Receives the resource immediately due to higher priority

Hint

Identify requester and holder ages before applying the protocol's action.

Answer C: Abort the younger requester; retry with its original timestamp.

The requester is younger and the resource holder older. ADB14 says wait-die permits an older requester to wait but aborts a younger requester needing an older holder. C therefore gives the correct action. Restart should retain the original timestamp to improve relative priority; immediately aborted does not require an immediate successful retry or resource grant.

Why each choice is right or wrong

A - Incorrect: Wait-die does not let the younger requester abort the older holder.

B - Incorrect: In wait-die, the younger requester dies rather than waiting for the older holder.

C - Correct: The younger requester is aborted and may restart using its original timestamp.

D - Incorrect: The older holder retains priority; the younger requester does not receive the resource immediately.

Primary note proof

Study-note section ADB14

> A smaller timestamp means an older transaction. In wait-die, an older requester may wait for a younger holder; a younger requester needing an older holder aborts itself. In wound-wait, an older requester aborts the younger holder; a younger requester waits for the older holder. Always identify requester versus holder before applying the names.

Question sources

Advanced Database Systems review questions · Question 22

Question 156 | NS0078

In the Wound-Wait strategy, when an OLDER transaction needs a resource held by a YOUNGER transaction:

A. The older transaction waits for the younger

B. The older transaction wounds (aborts) the younger transaction

C. Both transactions are simultaneously aborted

D. A deadlock detection algorithm runs

Hint

Identify requester and holder ages before applying the protocol's action.

Answer B: The older requester aborts the younger holder.

ADB14 distinguishes wound-wait from wait-die: an older requester wounds the younger holder, while a younger requester waits for an older holder. Here the requester is explicitly older, so B applies. The rule prevents waiting cycles through age-based action; it does not require detecting an existing deadlock first or aborting both participants.

Why each choice is right or wrong

A - Incorrect: An older requester wounds the younger holder rather than waiting under wound-wait.

B - Correct: The older requester causes the younger resource holder to abort and release its lock.

C - Incorrect: The protocol aborts the younger holder, not both transactions.

D - Incorrect: This is a prevention action based on ages, not a requirement to run cycle detection first.

Primary note proof

Study-note section ADB14

> A smaller timestamp means an older transaction. In wait-die, an older requester may wait for a younger holder; a younger requester needing an older holder aborts itself. In wound-wait, an older requester aborts the younger holder; a younger requester waits for the older holder. Always identify requester versus holder before applying the names.

Study-note section ADB13

> A cycle indicates deadlock under the ordinary lock-wait model.

Study-note section ADB13

> A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp.

Question sources

Advanced Database Systems review questions · Question 23

Question 157 | NS0079

Optimistic concurrency control is BEST suited for environments where:

A. Transactions frequently write to the same data items

B. Write operations always conflict with read operations

C. Conflicts between transactions are rare (read-heavy workloads)

D. All transactions must complete in strict timestamp order

Hint

Consider conflicts, private work and validation costs, or lock-management overhead.

Answer C: Conflicts between transactions are rare (read-heavy workloads)

Optimistic concurrency control lets transactions read and compute privately, then validates overlapping read/write activity before publishing changes. When conflicts are rare, most work can commit without being discarded. Frequent competing writes or persistent read/write conflicts instead lead to costly retries. Strict timestamp completion describes a separate ordering constraint; it does not supply the low-conflict condition that makes optimistic execution suitable. Read-heavy activity is useful here only insofar as actual conflicts remain uncommon.

Why each choice is right or wrong

A - Incorrect: Incorrect: frequent competing writes cause validation failures and discarded work.

B - Incorrect: Incorrect: persistent read/write conflicts increase retry costs rather than favoring optimistic execution.

C - Correct: Correct: rare conflicts allow most private work to validate and commit.

D - Incorrect: Incorrect: a required completion order does not imply low contention; optimistic suitability depends on conflicts and retry cost.

Primary note proof

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries. It works well when conflicts are uncommon, but high contention wastes work on aborts.

Question sources

Advanced Database Systems review questions · Question 24

Question 158 | NS0080

Lock upgrade refers to changing a lock from:

A. Exclusive → Shared

B. Shared → Exclusive

C. No lock → Shared

D. Exclusive → No lock

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer B: Shared → Exclusive.

ADB11 explicitly names S→X an upgrade and X→S a downgrade. The stronger exclusive lock permits writing and can be granted only after conflicting other holders leave. Acquiring a new shared lock is initial acquisition rather than conversion, while releasing an exclusive lock is unlocking. Thus B alone names a lock upgrade.

Why each choice is right or wrong

A - Incorrect: Exclusive-to-shared weakens permissions and is a downgrade.

B - Correct: Shared-to-exclusive strengthens the held lock to permit writing, subject to compatibility.

C - Incorrect: No-lock-to-shared is initial acquisition, not conversion of an already held lock.

D - Incorrect: Exclusive-to-no-lock is release rather than upgrade.

Primary note proof

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks. S→X is an upgrade, possible only after conflicting other holders leave. X→S is a downgrade. The lock manager decides requests; the lock table stores granted locks and waiting requests. One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy. No double-locking is not a prohibition on shared readers.

Question sources

Advanced Database Systems review questions · Question 25

Question 159 | NS0081

Which trace demonstrates an incorrect-summary anomaly? The intended anomaly is a mixed-version aggregate, distinct from a nonrepeatable read.

A. Two transactions both delete the same record simultaneously

B. Initially A=B=100. During T2's transfer of 10 from A to B, T1 sums old A=100 and new B=110, returning 210 although the total-balance invariant is 200.

C. A transaction reads uncommitted data from another transaction

D. A transaction fails to acquire all necessary locks

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: B. Initially A=B=100. During T2's transfer of 10 from A to B, T1 sums old A=100 and new B=110, returning 210 although the total-balance invariant is 200.

The transfer preserves the total balance at 200, but T1 combines the pre-transfer debit balance with the post-transfer credit balance. Its computed 100+110=210 is not a valid summary of a consistent state. This is an incorrect-summary anomaly. Dirty reading is defined by an uncommitted dependency, and lock failure or simultaneous deletion alone does not establish this mixed aggregate.

Why each choice is right or wrong

A - Incorrect: Concurrent deletion alone does not specify an inconsistent aggregate.

B - Correct: The mixed old/new balances yield 210 instead of the invariant 200.

C - Incorrect: Reading another transaction's uncommitted value defines a dirty read.

D - Incorrect: Failure to acquire locks concerns scheduling rather than this aggregate result.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB08

> Incorrect summary mixes mutually inconsistent versions: a transfer moves 10 from A=100 to B=100; a sum reading old A=100 and new B=110 gets 210 instead of the invariant 200.

Study-note section ADB08

> A single committed update from B=20 to B=50 does not alone prove an invalid summary of 90.

Question sources

Advanced Database Systems review questions · Question 26

Question 160 | NS0082

Starvation in a database system occurs when:

A. A transaction consumes excessive memory resources

B. A transaction is repeatedly bypassed by higher-priority transactions and never executes

C. A deadlock persists longer than the timeout period

D. The transaction log becomes completely full

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: Indefinite postponement while other transactions keep progressing.

ADB13 defines starvation as indefinite postponement while other work proceeds, without requiring any waiting cycle. Repeatedly favoring higher-priority work can prevent a transaction from executing, matching B. Memory consumption, a full log or a long deadlock are different resource/failure conditions; none alone defines the fairness failure of starvation.

Why each choice is right or wrong

A - Incorrect: Excessive memory use is a resource problem rather than the definition of indefinite postponement.

B - Correct: Repeated bypass by other transactions can indefinitely postpone this transaction while work continues.

C - Incorrect: Deadlock requires a waiting cycle; starvation can occur without one.

D - Incorrect: A full log can block work but does not itself define unfair indefinite postponement.

Primary note proof

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

> Starvation is indefinite postponement while other work proceeds, without a required cycle.

Study-note section ADB13

> Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Question sources

Advanced Database Systems review questions · Question 27

Question 161 | NS0083

Which concurrency control technique assigns a unique start-time value to each transaction and uses this to order execution and resolve conflicts?

A. Two-Phase Locking

B. Shadow Paging

C. Timestamp-Based Concurrency Control

D. Optimistic Concurrency Control

Hint

Compare the transaction timestamp with the item's recorded read and write timestamps.

Answer C: Timestamp-based concurrency control.

ADB15 assigns TS(T) to each transaction and compares it with item read_TS and write_TS values. Those tests reject operations that would violate timestamp order, so C matches the start-time ordering technique. 2PL orders locking, optimistic control validates private work, and shadow paging is a recovery/storage technique rather than this conflict-ordering protocol.

Why each choice is right or wrong

A - Incorrect: 2PL orders lock acquisition and release rather than defining the timestamp validity tests.

B - Incorrect: Shadow paging preserves page versions for recovery rather than ordering concurrent transaction operations by timestamp.

C - Correct: Timestamp ordering compares transaction timestamps with item read/write timestamps to resolve conflicts.

D - Incorrect: Optimistic control validates private read/write work rather than being defined by these timestamp-order tests.

Primary note proof

Study-note section ADB15

> Assign each transaction TS(T). For item X track read_TS(X), the largest timestamp of a successful reader, and write_TS(X), that of its latest successful writer. A read by T aborts if TS(T)<write_TS(X); otherwise it succeeds and updates read_TS. A write aborts if TS(T)<read_TS(X) or TS(T)<write_TS(X); otherwise it succeeds and sets write_TS. These tests preserve timestamp order.

Study-note section ADB16

> Multiversion control keeps versions so a reader can use an appropriate snapshot.

Study-note section ADB29

> Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers.

Related reading (context, not primary note proof)

Study-note section ADB12

Question sources

Advanced Database Systems review questions · Question 28

Question 162 | NS0084

A well-formed transaction must follow which rule regarding locking?

A. Lock all items at the end of the transaction

B. Lock a data item BEFORE reading or writing it

C. Lock items only when modifying them, not when reading

D. Acquire all locks in alphabetical order by item name

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer B: Acquire the appropriate lock before accessing the item.

ADB11 requires ownership of an appropriate lock before access and permits releasing only one's own locks. Reads therefore need shared or suitable stronger locks in this basic model; writes need exclusive ownership. Alphabetical acquisition may be a separate prevention convention, but it is not the general well-formedness rule and locks cannot be postponed until after access.

Why each choice is right or wrong

A - Incorrect: Locking only at the end leaves the earlier accesses unprotected.

B - Correct: A well-formed transaction holds an appropriate lock before reading or writing an item.

C - Incorrect: Basic lock-based access also requires an appropriate read lock, not only locks for writes.

D - Incorrect: A fixed resource order can help prevention, but alphabetical order is not the general well-formedness definition.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks.

Study-note section ADB13

> Aborting every deadlocked transaction also breaks cycles but wastes more work.

Question sources

Advanced Database Systems review questions · Question 29

Question 163 | NS0085

Lock timeout as a deadlock solution works by:

A. Detecting cycles in the wait-for graph

B. Building a priority queue of all waiting transactions

C. Automatically aborting a transaction that has waited too long for a lock

D. Locking all required data items before execution begins

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer C: Abort a transaction after its lock wait exceeds the timeout.

ADB13 describes timeout as aborting long waits, while warning it may mistake slow progress for deadlock. C directly states that mechanism. Detecting cycles is a different method; acquiring all needed locks first is conservative locking; a priority queue alone neither defines a timeout nor guarantees elimination of waiting cycles.

Why each choice is right or wrong

A - Incorrect: Cycle detection is graph-based deadlock detection rather than a wait-duration timeout.

B - Incorrect: A priority queue alone does not enforce the stated wait-duration abort policy.

C - Correct: The policy aborts work whose lock wait exceeds the chosen threshold.

D - Incorrect: Obtaining all locks before execution is conservative locking, not timeout handling.

Primary note proof

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

> A timeout aborts long waits but may mistake slow progress for deadlock.

Study-note section ADB12

> Conservative/static 2PL obtains all needed locks before execution and does not begin while only a subset is available, preventing hold-and-wait deadlock.

Question sources

Advanced Database Systems review questions · Question 30

Question 164 | NS0086

What are the three phases of query processing in the correct order?

A. Optimization, Parsing, Evaluation

B. Parsing & Translation, Optimization, Evaluation

C. Evaluation, Translation, Optimization

D. Parsing, Evaluation, Optimization

Hint

Separate query representation, strategy selection and execution.

Answer B: Parsing/translation → optimization → evaluation.

ADB17 checks grammar and semantics, constructs a logical representation, chooses an equivalent efficient execution plan and evaluates it. B groups parsing and translation into the first phase and preserves that dependency order. Plans cannot sensibly be evaluated before construction and selection, and optimization does not precede understanding the query's syntax and meaning.

Why each choice is right or wrong

A - Incorrect: Optimization cannot precede parsing and understanding the query in the stated pipeline.

B - Correct: It preserves analysis/translation, plan optimization and then execution/evaluation.

C - Incorrect: Evaluation cannot be the first phase before the query is represented and planned.

D - Incorrect: It executes before selecting the optimized plan, reversing optimization and evaluation.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB17

> DDL is not a mandatory stage before each SELECT, so authentication→DDL→DML permutations do not describe the processing sequence.

Question sources

Advanced Database Systems review questions · Question 31

Question 165 | NS0087

During query translation, SQL is typically converted into which internal representation for processing?

A. Machine code instructions

B. Binary decision trees

C. Relational algebra expressions

D. Compressed byte code

Hint

Separate query representation, strategy selection and execution.

Answer C: Relational algebra expressions or a logical query tree.

ADB17 states that translation constructs a logical representation, typically relational algebra or a logical query tree. This allows equivalent rewrites and plan choice before physical execution. Machine code or bytecode may appear in particular execution implementations, but they are not the general logical translation target asked for here; a binary decision tree is also not the defined representation.

Why each choice is right or wrong

A - Incorrect: Machine code is not the logical relational representation described for query translation.

B - Incorrect: A binary decision tree is not the note's general query-algebra representation.

C - Correct: Relational algebra expresses the logical operators that can be rewritten and optimized.

D - Incorrect: Compressed bytecode is not the stated general logical query representation.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> A query tree has base relations at leaves and operators at internal nodes; a query graph emphasizes relations and predicates.

Study-note section ADB17

> A physical execution plan adds scan/index choices and algorithms.

Question sources

Advanced Database Systems review questions · Question 32

Question 166 | NS0088

The query optimizer's primary goal is to:

A. Ensure queries return results in sorted order

B. Find an execution plan that minimizes the overall cost of retrieving data

C. Maximize the number of indexes used in the query

D. Ensure all tables are accessed in the order written in SQL

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: Choose a low-cost equivalent execution plan.

ADB19 estimates resource costs such as I/O, CPU, memory, temporary storage and communication for candidate plans. Optimization compares equivalent ways to return the same result and chooses a suitable low-cost strategy. It does not seek index count, force textual table order or impose sorting that the query never requested. The estimate need not prove a global real-world minimum.

Why each choice is right or wrong

A - Incorrect: Sorting is required only when query semantics need it, not as the universal optimizer goal.

B - Correct: The optimizer seeks an efficient equivalent plan under the selected resource-cost model.

C - Incorrect: More indexes used does not necessarily mean lower cost or better execution.

D - Incorrect: Equivalent join reordering can reduce cost; SQL textual order need not dictate access order.

Primary note proof

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic. Catalog statistics include row/page counts, distinct values and distributions. Selection with fraction s predicts s×N rows. For non-NULL equijoin keys, a common estimate is $|R|\times|S|/\max(V(R,k),V(S,k))$, assuming uniform values and containment of the smaller distinct-value domain within the larger. Adjust counts for NULLs; if either non-NULL input is empty, the result is zero. Skew, correlations or incomplete domain overlap can invalidate this estimate; known disjoint domains give zero matches.

Study-note section ADB23

> Avoid universal rules that joins always beat subqueries, functions can never use indexes or more indexes always improve performance.

Study-note section ADB18

> Cost-based optimization estimates candidate plans instead of relying on rules alone.

Question sources

Advanced Database Systems review questions · Question 33

Question 167 | NS0089

In the textbook local rotating-disk cost model, data is not memory-resident and query execution is disk-bound; client/network and CPU costs are negligible by comparison. Which component dominates the estimated query cost?

A. CPU processing time

B. Disk I/O time (seek time + transfer time)

C. Memory allocation overhead

D. Network latency to the client

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: B. Disk I/O time (seek time + transfer time)

The specified disk-bound model estimates the cost of seeks and transferred blocks. Because CPU, memory-management and network costs are negligible by assumption, disk I/O dominates this query's estimated cost. This is a contextual model rather than a universal fact about all databases: memory-resident or distributed workloads can have different dominant resources. The answer follows the scenario's stated objective.

Why each choice is right or wrong

A - Incorrect: CPU cost is negligible in the specified model.

B - Correct: Seek and transfer time dominate the stated disk-bound workload.

C - Incorrect: Memory-management overhead is negligible here.

D - Incorrect: Client/network cost is negligible in this local scenario.

Primary note proof

Study-note section ADB19

> For a simple rotating-disk model, cost = seeks×seek_time + transferred_blocks×block_transfer_time. Five seeks at 5 ms plus ten blocks at 1 ms cost 35 ms. Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans. State the assumed cost objective before ranking strategies.

Question sources

Advanced Database Systems review questions · Question 34

Question 168 | NS0090

In the disk I/O cost formula Cost = (S x tS) + (B x tT), what does 'B' represent?

A. Block size in bytes

B. Number of disk blocks transferred

C. Buffer pool size

D. Binary search operations

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: The number of transferred disk blocks.

ADB19 computes disk cost as seeks times seek duration plus transferred blocks times per-block transfer time. Therefore B in the formula is a block count, making B×tT the total transfer time. Block size, buffer capacity and search-operation count are different quantities and would not have the units required by this expression.

Why each choice is right or wrong

A - Incorrect: Block size is bytes per block rather than the number multiplied by per-block transfer time.

B - Correct: B counts transferred blocks, so B×tT gives their total transfer time.

C - Incorrect: Buffer-pool capacity is not the transferred-block count in this formula.

D - Incorrect: Binary-search operation count is not the disk-block transfer term.

Primary note proof

Study-note section ADB19

> For a simple rotating-disk model, cost = seeks×seek_time + transferred_blocks×block_transfer_time. Five seeks at 5 ms plus ten blocks at 1 ms cost 35 ms. Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans. State the assumed cost objective before ranking strategies.

Question sources

Advanced Database Systems review questions · Question 35

Question 169 | NS0091

The UNDO operation in database recovery is used to:

A. Re-apply changes of committed transactions not yet written to disk

B. Create new checkpoint entries in the log

C. Reverse changes made by incomplete or aborted transactions

D. Transfer data from cache to permanent disk storage

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer C: Reverse incomplete or aborted transaction effects.

ADB27 uses undo to restore before-images of incomplete effects, usually traversing a transaction’s updates backward. This removes changes that must not survive. Reapplying committed changes is redo, checkpoint records bound recovery searches, and transferring dirty buffers to storage is page flushing. Only C describes undo rather than another recovery or storage operation.

Why each choice is right or wrong

A - Incorrect: Reapplying missing committed updates is redo rather than undo.

B - Incorrect: Checkpoint creation records recovery state and is not reversal of incomplete effects.

C - Correct: Undo restores old values to remove incomplete or aborted transaction changes.

D - Incorrect: Writing a buffer to disk is flushing rather than restoring before-images.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history.

Study-note section ADB30

> In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo.

Related reading (context, not primary note proof)

Study-note section ADB07

Question sources

Advanced Database Systems review questions · Question 36

Question 170 | NS0092

The REDO operation in database recovery is used to:

A. Abort all active transactions found in the log

B. Reverse changes made by failed transactions

C. Re-apply changes of committed transactions not yet written to disk

D. Create shadow copies of database pages

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer C: Reapply required logged after-images, including committed changes missing from pages.

ADB27 describes redo as installing logged after-images when necessary. In the simple teaching model it preserves committed work not yet on disk, matching C. ARIES may repeat history more broadly before loser undo, so C describes the ordinary purpose rather than every algorithm's full phase. Aborting transactions and reversing failed changes are different actions.

Why each choice is right or wrong

A - Incorrect: Aborting active transactions is not the operation of applying logged after-images.

B - Incorrect: Reversing incomplete effects is undo rather than redo.

C - Correct: Redo can restore committed changes whose data pages were not durable at the crash.

D - Incorrect: Shadow-page creation is a separate copy-on-write technique rather than logged redo.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB29

> Shadow/copy-on-write update writes a new location while preserving the committed version.

Study-note section ADB29

> Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable.

Question sources

Advanced Database Systems review questions · Question 37

Question 171 | NS0093

What is the fundamental durable-write ordering rule of write-ahead logging? In B, 'actual data changes' means writing the corresponding changed data page to persistent storage, not modifying a volatile buffer.

A. Write data to disk before creating the log record

B. Log records must be written to stable storage BEFORE making the actual data changes

C. All transactions must commit before any log entries are written

D. Log records are written only after a system crash occurs

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer B: B. Log records must be written to stable storage BEFORE making the actual data changes

WAL requires the recovery log for an update to reach stable storage before its corresponding changed page is written to disk. A volatile buffer may already contain the change, so this rule does not order every CPU assignment after a log flush. A reverses the required durable order; C and D postpone logs until too late to support recovery of earlier persistent changes.

Why each choice is right or wrong

A - Incorrect: Data-before-log reverses WAL's required durable ordering.

B - Correct: With the defined durable-write scope, the log reaches stable storage first.

C - Incorrect: Update logs may need to be durable before commit and before data flushing.

D - Incorrect: Recovery requires logs that were durable before the crash.

Primary note proof

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk. To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage. A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not.

Study-note section ADB27

> Redo installs logged after-images when necessary, usually forward.

Question sources

Advanced Database Systems review questions · Question 38

Question 172 | NS0094

In a simple quiescent checkpoint exercise, all transactions have completed and the required log/data pages are flushed at the checkpoint. How does that checkpoint reduce later recovery search work?

A. Preventing all types of hardware failures

B. Encrypting log records for faster access

C. Providing a known consistent state so recovery can start from there instead of the log beginning

D. Automatically committing all active transactions periodically

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer C: C. Providing a known consistent state so recovery can start from there instead of the log beginning

The quiescent checkpoint supplies a durable boundary for the simplified exercise. With no active transaction spanning it and the required pages flushed, recovery need not scan all earlier history to redo those completed changes. C captures that bounded search. Checkpointing does not prevent hardware failure, gain its benefit from encryption or automatically commit active transactions. Fuzzy checkpoints require different starting-point qualifications.

Why each choice is right or wrong

A - Incorrect: Recording recovery state does not prevent hardware failures.

B - Incorrect: Encryption is not the mechanism reducing log-search work.

C - Correct: The specified durable checkpoint bounds the history needed in this exercise.

D - Incorrect: Transactions have completed here; a checkpoint does not automatically commit active work.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB30

> A checkpoint is neither a full backup nor an automatic commit of active transactions.

Question sources

Advanced Database Systems review questions · Question 39

Question 173 | NS0095

In deferred update (No-Undo) recovery, actual database modifications are written to disk:

A. Immediately as each SQL statement executes

B. Only AFTER the transaction successfully commits

C. When a checkpoint operation is triggered

D. At regular 5-minute intervals regardless of transaction state

Hint

Separate update timing from update location and follow the commit boundary.

Answer B: After successful commit, in the deferred-update model.

ADB29 keeps uncommitted changes outside the persistent database. Only committed effects may therefore be installed in its actual database pages, which is the property B expresses. This does not require every committed page to be flushed immediately. Checkpoints and periodic timers cannot independently authorize writing uncommitted changes while retaining the claimed No-Undo model.

Why each choice is right or wrong

A - Incorrect: Immediate per-statement persistent writes can expose uncommitted effects and do not define deferred update.

B - Correct: The model does not install uncommitted effects in the persistent database; committed changes can be written later.

C - Incorrect: A checkpoint alone does not make an uncommitted transaction eligible for persistent installation.

D - Incorrect: A fixed timer cannot ignore transaction commit status while preserving deferred-update semantics.

Primary note proof

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

> Shadow/copy-on-write update writes a new location while preserving the committed version.

Study-note section ADB29

> Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable.

Question sources

Advanced Database Systems review questions · Question 40

Question 174 | NS0096

Shadow paging recovery technique preserves the original database pages by:

A. Creating a backup on a separate server

B. Writing all modifications to NEW shadow pages, keeping originals intact until commit

C. Encrypting pages before any modification

D. Storing pages in sorted order for faster access

Hint

Separate update timing from update location and follow the commit boundary.

Answer B: Write changed data to new pages while preserving the committed page map.

ADB29 preserves the committed version and writes changed data at new locations. Shadow paging keeps the old page map plus a modified map; commit atomically installs the new durable root after needed pages are durable. B captures the copy-on-write preservation, although the unchanged old map is the shadow rather than the name of each newly modified page.

Why each choice is right or wrong

A - Incorrect: A separate-server backup is not the page-map copy-on-write mechanism of shadow paging.

B - Correct: Changes use new pages, preserving the originals until the new committed map is safely installed.

C - Incorrect: Encryption does not preserve the old page mapping or provide the stated rollback mechanism.

D - Incorrect: Sorting pages is an access-layout choice rather than preserving committed copies.

Primary note proof

Study-note section ADB29

> In-place update overwrites the original location. Shadow/copy-on-write update writes a new location while preserving the committed version. Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable. Before commit, a crash uses the old map. It is an

alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Study-note section ADB26

A full backup copies the database to separate storage; scheduled backups plus archived logs support restoration to a chosen recoverable point.

Study-note section ADB26

Rollback handles failed/uncommitted work; redo preserves committed work missing from pages; checkpoints reduce log-search work; backups provide another stored copy.

Related reading (context, not primary note proof)

Study-note section ADB24

Question sources

Advanced Database Systems review questions · Question 41

Question 175 | NS0097

Which type of database failure is caused by accidental deletion of important data by a user?

A. Hardware failure

B. Natural disaster

C. Software error

D. Human error

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer D: Human error.

The described cause is an accidental user action, not faulty hardware, a natural event or a software defect. ADB26 treats human mistakes separately and notes they may need point-in-time restoration before a bad committed operation. Ordinary crash redo would otherwise preserve a committed deletion rather than recognize the user’s intention.

Why each choice is right or wrong

A - Incorrect: No hardware malfunction is stated; the cause is the user's mistaken deletion.

B - Incorrect: A natural disaster is not the cause described in the prompt.

C - Incorrect: The software may have executed the requested deletion correctly; the stated mistake belongs to the user.

D - Correct: Accidental deletion by a user is a human error.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB-S05

> An accidental user deletion is human error even if the software executes the requested command correctly.

Study-note section ADB-S05

> A software failure originates in erroneous program behavior, such as a logical programming defect, invalid operation or operating-system fault.

Question sources

Advanced Database Systems review questions · Question 42

Question 176 | NS0098

The transaction log records 'before and after values' for data modifications primarily to support:

A. Query optimization

B. User access control

C. UNDO (restore old value) and REDO (apply new value) operations during recovery

D. Database compression and storage efficiency

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer C: Undo with old values and redo with logged new values.

ADB27 restores before-images when undoing incomplete work and installs after-images when redoing logged effects. Recording both values therefore supports reversal and reapplication after failure. Query costing uses catalog statistics, access control uses privileges, and compression is a different storage concern; none explains the specific recovery value pair as directly.

Why each choice is right or wrong

A - Incorrect: Query optimization estimates plan cost rather than restoring old/new transaction values.

B - Incorrect: Access control checks privileges rather than using before/after images to authorize users.

C - Correct: Old values support reversal and new values support necessary reapplication during recovery.

D - Incorrect: Recording two values can add storage overhead and is not primarily a compression mechanism.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic.

Study-note section ADB19

> Catalog statistics include row/page counts, distinct values and distributions.

Related reading (context, not primary note proof)

Study-note section ADB24

Question sources

Advanced Database Systems review questions · Question 43

Question 177 | NS0099

Which type of distributed database uses the SAME DBMS software at every site?

A. Heterogeneous distributed database

B. Homogeneous distributed database

C. Middleware-based database

D. Partially replicated database

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer B: Homogeneous distributed database.

ADB33 identifies homogeneous systems by compatible DBMS technology across sites. Using the same DBMS software fits that classification. Heterogeneous systems integrate differing engines, models or interfaces. Middleware describes integration architecture and partial replication describes placement of copies, so neither substitutes for the homogeneity classification asked for here.

Why each choice is right or wrong

A - Incorrect: Heterogeneous systems integrate differing database technologies rather than the same DBMS throughout.

B - Correct: Using the same DBMS at each site fits homogeneous distributed database technology.

C - Incorrect: Middleware describes communication/integration responsibilities, not whether DBMS technology is the same.

D - Incorrect: Partial replication describes where selected copies reside rather than which DBMS software sites use.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB34

> These describe communication responsibilities, not fragmentation types.

Question sources

Advanced Database Systems review questions · Question 44

Question 178 | NS0100

Horizontal fragmentation divides a table based on:

A. Splitting columns into separate subsets

B. Splitting rows into subsets based on a condition

C. Combining two tables into one larger table

D. Replicating the table to multiple sites

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Split rows by predicates.

ADB36 defines horizontal fragmentation as selecting row subsets using predicates and reconstructing the relation by union. Column subsets belong to vertical fragmentation. Combining tables is not fragmentation, and copying a whole table is replication. Thus B identifies both the item being split and the rule used to decide fragment membership.

Why each choice is right or wrong

A - Incorrect: Column splitting is vertical fragmentation rather than horizontal fragmentation.

B - Correct: Horizontal fragments contain rows selected by stated predicates.

C - Incorrect: Combining tables increases the combined relation rather than splitting one table into row subsets.

D - Incorrect: Replicating copies preserves the whole table at more sites rather than partitioning its rows.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Question sources

Advanced Database Systems review questions · Question 45

Question 179 | NS0101

In vertical fragmentation, why must the PRIMARY KEY be included in every fragment?

A. To speed up INSERT operations

B. To prevent duplicate rows

C. To enable reconstruction of the original table by joining fragments on the primary key

D. To ensure proper replication across sites

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer C: A repeated key or row identifier lets vertical pieces join back losslessly.

ADB36 repeats a key or row identifier across vertical fragments so attributes from the same original row can be matched during reconstruction. C states that purpose. The general requirement is a suitable identifier, not necessarily the particular declared primary key if another row identifier is used. It is unrelated to mandatory replication or inherently faster inserts.

Why each choice is right or wrong

A - Incorrect: Faster insertion is not why vertical fragments share a row identifier.

B - Incorrect: Avoiding duplicate values is not sufficient to align column pieces from the same original row.

C - Correct: The shared key identifies matching original rows when the column fragments are joined.

D - Incorrect: Replication copies data and is distinct from reconstructing vertical column fragments.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Hybrid fragmentation combines row and column splits in either suitable order.

Study-note section ADB36

> Reconstruct by reversing the operations: key joins for corresponding vertical pieces and unions for horizontal pieces.

Question sources

Advanced Database Systems review questions · Question 46

Question 180 | NS0102

Which data replication scheme stores each data fragment at ONLY ONE site with NO copies?

A. Full replication

B. Partial replication

C. No replication (non-redundant allocation)

D. Hybrid replication

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer C: No replication, or nonredundant allocation.

ADB37 explicitly calls storing each fragment once nonredundant allocation. Full replication places the whole database at every relevant site, while partial replication copies selected fragments to selected sites. A hybrid can mix strategies, but the stem specifies exactly one location and no copies for every fragment, so C is the precise classification.

Why each choice is right or wrong

A - Incorrect: Full replication stores copies throughout the relevant sites rather than one copy per fragment.

B - Incorrect: Partial replication still copies selected fragments and therefore does not mean no copies.

C - Correct: Nonredundant allocation stores each fragment at one site without replicas.

D - Incorrect: A hybrid can combine policies, but the stated universal no-copy arrangement is specifically nonreplicated.

Primary note proof

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once. Copies improve read locality and can improve availability but require storage and update propagation. They can also spread mistakes, so replication does not replace independent backups.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Study-note section ADB37

> Snapshot replication transfers a copy at intervals; incremental replication transfers changes.

Question sources

Advanced Database Systems review questions · Question 47

Question 181 | NS0103

Transparency in distributed databases means that:

A. All data is encrypted and transparent to hackers

B. The distribution complexity is hidden from users who interact as if using a single database

C. Users can see all technical details of how data is distributed

D. All sites must use the same operating system

Hint

Identify exactly which detail is hidden from the user.

Answer B: Hide distribution details behind a single logical database experience.

ADB35 defines transparency through which distribution details are hidden: sites, fragment reconstruction or replica selection/maintenance. Together these can let clients work with a single logical database abstraction. B captures that broad meaning. Transparency is not encryption, disclosure of every technical detail or a requirement that operating systems be identical.

Why each choice is right or wrong

A - Incorrect: Encryption protects content; transparency here concerns hiding distribution complexity from legitimate users.

B - Correct: Distribution details can be hidden so users work with one logical database abstraction.

C - Incorrect: Transparency hides selected implementation details rather than exposing all of them.

D - Incorrect: Operating-system uniformity does not define distribution transparency.

Primary note proof

Study-note section ADB35

> Location transparency hides which site holds a named object. Fragmentation transparency hides the existence and reconstruction of fragments: users query the whole logical relation. Replication transparency hides which copy is used and how copies are maintained. The broad single-logical-database experience may involve all three.

Study-note section ADB33

> Different operating systems are possible but do not define the distinction.

Study-note section ADB24

> Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions.

Question sources

Advanced Database Systems review questions · Question 48

Question 182 | NS0104

Hybrid fragmentation combines which two fragmentation types?

A. Replication and Partitioning

B. Horizontal and Vertical fragmentation

C. Client-Server and Middleware architectures

D. Homogeneous and Heterogeneous distribution

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Horizontal and vertical fragmentation.

ADB36 defines hybrid fragmentation as combining row and column splits in a suitable order. Reconstructing reverses those operations using unions for horizontal pieces and key joins for corresponding vertical pieces. Replication, deployment architectures and DBMS homogeneity are different design dimensions rather than the two fragmentation types combined by hybrid fragmentation.

Why each choice is right or wrong

A - Incorrect: Replication copies data and is not the second fragmentation dimension in the hybrid definition.

B - Correct: Hybrid fragmentation combines horizontal row splits with vertical column splits.

C - Incorrect: Client-server and middleware classify architectures rather than fragmentation operations.

D - Incorrect: Homogeneous/heterogeneous classifies system technology compatibility rather than row/column splitting.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Hybrid fragmentation combines row and column splits in either suitable order.

Study-note section ADB36

> Reconstruct by reversing the operations: key joins for corresponding vertical pieces and unions for horizontal pieces.

Related reading (context, not primary note proof)

Study-note section ADB37

Study-note section ADB34

Study-note section ADB33

Question sources

Advanced Database Systems review questions · Question 49

Question 183 | NS0105

With full replication and synchronous propagation before acknowledgment, which listed disadvantage specifically concerns maintaining copies during an update-heavy workload? Assume usable replicas can serve reads.

A. Data becomes unavailable when any site fails

B. Updates incur propagation and coordination overhead because changes must reach the required copies before acknowledgment.

C. It requires all sites to use the same DBMS software

D. Queries can only be answered from the primary site

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer B: B. Updates incur propagation and coordination overhead because changes must reach the required copies before acknowledgment.

Full replication stores the entire database at each relevant site, so updates create propagation and coordination work. The stated synchronous policy places that work before acknowledgment, adding update latency in the update-heavy scenario. This is the disadvantage described by B. Usable copies can serve reads and improve availability; full replication neither implies identical DBMS software nor restricts queries to one primary site.

Why each choice is right or wrong

A - Incorrect: Usable replicas can keep data available after one site fails.

B - Correct: The stated synchronous policy adds propagation and coordination work before acknowledgment.

C - Incorrect: Full replication is a copy-placement policy, not a requirement to standardize every DBMS.

D - Incorrect: Usable replicas can serve reads in the stated scenario.

Primary note proof

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once. Copies improve read locality and can improve availability but require storage and update propagation. They can also spread mistakes, so replication does not replace independent backups.

Study-note section ADB37

> Eager/synchronous replication coordinates propagation before the specified commit/acknowledgment point, increasing update latency.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Question sources

Advanced Database Systems review questions · Question 50

Question 184 | NS0106

What does ACID stand for in database transactions?

A. Atomicity, Concurrency, Isolation, Durability

B. Atomicity, Consistency, Isolation, Durability

C. Availability, Consistency, Integrity, Durability

D. Atomicity, Correctness, Isolation, Distribution

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Atomicity, Consistency, Isolation, Durability.

ADB06 gives all-or-none effects, preservation of declared rules, controlled interference and persistence after failure as the four ACID properties. B names them correctly. The other options substitute concurrency, availability, integrity, correctness or distribution for the defined terms, changing the acronym rather than describing an equivalent expansion.

Why each choice is right or wrong

A - Incorrect: Concurrency incorrectly replaces Consistency in the acronym.

B - Correct: The expansion exactly matches the four ACID property names.

C - Incorrect: Availability and Integrity replace Atomicity and Isolation, which are the defined terms.

D - Incorrect: Correctness and Distribution do not replace Consistency and Durability in ACID.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 1

Question 185 | NS0107

Which ACID property ensures that either ALL operations of a transaction complete or NONE do?

A. Consistency

B. Isolation

C. Atomicity

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer C: Atomicity.

The requested property concerns whether partial transaction effects can survive. ADB06 assigns all-or-none application to atomicity. A correct transaction also preserves consistency, concurrency needs isolation and completed commits need durability, but those are different guarantees. C is therefore the property matching the explicit ALL or NONE condition.

Why each choice is right or wrong

A - Incorrect: Consistency concerns preservation of integrity rules rather than the indivisible effect boundary itself.

B - Incorrect: Isolation governs concurrent interference rather than all-or-none transaction effects.

C - Correct: Atomicity makes the transaction effects succeed together or be removed together.

D - Incorrect: Durability protects committed effects rather than defining partial-success handling.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 2

Question 186 | NS0108

If A=5000 and B=8000, and 750 is transferred from A to B, what values satisfy the Consistency property?

A. A=4250, B=8750, sum=13000

B. A=4250, B=8000, sum=12250

C. A=5000, B=8750, sum=13750

D. A=5000, B=8000, sum=13000

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer A: A=4250, B=8750, total=13000.

Subtracting 750 from A gives 4250 and adding it to B gives 8750; their total remains 13000. A implements both the specified operation and its total-balance invariant. B and C apply only one side. D preserves the initial total but does not perform the requested transfer, so invariant preservation alone does not make it the successful-transfer result.

Why each choice is right or wrong

A - Correct: It performs the 750 debit and credit and preserves the total of 13000.

B - Incorrect: The debit occurs without its credit, reducing the total to 12250.

C - Incorrect: The credit occurs without its debit, increasing the total to 13750.

D - Incorrect: It preserves the starting total but leaves both accounts unchanged rather than performing the transfer.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Study-note section ADB06

> Unchanged or reverse-transfer balances can also preserve that sum; only the requested debit/credit implements the specified successful transfer.

Study-note section ADB06

> Both balances total 13,000.

Question sources

Advanced Database Systems review questions · Question 3

Question 187 | NS0109

Which ACID property ensures concurrent transactions do not interfere with each other?

A. Atomicity

B. Durability

C. Consistency

D. Isolation

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Isolation.

ADB06 assigns concurrent-interference control to isolation. That is the distinguishing phrase in the stem. Atomicity handles partial effects, durability handles committed persistence, and consistency means correct work preserves declared rules. Different isolation levels provide different detailed guarantees, so this classification does not imply that all implementations offer serializability by default.

Why each choice is right or wrong

A - Incorrect: Atomicity governs partial effects within a transaction rather than concurrent interference.

B - Incorrect: Durability concerns persistence of committed changes after failures.

C - Incorrect: Consistency concerns declared integrity rules under correct business logic.

D - Correct: Isolation controls interference among concurrently executing transactions.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 4

Question 188 | NS0110

Which ACID property guarantees committed changes survive system crashes?

A. Atomicity

B. Consistency

C. Isolation

D. Durability

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer D: Durability.

ADB06 defines durability by persistence of committed effects despite failure. Recovery information can provide this even when changed data pages were buffered at the instant of commit. All-or-none application, invariant preservation and concurrent isolation serve different transaction guarantees, so only durability matches survival of successful commits after a crash.

Why each choice is right or wrong

A - Incorrect: Atomicity concerns all-or-none application rather than the persistence of completed commits.

B - Incorrect: Consistency concerns validity of declared invariants rather than crash survival of commits.

C - Incorrect: Isolation controls concurrent interactions rather than persistence after a crash.

D - Correct: Durability preserves committed effects across system failures.

Primary note proof

Study-note section ADB06

> A transaction is a logical unit of database work, including read-only work. Atomicity means all effects or none; consistency means correct work preserves integrity rules; isolation controls concurrent interference; durability preserves committed effects despite failure. A DBMS cannot automatically make incorrect business logic consistent.

Question sources

Advanced Database Systems review questions · Question 5

Question 189 | NS0111

A transaction that has executed all operations but NOT YET committed is in which state?

A. Active

B. Partially Committed

C. Committed

D. Failed

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Partially committed.

ADB06 places a transaction in Partially committed after its last operation finishes but before durable commitment is assured. The stem states exactly that boundary. Active means operations are still executing; Committed needs successful commit; Failed needs an error. Finishing the operation list alone does not establish the committed state.

Why each choice is right or wrong

A - Incorrect: Active describes ongoing operation execution, not the interval after the last operation finishes.

B - Correct: The last operation finished, but commitment is not yet assured.

C - Incorrect: The stem explicitly says the transaction has not yet committed.

D - Incorrect: No execution or commit failure is reported.

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Question sources

Advanced Database Systems review questions · Question 6

Question 190 | NS0112

In the Aborted state, what happens to all changes made by the transaction?

A. Changes are permanently saved

B. All changes are rolled back restoring the database to its pre-transaction state

C. The transaction waits for resources

D. The transaction restarts automatically

Hint

Track the transaction's guarantee or lifecycle boundary described by the scenario.

Answer B: Its effects have been rolled back; other committed transactions remain.

ADB06 reaches Aborted after rollback. Thus the aborted transaction’s own changes are removed rather than permanently saved. Restoring pre-transaction state must be understood relative to its effects, not erasing unrelated concurrent commits. A restart can be a new attempt, but is not automatic merely because the aborted state was reached.

Why each choice is right or wrong

A - Incorrect: Permanent saving is a committed outcome, not an aborted outcome.

B - Correct: Rollback removes this transaction's effects while retaining other transactions' legitimate committed work.

C - Incorrect: Waiting for a lock describes a blocked executing transaction rather than completed rollback.

D - Incorrect: An aborted transaction may be retried, but automatic restart is not guaranteed.

Primary note proof

Study-note section ADB06

> Active means operations are executing. Partially committed means the last operation finished but durable commitment is not yet assured. Committed means commitment succeeded. A failure leads to Failed, then Aborted after rollback; a transaction may terminate or be restarted as a new attempt. Concurrency control primarily enforces isolation; recovery supports atomicity and durability. Transaction management is the broader subsystem, so both it and recovery management can be legitimate answers to broad responsibility questions.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk.

Question sources

Advanced Database Systems review questions · Question 7

Question 191 | NS0113

Which log record signals successful completion of transaction T?

A. [start_transaction, T]

B. [abort, T]

C. [write_item, T, X, old, new]

D. [commit, T]

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer D: [commit, T].

ADB07 identifies COMMIT as the record of successful transaction completion. START begins work, UPDATE records a data change and ABORT marks failed termination/rollback handling. A data write by itself does not prove the enclosing transaction committed, and an abort record can begin rollback rather than confirm it has completed.

Why each choice is right or wrong

A - Incorrect: The start record indicates beginning work rather than successful completion.

B - Incorrect: Abort records failure/rollback handling instead of successful commitment.

C - Incorrect: An update record describes one changed item, not successful transaction completion.

D - Correct: The commit record signals successful completion of transaction T.

Primary note proof

Study-note section ADB07

> A conceptual log can contain START, UPDATE(T,X,old,new), COMMIT, ABORT and END records. Before-images support undo; after-images support redo. Real systems may use physical or logical records, and need not log every read. COMMIT records successful completion; an ABORT record can mark the start of rollback rather than prove every undo step is complete.

Question sources

Advanced Database Systems review questions · Question 8

Question 192 | NS0114

The Read_item(X) command transfers data in which order?

A. Disk -> Program variable -> Buffer

B. Buffer -> Disk -> Program variable

C. Disk -> Buffer -> Program variable

D. Program variable -> Buffer -> Disk

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: Disk page → buffer → program variable.

ADB07 locates X's page, loads it into a buffer if absent and copies the item from that buffer into the program variable. C gives that conceptual data movement order. A cached page may avoid the physical disk read, but the program still reads through the buffer. The reverse direction describes a write path rather than read_item.

Why each choice is right or wrong

A - Incorrect: The page is buffered before the item is copied to the program variable.

B - Incorrect: A read does not first send the buffer to disk before obtaining the variable value.

C - Correct: The page enters the buffer if needed, then its item is copied into the program variable.

D - Incorrect: Program-variable-to-buffer-to-disk describes writing, not reading into a variable.

Primary note proof

Study-note section ADB07

> read_item(X) locates X's page, brings it into a buffer if absent, and copies X into a program variable. write_item(X) copies the new program value into the buffered page, marking it dirty. Writing that page to disk can occur later according to the buffer policy; write_item is not inherently an immediate disk flush. Insert and delete also modify data; sorting is a query-processing operation rather than a primitive item update.

Study-note section ADB07

> Real systems may use physical or logical records, and need not log every read.

Study-note section ADB07

> Before-images support undo; after-images support redo.

Question sources

Advanced Database Systems review questions · Question 9

Question 193 | NS0115

In a multi-user system, which mechanisms does the DBMS use to manage concurrent access?

A. Only backup systems

B. Locks, timestamps, and scheduling algorithms

C. Physical user separation

D. Only file system permissions

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer B: Locks, timestamp rules and transaction scheduling.

ADB11 defines compatible shared/exclusive locks and ADB15 orders conflicting operations with timestamp tests. These are actual concurrency-control mechanisms that constrain transaction schedules. Backups provide copies for recovery, and file permissions determine access rather than safe interleavings. Physical separation of people also does not coordinate their concurrent database operations.

Why each choice is right or wrong

A - Incorrect: Backups support restoration rather than scheduling safe concurrent reads and writes.

B - Correct: Locks and timestamp rules control conflicting accesses and permissible schedules.

C - Incorrect: Separating users physically does not coordinate their concurrent operations on shared database items.

D - Incorrect: File permissions alone do not enforce isolation among authorized concurrent transactions.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts.

Study-note section ADB09

> Two operations conflict when they belong to different transactions, access the same item and at least one writes.

Related reading (context, not primary note proof)

Study-note section ADB15

Study-note section ADB24

Question sources

Advanced Database Systems review questions · Question 10

Question 194 | NS0116

The Lost Update problem is also called:

A. Read-Read Problem

B. Write-Write Problem

C. Read-Write Problem

D. Phantom Read Problem

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Write-write problem, as shorthand for a lost-update overwrite.

ADB08 illustrates both transactions reading 500, one writing 700 and the other later overwriting it with stale 600. The conflict is between their writes, which explains the offered write-write shorthand. This is not a claim that every pair of writes loses an update: the stale calculation and overwrite are essential. Read/read and phantom behavior describe different situations.

Why each choice is right or wrong

A - Incorrect: Two reads alone do not overwrite and lose an update.

B - Correct: The stale write overwrites another write, motivating this write-write shorthand.

C - Incorrect: The defining loss is the later stale overwrite, not merely an arbitrary read/write pair.

D - Incorrect: A phantom changes a predicate query's qualifying row set rather than overwriting one balance.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Question sources

Advanced Database Systems review questions · Question 11

Question 195 | NS0117

A Dirty Read occurs when:

A. A transaction reads only committed data

B. A transaction reads data modified by another uncommitted transaction

C. Two transactions read the same record simultaneously

D. A transaction writes invalid format data

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: Reading another transaction’s uncommitted data.

ADB08 defines dirty read through another transaction’s uncommitted write. The value can disappear if its writer aborts, leaving dependent work based on invalid data. Reading committed data or two readers accessing one item is not inherently dirty. Invalid-format writes concern data validity rather than whether the read-from transaction has committed.

Why each choice is right or wrong

A - Incorrect: Reading only committed values avoids the defining uncommitted-writer dependency.

B - Correct: The read consumes another transaction's modification before that writer commits.

C - Incorrect: Multiple readers alone do not create an uncommitted data dependency.

D - Incorrect: Invalid format concerns validation or integrity rather than the definition of a dirty read.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Question sources

Advanced Database Systems review questions · Question 12

Question 196 | NS0118

Which trace demonstrates an incorrect-summary anomaly, meaning an aggregate that mixes mutually inconsistent versions? Treat nonrepeatable read as a separate anomaly.

A. A transaction reads data after another commits

B. Initially A=B=100. T2 transfers 10 from A to B while T1 reads old A=100 and new B=110, reporting 210 rather than the invariant total 200.

C. Two transactions delete the same record

D. A transaction reads its own uncommitted data

Hint

Trace which transaction wrote each value and whether an overwrite or mixed-version read occurs.

Answer B: B. Initially A=B=100. T2 transfers 10 from A to B while T1 reads old A=100 and new B=110, reporting 210 rather than the invariant total 200.

The transfer subtracts and adds the same amount, preserving a total of 200. T1 instead adds an old A balance to a new B balance and obtains 210, combining versions that do not form a consistent summary. That is the incorrect-summary anomaly. Reading committed data, deleting a record or reading one's own uncommitted changes alone does not establish this aggregate defect.

Why each choice is right or wrong

A - Incorrect: A read after another commit is not by itself an inconsistent aggregate.

B - Correct: 100+110=210 mixes versions and violates the preserved total 200.

C - Incorrect: Concurrent deletions alone do not show this summary anomaly.

D - Incorrect: Reading one's own changes is not the stated mixed-version aggregate.

Primary note proof

Study-note section ADB08

> Lost update: T1 and T2 both read 500; T2 adds 200 and writes 700; T1 adds 100 to its stale value and writes 600. Correct serial execution yields 800. Dirty read: T2 reads a value written by uncommitted T1; if T1 aborts, dependent work must be reconsidered. Reading one's own changes is not a dirty read.

Study-note section ADB08

> Incorrect summary mixes mutually inconsistent versions: a transfer moves 10 from A=100 to B=100; a sum reading old A=100 and new B=110 gets 210 instead of the invariant 200.

Study-note section ADB08

> A single committed update from B=20 to B=50 does not alone prove an invalid summary of 90.

Question sources

Advanced Database Systems review questions · Question 13

Question 197 | NS0119

A Shared Lock allows:

A. Only one transaction to read an item

B. Multiple transactions to read simultaneously but not write

C. One transaction to read and write exclusively

D. Multiple transactions to write simultaneously

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer B: Compatible readers, with no writing under a shared lock.

ADB11 permits multiple shared locks on one item, each for reading. Writing requires a suitable exclusive lock, which conflicts with other shared holders. Thus B describes shared-lock behavior in the basic lock-based model. Exclusive read/write ownership and multiple simultaneous writers are not shared locking; multiversion snapshot rules are a separate model.

Why each choice is right or wrong

A - Incorrect: More than one transaction may hold compatible shared read locks on the same item.

B - Correct: Shared locks permit multiple readers but do not authorize writes under that lock mode.

C - Incorrect: Sole read/write ownership describes an exclusive lock.

D - Incorrect: Shared locks do not allow multiple transactions to write simultaneously.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB11

> One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Study-note section ADB11

> No double-locking is not a prohibition on shared readers.

Question sources

Advanced Database Systems review questions · Question 14

Question 198 | NS0120

An Exclusive Lock allows:

A. Multiple transactions to read concurrently

B. Multiple transactions to write simultaneously

C. Only one transaction to both read and write, blocking all other access

D. Multiple readers and one writer simultaneously

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer C: The exclusive holder may read and write; conflicting access to that item is blocked.

ADB11 permits an exclusive lock holder to read and write and makes that item's X lock conflict with another S or X. C is correct in this basic model. Blocking all other access means conflicting access to the same item, not the entire database; locks on other items remain independent and multiversion systems can have additional snapshot-reader behavior.

Why each choice is right or wrong

A - Incorrect: Other transactions' shared reads conflict with an exclusive lock on the same item in this model.

B - Incorrect: Two exclusive writers cannot hold compatible locks on the same item.

C - Correct: One holder can read/write the locked item while incompatible same-item accesses wait or abort.

D - Incorrect: Concurrent same-item shared readers conflict with this exclusive writer under the basic rules.

Primary note proof

Study-note section ADB11

> A shared (S) lock permits reading; multiple S locks on the same item can coexist. An exclusive (X) lock permits its holder to read and write and conflicts with another S or X on that item. Locks on different items do not conflict merely because one is exclusive. These are the basic lock-based rules; multiversion systems can permit additional snapshot-reader behavior.

Study-note section ADB11

> One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy.

Study-note section ADB11

> S→X is an upgrade, possible only after conflicting other holders leave.

Question sources

Advanced Database Systems review questions · Question 15

Question 199 | NS0121

In Two-Phase Locking (2PL), during the Growing Phase a transaction:

A. Releases locks only

B. Acquires locks without releasing any

C. Acquires and releases locks freely

D. Makes no lock requests

Hint

Track the first lock release and the transaction-completion boundary.

Answer B: Acquire or upgrade locks without releasing them.

ADB12 defines growing as the acquisition/upgrade phase and shrinking as the release/downgrade phase. Once any lock is released, no further acquisition is allowed. B therefore matches growing. Freely mixing acquisition and release would abandon the two-phase discipline that guarantees conflict serializability for well-formed transactions.

Why each choice is right or wrong

A - Incorrect: Releasing locks belongs to shrinking rather than growing.

B - Correct: Growing acquires or upgrades locks before any release occurs.

C - Incorrect: Freely acquiring after release violates the defining 2PL phase restriction.

D - Incorrect: Growing is precisely the phase in which lock requests may be made.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB12

> Conservative/static 2PL obtains all needed locks before execution and does not begin while only a subset is available, preventing hold-and-wait deadlock.

Question sources

Advanced Database Systems review questions · Question 16

Question 200 | NS0122

Which concurrency control technique uses start-times to order transaction execution?

A. Locking

B. Optimistic Concurrency Control

C. Timestamp-based Concurrency Control

D. Shadow Paging

Hint

Compare the transaction timestamp with the item's recorded read and write timestamps.

Answer C: Timestamp-based concurrency control.

ADB15 assigns each transaction a timestamp and checks reads and writes against item read/write timestamps. These checks preserve timestamp order and can abort stale operations. Locking instead uses ownership compatibility, optimistic control validates private work and shadow paging concerns recovery page maps. Only C names the start-time ordering technique described.

Why each choice is right or wrong

A - Incorrect: Locking uses lock compatibility and protocols rather than being defined by transaction start-time ordering.

B - Incorrect: Optimistic control validates private read/write sets rather than applying these timestamp-order tests.

C - Correct: Timestamp-based control orders valid conflicting operations using transaction timestamps.

D - Incorrect: Shadow paging preserves committed page maps for recovery rather than ordering concurrent operations.

Primary note proof

Study-note section ADB15

> Assign each transaction TS(T). For item X track read_TS(X), the largest timestamp of a successful reader, and write_TS(X), that of its latest successful writer. A read by T aborts if TS(T)<write_TS(X); otherwise it succeeds and updates read_TS. A write aborts if TS(T)<read_TS(X) or TS(T)<write_TS(X); otherwise it succeeds and sets write_TS. These tests preserve timestamp order.

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries.

Study-note section ADB29

> Shadow/copy-on-write update writes a new location while preserving the committed version.

Related reading (context, not primary note proof)

Study-note section ADB11

Question sources

Advanced Database Systems review questions · Question 17

Question 201 | NS0123

Optimistic Concurrency Control (OCC) is best suited for:

A. High-conflict write-heavy workloads

B. Read-heavy workloads where conflicts are rare

C. Single-user database systems

D. Distributed transactions only

Hint

Consider conflicts, private work and validation costs, or lock-management overhead.

Answer B: Read-heavy, low-conflict workloads.

ADB16 computes privately, validates and publishes, retrying failed validation. Low-conflict workloads let most work succeed; high write contention wastes computation on aborts. B states that criterion explicitly. Single-user systems do not need intertransaction conflict control, and OCC is not restricted to distributed transactions. Actual conflict frequency matters more than the read-heavy label alone.

Why each choice is right or wrong

A - Incorrect: High conflicts cause validation failures and repeated wasted work.

B - Correct: Rare conflicts allow optimistic private work to validate successfully with few retries.

C - Incorrect: A single-user setting lacks the concurrent conflict problem motivating this choice.

D - Incorrect: Optimistic control is not limited to distributed transaction environments.

Primary note proof

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries. It works well when conflicts are uncommon, but high contention wastes work on aborts. Multiversion control keeps versions so a reader can use an appropriate snapshot. Snapshot isolation is not automatically serializable: concurrent writes to different rows can violate a shared invariant.

Question sources

Advanced Database Systems review questions · Question 18

Question 202 | NS0124

Deadlock occurs when:

A. One transaction waits for a resource held by no one

B. Two or more transactions each wait for resources held by the other in a circular chain

C. A transaction commits too slowly

D. Multiple transactions read identical data

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: A circular resource-wait chain.

ADB13 requires a waiting cycle: participants hold resources needed by others and cannot make progress without a victim release or another intervention. Waiting for an unheld resource does not state such a cycle. Slow commit and simultaneous reads can occur without deadlock; compatible shared readers in particular need not block one another.

Why each choice is right or wrong

A - Incorrect: A resource held by no one does not create the stated circular hold-and-wait dependency.

B - Correct: Each participant waits for another holder in a cycle, preventing independent progress.

C - Incorrect: A slow commit may still progress and need not form a waiting cycle.

D - Incorrect: Compatible shared reads of the same data do not inherently deadlock.

Primary note proof

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

> A timeout aborts long waits but may mistake slow progress for deadlock.

Study-note section ADB11

> No double-locking is not a prohibition on shared readers.

Question sources

Advanced Database Systems review questions · Question 19

Question 203 | NS0125

In the Wait-Die protocol, when a YOUNGER transaction requests a resource held by an OLDER transaction:

A. The younger transaction dies (aborts)

B. The older transaction dies

C. The younger transaction waits

D. Both transactions abort

Hint

Identify requester and holder ages before applying the protocol's action.

Answer A: The younger requester aborts.

The requester is younger and the holder older. ADB14 makes that requester die in wait-die; only an older requester may wait for a younger holder. On retry the aborted transaction retains its original timestamp. Wounding a holder belongs to a different protocol, and both participants need not be aborted.

Why each choice is right or wrong

A - Correct: Wait-die aborts the younger requester when the needed resource is held by an older transaction.

B - Incorrect: The older holder keeps its resource; this protocol does not kill it for a younger requester.

C - Incorrect: Waiting is allowed for an older requester facing a younger holder, not this case.

D - Incorrect: The policy aborts the younger requester, not both participants.

Primary note proof

Study-note section ADB14

> A smaller timestamp means an older transaction. In wait-die, an older requester may wait for a younger holder; a younger requester needing an older holder aborts itself. In wound-wait, an older requester aborts the younger holder; a younger requester waits for the older holder. Always identify requester versus holder before applying the names.

Study-note section ADB14

> Retain the original timestamp when restarting an aborted transaction so it gains relative priority as new transactions arrive.

Study-note section ADB14

> In wait-die, an older transaction may wait for a younger one; a younger transaction requesting a resource held by an older one aborts.

Question sources

Advanced Database Systems review questions · Question 20

Question 204 | NS0126

In the Wound-Wait protocol, when an OLDER transaction requests a resource held by a YOUNGER transaction:

A. The older transaction waits

B. The younger is wounded (aborted), allowing the older to proceed

C. Both transactions are aborted

D. A new transaction takes over

Hint

Identify requester and holder ages before applying the protocol's action.

Answer B: Abort the younger holder so the older requester can proceed.

ADB14 says an older requester wounds a younger holder, while a younger requester waits. B is therefore the rule for the given ages. The protocol uses priority to prevent cycles rather than arbitrarily aborting both or handing the work to a new transaction. Any retry should preserve the original timestamp to reduce starvation.

Why each choice is right or wrong

A - Incorrect: An older requester wounds the younger holder rather than waiting under this policy.

B - Correct: The younger holder is aborted, releasing the resource for the older requester.

C - Incorrect: Aborting both is unnecessary and is not the wound-wait rule.

D - Incorrect: No replacement transaction takes over; the older requester continues after the conflicting holder releases.

Primary note proof

Study-note section ADB14

> A smaller timestamp means an older transaction. In wait-die, an older requester may wait for a younger holder; a younger requester needing an older holder aborts itself. In wound-wait, an older requester aborts the younger holder; a younger requester waits for the older holder. Always identify requester versus holder before applying the names.

Study-note section ADB14

> Retain the original timestamp when restarting an aborted transaction so it gains relative priority as new transactions arrive.

Study-note section ADB14

> In wait-die, an older transaction may wait for a younger one; a younger transaction requesting a resource held by an older one aborts.

Question sources

Advanced Database Systems review questions · Question 21

Question 205 | NS0127

Lock Timeout as a deadlock solution works by:

A. Building and analyzing wait-for graphs

B. Aborting any transaction that waits longer than a specified time limit

C. Locking all items before a transaction starts

D. Assigning age-based priorities

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: Abort a transaction whose wait exceeds the configured limit.

ADB13 defines timeout by excessive wait duration and warns it can mistake slow progress for deadlock. B matches that mechanism. Wait-for graphs detect cycles, conservative locking obtains all locks before execution and age priorities define wait-die/wound-wait decisions. Those are distinct policies rather than implementations of a time limit.

Why each choice is right or wrong

A - Incorrect: Graph analysis detects cycles rather than deciding from elapsed lock-wait time.

B - Correct: Timeout aborts a waiting transaction after its configured wait limit is exceeded.

C - Incorrect: Preacquiring all locks is conservative locking rather than timeout handling.

D - Incorrect: Age-based priorities define other prevention policies, not an elapsed-wait threshold.

Primary note proof

Study-note section ADB13

> Deadlock requires a waiting cycle: T1 holds X and wants Y; T2 holds Y and wants X. In a wait-for graph, an edge Ti→Tj means Ti waits for a lock held by Tj. A cycle indicates deadlock under the ordinary lock-wait model. Abort a suitable victim, undo its effects and release its locks; repeat if other cycles remain. Aborting every deadlocked transaction also breaks cycles but wastes more work.

Study-note section ADB13

> A timeout aborts long waits but may mistake slow progress for deadlock.

Study-note section ADB12

> Conservative/static 2PL obtains all needed locks before execution and does not begin while only a subset is available, preventing hold-and-wait deadlock.

Related reading (context, not primary note proof)

Study-note section ADB14

Question sources

Advanced Database Systems review questions · Question 22

Question 206 | NS0128

A schedule is serializable if:

A. All transactions execute one after another with no overlap

B. Its concurrent execution produces the same result as some serial execution of those transactions

C. All transactions commit at the same time

D. No two transactions access the same table

Hint

Distinguish transaction ordering, conflict conditions and full execution equivalence.

Answer B: Equivalent behavior to some serial execution, even if interleaved.

ADB09 defines a serializable schedule as equivalent to a serial order while permitting interleaving. B matches when same result means the required full transaction equivalence, not an accidental equality of one final value. A describes a serial schedule, which is a special case but not the general definition. Commit simultaneity and disjoint tables are not required.

Why each choice is right or wrong

A - Incorrect: This defines serial execution, a stricter special case that omits permitted serializable interleavings.

B - Correct: Serializable behavior is equivalent to some serial ordering; coincidental equality of one value alone is insufficient.

C - Incorrect: Simultaneous commits do not establish the required transaction-execution equivalence.

D - Incorrect: Serializable transactions may access the same tables if their operations are correctly ordered.

Primary note proof

Study-note section ADB09

> A serial schedule executes each transaction's operations consecutively. A serializable schedule is equivalent to some serial order; it may interleave transactions. Equality of one final value is insufficient evidence because different executions can accidentally coincide. Two operations conflict when they belong to different transactions, access the same item and at least one writes. Read/read pairs do not conflict.

Question sources

Advanced Database Systems review questions · Question 23

Question 207 | NS0129

Lock Upgrade refers to:

A. Releasing an exclusive lock to a shared lock

B. Converting a shared (read) lock to an exclusive (write) lock

C. Removing all locks at commit

D. Granting a lock to multiple transactions

Hint

Compare access permissions and same-item compatibility before granting a lock request.

Answer B: Convert Shared to Exclusive.

ADB11 identifies S→X as an upgrade, possible only after incompatible other holders leave. It strengthens an existing read lock to permit writing. X→S is a downgrade, releasing all locks is transaction cleanup and granting compatible readers is shared acquisition. Only B names the requested conversion of a held lock.

Why each choice is right or wrong

A - Incorrect: Exclusive-to-shared is a downgrade rather than an upgrade.

B - Correct: Shared-to-exclusive strengthens the lock to permit writing when compatibility allows.

C - Incorrect: Removing locks is release rather than conversion to a stronger mode.

D - Incorrect: Granting compatible shared readers is not the upgrade of one existing lock.

Primary note proof

Study-note section ADB11

> A transaction must own the appropriate lock before access and can release only its own locks. S→X is an upgrade, possible only after conflicting other holders leave. X→S is a downgrade. The lock manager decides requests; the lock table stores granted locks and waiting requests. One transaction may request a lock on an already locked item: it is granted if compatible, otherwise waits or aborts under the policy. No double-locking is not a prohibition on shared readers.

Question sources

Advanced Database Systems review questions · Question 24

Question 208 | NS0130

Starvation in concurrency control means:

A. A transaction holding too many locks

B. A transaction is perpetually delayed and never executes due to other transactions always taking priority

C. The database runs out of storage space

D. A transaction commits before completing all operations

Hint

Distinguish a waiting cycle, a long wait and indefinite unfair postponement.

Answer B: Indefinite postponement caused by repeated priority to other work.

ADB13 defines starvation as indefinite postponement while other transactions progress, without requiring a deadlock cycle. B states that repeated bypass precisely. Holding many locks can create contention but does not itself prove starvation. Storage exhaustion and premature commit are different failures, so neither supplies the scheduling-fairness definition.

Why each choice is right or wrong

A - Incorrect: Holding many locks describes resource use rather than being perpetually denied execution.

B - Correct: Repeated priority to others can indefinitely postpone the transaction while other work continues.

C - Incorrect: Running out of storage is a capacity failure rather than the defined scheduling starvation.

D - Incorrect: Premature commit concerns transaction correctness rather than indefinite postponement.

Primary note proof

Study-note section ADB13

> Fixed resource ordering and conservative 2PL prevent cycles. A no-wait policy aborts immediately on an incompatible lock request and needs no age timestamp. A timeout aborts long waits but may mistake slow progress for deadlock. Starvation is indefinite postponement while other work proceeds, without a required cycle. Fair queues, sensible victim selection and retaining original transaction timestamps on retry reduce starvation; deadlock freedom alone does not prove fairness.

Question sources

Advanced Database Systems review questions · Question 25

Question 209 | NS0131

The first step in SQL query processing is:

A. Optimization

B. Evaluation

C. Parsing and Translation

D. Result Delivery

Hint

Separate query representation, strategy selection and execution.

Answer C: Parsing and translation.

ADB17 first checks syntax and semantics and constructs a logical query representation, before optimization and evaluation. C groups those initial activities. Result delivery follows execution, while optimization needs an understood query to compare valid plans. The phase grouping is conventional, but the ordering dependency is explicit in the complete note.

Why each choice is right or wrong

A - Incorrect: Optimization follows understanding and representing the query rather than preceding it.

B - Incorrect: Evaluation executes a selected plan and is not the first processing activity.

C - Correct: Parsing/translation establishes query meaning and logical representation before optimization.

D - Incorrect: Results can be delivered only after the required evaluation produces them.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> Query processing is the whole activity; selecting a suitable efficient strategy is optimization.

Study-note section ADB17

> DDL is not a mandatory stage before each SELECT, so authentication→DDL→DML permutations do not describe the processing sequence.

Question sources

Advanced Database Systems review questions · Question 26

Question 210 | NS0132

SQL queries are internally translated into:

A. Machine code

B. Relational algebra

C. Python scripts

D. XML documents

Hint

Separate query representation, strategy selection and execution.

Answer B: Relational algebra.

ADB17 translates a query into relational algebra or a logical query tree for equivalent rewrites and plan selection. B names that logical representation. A particular engine may later compile execution code, but machine code, Python scripts and XML documents are not the general logical translation target taught here. Logical meaning precedes implementation-specific execution choices.

Why each choice is right or wrong

A - Incorrect: Machine code may be an implementation detail but is not the stated logical translation target.

B - Correct: Relational algebra represents the logical operations used for rewriting and planning.

C - Incorrect: Python scripts are not the general internal representation specified here.

D - Incorrect: XML documents are not the defined general relational query representation.

Primary note proof

Study-note section ADB17

> Parsing checks SQL grammar; semantic analysis resolves names, types and access permissions. Translation constructs a logical representation, typically relational algebra or a logical query tree. Optimization chooses equivalent operations, join order and physical access methods. Evaluation executes that plan and returns results. Not every implementation exposes exactly the same phase boundaries.

Study-note section ADB17

> A physical execution plan adds scan/index choices and algorithms.

Study-note section ADB17

> A query tree has base relations at leaves and operators at internal nodes; a query graph emphasizes relations and predicates.

Question sources

Advanced Database Systems review questions · Question 27

Question 211 | NS0133

What does query optimization aim to minimize?

A. The number of tables referenced

B. The overall cost (time and resources) of executing the query

C. The number of join operations

D. The character length of the SQL query

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: Overall execution cost under the chosen resource model.

ADB19 measures candidate plans using I/O, CPU, memory, temporary storage and network traffic. The aim is a suitable low-cost equivalent plan, not syntactic brevity or indiscriminate removal of joins and tables. Required tables and semantics must be preserved; even fewer operations do not necessarily imply lower runtime or resource cost.

Why each choice is right or wrong

A - Incorrect: Required tables cannot be discarded merely to reduce the table count.

B - Correct: Optimization compares equivalent plans using estimated time and resource costs.

C - Incorrect: Join count alone ignores access paths, cardinalities and the cost of each join.

D - Incorrect: Short SQL text does not necessarily produce a cheaper execution plan.

Primary note proof

Study-note section ADB19

> A cost model estimates resources for a candidate plan: page I/O, CPU, memory, temporary storage and network traffic. Catalog statistics include row/page counts, distinct values and distributions. Selection with fraction s predicts s×N rows. For non-NULL equijoin keys, a common estimate is $|R|\times|S|/\max(V(R,k),V(S,k))$, assuming uniform values and containment of the smaller distinct-value domain within the larger. Adjust counts for NULLs; if either non-NULL input is empty, the result is zero. Skew, correlations or incomplete domain overlap can invalidate this estimate; known disjoint domains give zero matches.

Study-note section ADB19

> Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans.

Study-note section ADB19

> For a simple rotating-disk model, cost = seeks×seek_time + transferred_blocks×block_transfer_time.

Question sources

Advanced Database Systems review questions · Question 28

Question 212 | NS0134

The dominant cost factor in query execution is typically:

A. CPU processing time

B. Network transmission time

C. Disk I/O time (seek + transfer)

D. Memory allocation time

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer C: Disk I/O in the conventional local-disk query-cost model.

ADB19 says disk I/O often dominates textbook local-disk examples, matching the question's typical teaching model. C includes both seek and transfer work. This is a contextual tendency, not a universal claim: CPU can dominate in-memory work and communication can dominate distributed plans. The answer should retain that local-disk qualification rather than extend it to every system.

Why each choice is right or wrong

A - Incorrect: CPU can dominate in-memory work but is not the typical local-disk-model answer.

B - Incorrect: Network cost can dominate distributed plans rather than the conventional local-disk model.

C - Correct: Seek and transfer I/O often dominate the conventional disk-based cost model.

D - Incorrect: Memory allocation is not the dominant term specified for the textbook local-disk model.

Primary note proof

Study-note section ADB19

> For a simple rotating-disk model, cost = seeks×seek_time + transferred_blocks×block_transfer_time. Five seeks at 5 ms plus ten blocks at 1 ms cost 35 ms. Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans. State the assumed cost objective before ranking strategies.

Question sources

Advanced Database Systems review questions · Question 29

Question 213 | NS0135

If a query requires 5 seeks (tS=5ms each) and transfers 10 blocks (tT=1ms each), what is the disk I/O cost?

A. 50ms

B. 35ms

C. 15ms

D. 25ms

Hint

Identify the resource objective, counts and unit costs before comparing execution strategies.

Answer B: 35 ms.

ADB19 adds seeks times seek duration to blocks times per-block transfer duration. Five seeks cost 5×5=25 ms and ten block transfers cost 10×1=10 ms. Their sum is 35 ms. The terms must be added, not multiplied together or considered separately; this is the supplied simple disk model rather than a measured universal query runtime.

Why each choice is right or wrong

A - Incorrect: Fifty does not equal the sum of 25 ms seeking and 10 ms transfer.

B - Correct: Five times five plus ten times one gives 35 ms.

C - Incorrect: Fifteen omits the correct five-seek multiplication.

D - Incorrect: Twenty-five counts only seek time and omits ten milliseconds of transfer.

Primary note proof

Study-note section ADB19

> For a simple rotating-disk model, cost = seeks×seek_time + transferred_blocks×block_transfer_time. Five seeks at 5 ms plus ten blocks at 1 ms cost 35 ms. Disk I/O often dominates textbook local-disk examples; CPU can dominate in-memory work and network can dominate distributed plans. State the assumed cost objective before ranking strategies.

Question sources

Advanced Database Systems review questions · Question 30

Question 214 | NS0136

Database recovery restores the database after failure to:

A. Its state at database creation

B. A consistent and correct state, undoing incomplete transactions and reapplying committed ones

C. The state at the last user login

D. An empty state for fresh start

Hint

Identify what failed, what durable resources survive and which restoration action is required.

Answer B: A valid recovered state preserving durable commits and removing incomplete effects.

ADB26 defines restart recovery by preserving durable commits and removing incomplete effects after failure. B matches the simple undo/redo teaching model. The detailed work depends on buffer policy and algorithm; not every committed update needs physical redo if already installed. Recovery does not reset the database to creation, last login or an empty replacement state.

Why each choice is right or wrong

A - Incorrect: Returning to creation would discard legitimate later durable commits.

B - Correct: Recovery preserves committed work and removes incomplete effects to restore valid state.

C - Incorrect: User login time is not the transaction recovery boundary.

D - Incorrect: Emptying the database would destroy committed data rather than recover it.

Primary note proof

Study-note section ADB26

> Transaction failures include invalid operations, constraint violations and concurrency-control aborts. Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure. A system crash loses volatile memory while disks may survive. Media failure damages persistent storage and normally requires restoring a backup, then replaying available archived logs. Human mistakes may require point-in-time restoration before the bad committed operation.

Study-note section ADB26

> Restart recovery restores a valid database after a crash while preserving durable commits and removing incomplete effects.

Study-note section ADB27

> In a simple undo/redo teaching model, undo losers and redo winners.

Question sources

Advanced Database Systems review questions · Question 31

Question 215 | NS0137

Which type of failure occurs when DBMS software bugs cause unexpected behavior?

A. Hardware Failure

B. Natural Disaster

C. Software Error

D. Network Failure

Hint

Classify the stated cause rather than inferring it from the observed symptom.

Answer C: Software error.

The stem directly identifies DBMS software bugs as the cause. That is a software error rather than failed hardware, a natural event or broken communication. ADB26 separates transaction/system/media failure concerns so the recovery resource can be chosen appropriately; an observed crash alone would not identify its cause, but this question explicitly supplies it.

Why each choice is right or wrong

A - Incorrect: The stated cause is faulty software behavior rather than a malfunctioning hardware component.

B - Incorrect: No natural event is described; the prompt names a software bug.

C - Correct: Unexpected behavior caused by DBMS bugs is a software error.

D - Incorrect: No failed network connection is given as the cause of the behavior.

Primary note proof

Study-note section ADB-S05

> A software failure originates in erroneous program behavior, such as a logical programming defect, invalid operation or operating-system fault. A hardware failure originates in malfunctioning physical equipment; loss of disk contents is media failure. An accidental user deletion is human error even if the software executes the requested command correctly. Physical catastrophes such as fire can also cause failure. The observed symptom of a crash does not alone identify which cause occurred.

Study-note section ADB26

> Generic recovery-manager terminology should not be mistaken for an exact MySQL component specification.

Study-note section ADB26

Enforcing concurrency can deliberately abort work; it does not prevent every transaction failure.

Question sources

Advanced Database Systems review questions · Question 32

Question 216 | NS0138

What does WAL require? In B, 'corresponding data changes' specifically means durable output of the changed data page; volatile buffer changes may occur earlier.

A. Data written to disk before the log record

B. Log records written to stable storage BEFORE corresponding data changes

C. Logs written only after transaction commits

D. Checkpoints before every write operation

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer B: B. Log records written to stable storage BEFORE corresponding data changes

The update log needed for recovery must be stable before the changed data page becomes persistent. B states that order under the question's explicit durable-write meaning. The buffer can be modified earlier. Data-before-log reverses the rule, waiting until after commit is not the required logging policy and checkpointing before every write is not a WAL requirement.

Why each choice is right or wrong

A - Incorrect: The durable log must precede the changed page, not follow it.

B - Correct: With the stated scope, stable logs precede corresponding durable page output.

C - Incorrect: Update logging is not deferred until after commit.

D - Incorrect: WAL does not require a checkpoint for every write.

Primary note proof

Study-note section ADB27

> Write-ahead logging requires the update log needed for recovery to reach stable storage before the corresponding changed data page reaches disk. To acknowledge a durable commit, force the transaction's required log records, including its commit record, to stable storage. A buffer can be changed before that log flush; WAL orders durable writes, not every CPU assignment.

Study-note section ADB27

> Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not.

Study-note section ADB27

> Redo installs logged after-images when necessary, usually forward.

Question sources

Advanced Database Systems review questions · Question 33

Question 217 | NS0139

The Redo operation during recovery:

A. Reverses incomplete transactions

B. Re-applies committed transactions whose changes were not yet persisted to disk

C. Creates new checkpoints

D. Locks data items during the recovery process

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer B: Reapply missing logged committed effects in the simple undo/redo model.

ADB27 uses redo to install logged after-images when necessary. B describes preserving committed work missing from disk pages in the simple teaching model. It does not mean re-running application side effects or blindly adding an amount twice. ARIES has a broader repeat-history phase before undo, but that does not make reversal, checkpoint creation or lock acquisition the definition of redo.

Why each choice is right or wrong

A - Incorrect: Reversing incomplete changes is undo rather than redo.

B - Correct: Redo installs required logged effects missing from data pages, preserving committed work in the simple model.

C - Incorrect: Checkpoint creation records recovery state rather than reapplying update after-images.

D - Incorrect: Locking may coordinate access but is not the operation of logged redo.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB30

> In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Related reading (context, not primary note proof)

Study-note section ADB11

Question sources

Advanced Database Systems review questions · Question 34

Question 218 | NS0140

The Undo operation during recovery:

A. Re-applies committed transactions

B. Reverses changes of incomplete/aborted transactions using before-images from the log

C. Writes the WAL to stable storage

D. Creates shadow page copies

Hint

Track durable-write ordering and distinguish reversing incomplete work from installing logged effects.

Answer B: Restore before-images of incomplete or aborted work.

ADB27 restores before-images when undoing incomplete effects, normally following that transaction's updates backward. B states both the affected work and the supporting log information. Applying committed after-images is redo, forcing logs is a WAL durability action, and copying pages is part of a different update/recovery approach. None is the reversal operation itself.

Why each choice is right or wrong

A - Incorrect: Reapplying committed effects is the redo purpose in the simple model.

B - Correct: Undo restores logged old values to remove incomplete or aborted transaction effects.

C - Incorrect: Forcing the log supports durable ordering but does not itself reverse data changes.

D - Incorrect: Shadow-page copies preserve versions instead of defining logged before-image undo.

Primary note proof

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward. Redo installs logged after-images when necessary, usually forward. Setting X to its logged after-value is idempotent; blindly replaying X=X+10 twice is not. In a simple undo/redo teaching model, undo losers and redo winners. ARIES has a different redo phase that repeats history before undoing losers.

Study-note section ADB29

It is an alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Study-note section ADB29

Shadow/copy-on-write update writes a new location while preserving the committed version.

Question sources

Advanced Database Systems review questions · Question 35

Question 219 | NS0141

Shadow Paging recovery technique works by:

A. Using timestamps to sequence recovery

B. Writing changes to shadow (copy) pages, keeping originals intact, then swapping on commit

C. Using transaction logs for all rollbacks

D. Taking checkpoints every transaction

Hint

Separate update timing from update location and follow the commit boundary.

Answer B: Copy changed pages and atomically install the new durable map on commit.

ADB29 keeps the committed page map unchanged while modifications use new locations and a modified map. Commit durably installs the new root/map pointer after required pages are durable. B captures that preservation-and-switch mechanism. Timestamp ordering, mandatory per-transaction checkpoints and universal log-based rollback are not the defining shadow-paging steps.

Why each choice is right or wrong

A - Incorrect: Timestamp ordering controls concurrent operations rather than defining shadow page-map replacement.

B - Correct: Copy-on-write preserves originals and commit safely installs the new page map.

C - Incorrect: Shadow paging is an alternative to conventional log-based recovery, not mandatory logged rollback for all changes.

D - Incorrect: A checkpoint every transaction is not the mechanism that preserves the old page map.

Primary note proof

Study-note section ADB29

In-place update overwrites the original location. Shadow/copy-on-write update writes a new location while preserving the committed version. Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable. Before commit, a crash uses the old map. It is an

> alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Study-note section ADB30

> Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history.

Related reading (context, not primary note proof)

Study-note section ADB15

Question sources

Advanced Database Systems review questions · Question 36

Question 220 | NS0142

A checkpoint in database recovery helps because:

A. It stops all transactions

B. Recovery can start from the checkpoint state rather than processing the entire log from the beginning

C. It creates a full database backup copy

D. It grants exclusive locks to all active transactions

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer B: A checkpoint supplies recovery state that avoids scanning all historical logs from the beginning.

ADB30 uses durable checkpoint state to avoid processing all old history. B describes that benefit if start means entering recovery analysis, not assuming every earlier update is irrelevant. Active transactions can span a checkpoint and fuzzy-checkpoint redo may require earlier log positions. A checkpoint is not a full backup, universal stop or lock grant.

Why each choice is right or wrong

A - Incorrect: Fuzzy checkpoints permit ongoing transactions, so stopping all work is not required.

B - Correct: Recorded recovery state provides a bounded starting point, while earlier necessary undo/redo must still be considered.

C - Incorrect: A checkpoint stores recovery metadata and is not a full separate database backup.

D - Incorrect: Checkpointing does not grant every active transaction exclusive locks.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB30

> Recovery may start analysis at the checkpoint but redo from an earlier required log position.

Study-note section ADB30

> A checkpoint is neither a full backup nor an automatic commit of active transactions.

Related reading (context, not primary note proof)

Study-note section ADB11

Question sources

Advanced Database Systems review questions · Question 37

Question 221 | NS0143

Under conventional deferred update with no-force, which recovery classifications describe the policy? Select all that apply. Interpret NO UNDO / REDO as 'no undo, redo required'.

A. UNDO/REDO

B. UNDO only

C. REDO only

D. NO UNDO / REDO

Hint

Separate update timing from update location and follow the commit boundary.

Answer C, D: C. REDO only; D. NO UNDO / REDO

Because uncommitted changes have not reached the persistent database, crash recovery does not need to undo them there. No-force permits committed changes to remain uninstalled, so redo may be required. REDO only and NO UNDO / REDO express the same classification under the stated notation. Undo-only and undo/redo both assert an undo requirement that this deferred model avoids.

Why each choice is right or wrong

A - Incorrect: Usual deferred update does not require undo of uncommitted persistent database changes.

B - Incorrect: Undo only gives the opposite recovery requirement from the usual deferred no-force model.

C - Correct: REDO only is the usual deferred-update recovery classification.

D - Correct: NO UNDO/REDO is another spelling of the same redo-only requirement.

Primary note proof

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB28

> Force writes a transaction's updated data pages at commit; no-force permits committed dirty pages to remain buffered, hence recovery may need redo.

Study-note section ADB28

> Steal permits a page with uncommitted changes to be written to disk; hence recovery may need undo.

Question sources

Advanced Database Systems review questions · Question 38

Question 222 | NS0144

Which data update strategy writes changes to a SEPARATE disk location preserving the original?

A. Immediate Update

B. Deferred Update

C. Shadow Update

D. In-Place Update

Hint

Separate update timing from update location and follow the commit boundary.

Answer C: Shadow update, or copy-on-write.

ADB29 distinguishes where an update is written from when it may become persistent. Shadow/copy-on-write uses a new location while retaining the committed version, matching C. Immediate and deferred classify commit timing, not necessarily a separate location. In-place update overwrites the original location, which directly contradicts the preservation condition in the stem.

Why each choice is right or wrong

A - Incorrect: Immediate update concerns allowing persistent changes before commit rather than requiring new locations.

B - Incorrect: Deferred update concerns withholding uncommitted persistent effects rather than necessarily copying pages.

C - Correct: Shadow/copy-on-write preserves the original and writes the changed value at a separate location.

D - Incorrect: In-place update overwrites the original location rather than preserving it intact.

Primary note proof

Study-note section ADB29

> In-place update overwrites the original location. Shadow/copy-on-write update writes a new location while preserving the committed version. Shadow paging keeps an unchanged committed page map and a modified map; commit durably installs a new root/map pointer atomically after required pages are durable. Before commit, a crash uses the old map. It is an alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Study-note section ADB29

> Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB29

> Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO.

Question sources

Advanced Database Systems review questions · Question 39

Question 223 | NS0145

In the course's conceptual transaction-log model supporting undo of incomplete changes and redo of committed changes, what value information does a write-operation record store?

A. Only new (after-image) values

B. Only old (before-image) values

C. Both old (before-image) AND new (after-image) values

D. Only transaction IDs and timestamps

Hint

Trace movement between program variables, buffers and disk, or the meaning of a recorded transaction event.

Answer C: C. Both old (before-image) AND new (after-image) values

The given UPDATE record contains both the old and new values. The old value supplies a before-image for undo, while the new value supplies an after-image for redo. Therefore the record stores both kinds of image. Real systems can use other physical or logical formats, but that variation does not alter the explicit conceptual record being examined here.

Why each choice is right or wrong

A - Incorrect: The given format includes old as well as new, so after-images alone are incomplete.

B - Incorrect: The format includes new as well as old, so before-images alone are incomplete.

C - Correct: Both before-image and after-image fields are present.

D - Incorrect: Transaction identifiers alone omit the explicitly supplied value fields.

Primary note proof

Study-note section ADB07

> A conceptual log can contain START, UPDATE(T,X,old,new), COMMIT, ABORT and END records. Before-images support undo; after-images support redo. Real systems may use physical or logical records, and need not log every read. COMMIT records successful completion; an ABORT record can mark the start of rollback rather than prove every undo step is complete.

Study-note section ADB27

> Undo restores before-images of incomplete effects, usually traversing that transaction's updates backward.

Study-note section ADB27

> In a simple undo/redo teaching model, undo losers and redo winners.

Question sources

Advanced Database Systems review questions · Question 40

Question 224 | NS0146

A distributed database is best described as:

A. A single database on one powerful server

B. Multiple logically related databases across networked computers, managed transparently

C. A fully replicated database at every network location

D. A web-based file storage system

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer B: Logically related data across networked sites under coordinated database management.

ADB33 defines logically related data across networked sites managed as a coordinated system. B matches that architecture and its logical abstraction. Full replication is only one placement choice, not a defining requirement. A single powerful server is centralized, and generic web file storage does not establish coordinated database semantics across related distributed data.

Why each choice is right or wrong

A - Incorrect: A database confined to one server is centralized rather than distributed across networked sites.

B - Correct: Networked, logically related data with coordinated management fits the distributed database definition.

C - Incorrect: A distributed database need not replicate every row at every network location.

D - Incorrect: Generic web file storage does not by itself establish coordinated distributed database management.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once.

Question sources

Advanced Database Systems review questions · Question 41

Question 225 | NS0147

A homogeneous distributed database system requires:

A. Different DBMS at each site

B. Same DBMS software at all sites

C. No network between sites

D. Full independence of each site

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer B: The same compatible DBMS technology at the sites.

ADB33 defines homogeneity through compatible DBMS technology across sites. B is the listed form of that condition. It does not imply identical local schemas, replication or immediate synchronization. Different DBMSs describe heterogeneity, while removing the network or demanding full independence contradicts coordinated distributed management rather than defining homogeneity.

Why each choice is right or wrong

A - Incorrect: Different DBMS technologies characterize heterogeneous integration rather than homogeneity.

B - Correct: The same DBMS software supplies the compatible database technology described by homogeneous systems.

C - Incorrect: Distributed sites need network communication for coordinated management.

D - Incorrect: Full local independence is an autonomy choice rather than a homogeneity requirement.

Primary note proof

Study-note section ADB33

> A distributed database contains logically related data across networked sites, managed as a coordinated system. A homogeneous system uses compatible DBMS technology across sites; a heterogeneous system integrates differing DBMSs, data models, schemas or interfaces. Different operating systems are possible but do not define the distinction. Same DBMS does not imply identical local schemas, automatic replication or immediate synchronization.

Study-note section ADB34

> These describe communication responsibilities, not fragmentation types.

Study-note section ADB34

State how much schema, transaction and administrative autonomy local sites retain.

Question sources

Advanced Database Systems review questions · Question 42

Question 226 | NS0148

Heterogeneous distributed databases use which standards for inter-database communication?

A. Same DBMS software only

B. Direct SQL between databases

C. ODBC and JDBC generic connectivity

D. LAN protocols exclusively

Hint

Separate coordinated network placement from engine compatibility and semantic integration.

Answer C: ODBC and JDBC connectivity interfaces.

ADB33 explicitly names ODBC and JDBC as connectivity interfaces for differing systems. C therefore identifies the offered communication interfaces. They do not automatically reconcile schemas, units or transaction semantics and do not create a distributed transaction by themselves. Identical DBMS software is not heterogeneity, and generic LAN protocols do not replace database-level connectivity interfaces.

Why each choice is right or wrong

A - Incorrect: Requiring the same DBMS removes the differing-engine situation rather than identifying its connectivity interfaces.

B - Incorrect: Direct SQL alone does not identify the generic cross-system connectivity standards named in the note.

C - Correct: ODBC and JDBC provide generic database connectivity while semantic mappings remain necessary.

D - Incorrect: LAN protocols alone do not supply the database API/interface compatibility described.

Primary note proof

Study-note section ADB33

> Heterogeneous design must reconcile naming, types, units, identifiers, query capabilities and transaction semantics. ODBC and JDBC provide connectivity interfaces; they do not automatically solve semantic differences or create one distributed transaction. Locality, expansion and replicated availability are potential benefits. Distribution adds communication, coordination and security costs; a site outage can still make needed data unavailable if no usable replica exists.

Related reading (context, not primary note proof)

Study-note section ADB24

Question sources

Advanced Database Systems review questions · Question 43

Question 227 | NS0149

Horizontal fragmentation divides a database table by:

A. Columns (attributes)

B. Rows based on a condition or criterion

C. Both rows and columns simultaneously

D. Database schemas

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Rows selected by conditions.

ADB36 defines horizontal fragmentation by selecting row subsets with predicates. B directly states that operation. Column division is vertical fragmentation, and combining both dimensions is hybrid fragmentation. Schema names and geographic placement can influence a design, but are not the definition of what is split by horizontal fragmentation.

Why each choice is right or wrong

A - Incorrect: Splitting attributes is vertical rather than horizontal fragmentation.

B - Correct: Horizontal fragmentation partitions rows according to stated predicates or criteria.

C - Incorrect: Splitting both dimensions combines horizontal and vertical operations in hybrid fragmentation.

D - Incorrect: Schemas are not the row units divided by the defined horizontal operation.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Hybrid fragmentation combines row and column splits in either suitable order.

Study-note section ADB36

> Reconstruct by reversing the operations: key joins for corresponding vertical pieces and unions for horizontal pieces.

Question sources

Advanced Database Systems review questions · Question 44

Question 228 | NS0150

To reconstruct a horizontally fragmented table, which operation is used?

A. JOIN

B. UNION

C. INTERSECTION

D. DIFFERENCE

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: UNION.

Horizontal fragments contain selected subsets of the original relation's rows. ADB36 reconstructs the relation by their union, assuming completeness and the intended overlap rules. Joining is used to align vertical column pieces by a key; intersection would keep only shared rows and difference would remove rows, so neither reconstructs all horizontal fragments.

Why each choice is right or wrong

A - Incorrect: Key joins reconstruct matching vertical column pieces, not ordinary horizontal row partitions.

B - Correct: Union collects the rows from every complete horizontal fragment.

C - Incorrect: Intersection keeps only common rows and would discard rows unique to individual fragments.

D - Incorrect: Difference removes rows instead of recombining all fragment members.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Reconstruct by reversing the operations: key joins for corresponding vertical pieces and unions for horizontal pieces.

Study-note section ADB-S01

> A relation is represented by a table of tuples.

Question sources

Advanced Database Systems review questions · Question 45

Question 229 | NS0151

Vertical fragmentation divides a table by:

A. Rows

B. Columns (attributes), with primary key repeated in each fragment

C. Number of concurrent transactions

D. Geographic location of data

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Column subsets with a repeated key or row identifier.

ADB36 splits a relation by columns and repeats a key or row identifier so the pieces can be joined without spurious matches. B describes that usual primary-key form. Rows are split horizontally; transaction counts and geography concern workload or placement rather than the attribute-level split defining vertical fragmentation. The shared identifier can be an appropriate equivalent key.

Why each choice is right or wrong

A - Incorrect: Row subsets define horizontal rather than vertical fragmentation.

B - Correct: Vertical fragments contain column subsets plus a repeated identifier for reconstruction.

C - Incorrect: Concurrent transaction count is a workload property rather than the dimension being split.

D - Incorrect: Geographic placement does not define the column projection used for vertical fragments.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Hybrid fragmentation combines row and column splits in either suitable order.

Study-note section ADB36

> Reconstruct by reversing the operations: key joins for corresponding vertical pieces and unions for horizontal pieces.

Question sources

Advanced Database Systems review questions · Question 46

Question 230 | NS0152

Full replication in distributed databases has which disadvantage?

A. Low data availability

B. Every update must be applied to all copies, creating high overhead

C. Fast read performance

D. Simple concurrency control

Hint

Separate copy placement, propagation timing and update-maintenance cost.

Answer B: Propagating each update across all copies adds maintenance overhead.

ADB37 says copies improve locality and can improve availability but require storage and update propagation. B names that maintenance burden for full replication. Propagation timing may be synchronous or asynchronous, so the answer does not imply every copy is updated simultaneously. Fast reads are a possible benefit, and concurrency control becomes more involved rather than inherently simple.

Why each choice is right or wrong

A - Incorrect: Usable replicas can improve availability rather than inherently making it low.

B - Correct: All copies must eventually reflect relevant updates, creating propagation and coordination overhead.

C - Incorrect: Fast reads are a potential benefit rather than the listed disadvantage.

D - Incorrect: Maintaining consistent copies does not inherently simplify concurrency control.

Primary note proof

Study-note section ADB37

> Full replication places the whole database at every relevant site; partial replication copies selected fragments to selected sites; nonredundant allocation stores each fragment once. Copies improve read locality and can improve availability but require storage and update propagation. They can also spread mistakes, so replication does not replace independent backups.

Study-note section ADB37

> Full replication does not inherently mean every copy is updated simultaneously; its synchronization rule must be stated.

Study-note section ADB37

> Lazy/asynchronous replication propagates later and may expose stale reads or need conflict resolution.

Question sources

Advanced Database Systems review questions · Question 47

Question 231 | NS0153

Fragmentation transparency in distributed databases means:

A. Users can see all the fragmentation details

B. Users interact with the database as a single entity, unaware of fragmentation existence

C. Only DBAs know about fragments

D. Fragments are listed as separate tables in the schema

Hint

Identify exactly which detail is hidden from the user.

Answer B: Users query the logical relation without knowing its fragments.

ADB35 defines fragmentation transparency as hiding the existence and reconstruction of fragments. B states that user-facing abstraction directly. It does not require literally only DBAs to know the implementation; developers and administrators may also need that knowledge. Exposing fragment tables or all fragmentation details to users would make fragmentation visible instead of transparent.

Why each choice is right or wrong

A - Incorrect: Showing all fragmentation details is the opposite of hiding them from ordinary queries.

B - Correct: Users operate on the whole logical relation without managing its fragment reconstruction.

C - Incorrect: Transparency describes the user abstraction, not exclusive knowledge restricted to DBAs alone.

D - Incorrect: Naming fragments separately exposes fragmentation rather than concealing it from the querying user.

Primary note proof

Study-note section ADB35

> Location transparency hides which site holds a named object. Fragmentation transparency hides the existence and reconstruction of fragments: users query the whole logical relation. Replication transparency hides which copy is used and how copies are maintained. The broad single-logical-database experience may involve all three.

Study-note section ADB35

> If users explicitly name a fragment but need not specify its site, fragmentation is visible while location can remain transparent.

Study-note section ADB35

> Hiding this column split from queries is fragmentation transparency; the split itself is a data-design operation.

Question sources

Advanced Database Systems review questions · Question 48

Question 232 | NS0154

The Middleware architecture in distributed DBMS uses:

A. Direct client-to-server connections

B. All servers collaborating on every query

C. A dedicated middleware server that mediates between clients and multiple independent database servers

D. A single centralized database server

Hint

Identify which tier submits, mediates or executes a request.

Answer C: Middleware mediates between clients and independent database servers.

ADB34 lets middleware translate requests, locate data, coordinate operations and combine results across independent servers. C describes that mediating tier. The logical responsibility does not require exactly one physical machine in every implementation. Direct connections omit this mediator, universal server collaboration is a different arrangement, and one centralized server is not the stated distributed middleware design.

Why each choice is right or wrong

A - Incorrect: Direct client-server links alone omit the middleware mediation that defines the requested architecture.

B - Incorrect: Every server need not participate in every query; relevant sources depend on the request.

C - Correct: A middleware tier mediates client requests across multiple independent database servers.

D - Incorrect: One centralized database server does not describe mediation across independent distributed servers.

Primary note proof

Study-note section ADB34

> In client/server architecture, clients submit requests while servers manage shared data, execute operations and enforce supported integrity/transaction rules. In collaborating-server designs, servers cooperate on subqueries and transactions. Middleware can translate requests, locate data, coordinate operations and combine results across independent servers. These describe communication responsibilities, not fragmentation types.

Question sources

Advanced Database Systems review questions · Question 49

Question 233 | NS0155

Hybrid fragmentation combines which two techniques?

A. Full replication and no replication

B. Horizontal fragmentation and vertical fragmentation

C. Locking and timestamp-based concurrency control

D. Client-server and middleware architectures

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer B: Horizontal and vertical fragmentation.

ADB36 explicitly defines hybrid fragmentation as combining row and column splits in an appropriate order. B names those two operations. Replication changes copy placement, locking and timestamps control concurrent access, and client-server/middleware describe architecture responsibilities. They can coexist with hybrid fragmentation but are not its two component fragmentation techniques.

Why each choice is right or wrong

A - Incorrect: Full/no replication concerns copies rather than the row and column splits of hybrid fragmentation.

B - Correct: Hybrid fragmentation combines horizontal row subsets with vertical column subsets.

C - Incorrect: Locking and timestamps are concurrency-control mechanisms, not fragmentation types.

D - Incorrect: These are architecture forms rather than the two data-fragmentation operations.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB36

> Hybrid fragmentation combines row and column splits in either suitable order.

Study-note section ADB34

> These describe communication responsibilities, not fragmentation types.

Related reading (context, not primary note proof)

Study-note section ADB37

Study-note section ADB11

Question sources

Advanced Database Systems review questions · Question 50

Question 234 | NS0286

In client-server database architecture, what is the primary role of the server?

A. Provide the GUI for end users

B. Initiate all network connections and poll clients for updates

C. Store and manage shared data, enforce ACID properties, and process client requests

D. Cache all user sessions locally for offline access

Hint

Identify which tier submits, mediates or executes a request.

Answer C: Manage shared data, process requests and enforce supported transaction/integrity guarantees.

ADB34 assigns clients request submission and servers shared-data management, operation execution and supported integrity/transaction enforcement. C matches those responsibilities. User interfaces usually belong to clients, while initiating all connections, polling every client or caching all sessions for offline use are not the server role defined by this architecture. Guarantees still depend on supported mechanisms and correct application use.

Why each choice is right or wrong

A - Incorrect: The user-facing GUI is normally a client responsibility rather than the database server's primary role.

B - Incorrect: The server need not initiate every connection or poll clients for every update.

C - Correct: Shared-data management, query execution and supported integrity/transaction enforcement are server duties.

D - Incorrect: Caching every session for offline access is not the defined database-server responsibility.

Primary note proof

Study-note section ADB34

> In client/server architecture, clients submit requests while servers manage shared data, execute operations and enforce supported integrity/transaction rules. In collaborating-server designs, servers cooperate on subqueries and transactions. Middleware can translate requests, locate data, coordinate operations and combine results across independent servers. These describe communication responsibilities, not fragmentation types.

Study-note section ADB34

> Calling a system heterogeneous does not require all sites to surrender autonomy or to copy every row everywhere.

Study-note section ADB34

> A tightly integrated distributed system can present one global schema and shared control.

Related reading (context, not primary note proof)

Study-note section ADB06

Question sources

Computer Science Exit Exam · 2018 · Question 3

Question 235 | NS0288

In two-phase locking (2PL), during which phase may a transaction acquire locks but not release any?

A. Growing phase

B. Commit phase

C. Validation phase

D. Shrinking phase

Hint

Track the first lock release and the transaction-completion boundary.

Answer A: Growing phase.

ADB12 allows acquiring or upgrading locks during growing and releasing or downgrading them during shrinking. Once any lock is released, a transaction may not acquire another. A is therefore the phase described. Commit is a transaction event, validation belongs to optimistic control and shrinking has the opposite lock-release role.

Why each choice is right or wrong

A - Correct: Growing permits acquisitions and upgrades before any release starts.

B - Incorrect: Commit is not the general 2PL phase for new lock acquisition.

C - Incorrect: Validation is an optimistic-control stage rather than the 2PL growing phase.

D - Incorrect: Shrinking releases or downgrades locks and forbids acquiring new ones.

Primary note proof

Study-note section ADB12

> Basic 2PL has a growing phase that acquires/upgrades locks and a shrinking phase that releases/downgrades them. Once any lock is released, no new lock may be acquired. The lock point is the instant the final lock is obtained. 2PL guarantees conflict serializability for well-formed transactions, but can deadlock and basic 2PL alone need not prevent cascading aborts.

Study-note section ADB12

> Two-phase locking orders access to data; two-phase commit coordinates a distributed commit decision.

Study-note section ADB16

> Optimistic/validation control reads and computes privately, validates its read/write sets against overlapping transactions, then publishes if validation succeeds; otherwise it retries.

Question sources

Computer Science Exit Exam · 2018 · Question 5

Question 236 | NS0306

Assume that we are able to specify that Mr. Getahun can view the database record, but cannot change the value of the record. Which security management technique was used in this scenario?

A. Access Control

B. Availability

C. Non-repudiation

D. Integrity

Hint

Identify whether the scenario concerns identity, permission, disclosure or valid values.

Answer A: Access control.

ADB24 explicitly describes a user allowed to view but not change a record as having restricted access privileges. That is authorization/access control. Integrity concerns valid values, availability concerns readiness for use and non-repudiation concerns attributable actions rather than whether a particular read or write is permitted. A matches the actual permission distinction.

Why each choice is right or wrong

A - Correct: Access control grants reading while denying modification for the specified user.

B - Incorrect: Availability concerns service readiness rather than the read-versus-write privilege distinction.

C - Incorrect: Non-repudiation does not define whether this user is authorized to modify a record.

D - Incorrect: Integrity restricts valid data states rather than assigning user-specific read/update permissions.

Primary note proof

Study-note section ADB24

> Authentication establishes identity; authorization/access control determines permitted actions. Encryption protects stored or transmitted content; integrity constraints restrict valid data; auditing records actions. A user who can view but cannot change a record has restricted access privileges. A view can restrict visible rows/columns, but its privilege and execution rules must support that restriction.

Study-note section ADB-S03

> Neither property specifies which particular user has permission to change a database row; that is the separate authorization decision.

Study-note section ADB-S03

> Availability concerns keeping services and resources accessible to their authorized users.

Question sources

Computer Science Exit Exam · 2018 · Question 27

Question 237 | NS0307

Suppose, a developer proposes using deferred update with no checkpoints in a high-transaction OLTP system. What is the most serious drawback?

A. Log scans become prohibitively expensive after long uptime

B. Increased disk I/O due to frequent page copying

C. Shared locks cannot be used

D. View serializability cannot be ensured

Hint

Identify which history a recovery boundary can safely exclude and whether active work spans it.

Answer A: Long-running log scans can make recovery expensive.

ADB30 warns that without checkpoints or another valid truncation boundary, a long-running system may scan expensive amounts of log history. Deferred update removes uncommitted persistent effects but does not remove that search problem. Page copying belongs to shadow approaches, while shared locks and serializability are independent concurrency concerns rather than consequences of omitting checkpoints.

Why each choice is right or wrong

A - Correct: High transaction volume and long uptime can produce costly recovery log scans without a valid boundary.

B - Incorrect: Frequent copy-on-write pages characterize shadow updating rather than the absence of checkpoints in deferred update.

C - Incorrect: Omitting checkpoints does not inherently prohibit shared locks.

D - Incorrect: View serializability is a concurrency-order property, not inherently lost by this checkpoint omission.

Primary note proof

Study-note section ADB30

> A checkpoint records enough durable recovery state to avoid scanning all old history. In a simple quiescent/checkpoint exercise, specified flushing may make earlier committed work unnecessary to redo. Active transactions can span the checkpoint, so their relevant earlier updates can still need undo. Do not assume every transaction before a checkpoint needs no processing.

Study-note section ADB30

> Without checkpoints or another valid truncation boundary, long-running systems may need expensive log scans; deferred update does not avoid that problem.

Study-note section ADB29

> Shadow/copy-on-write update writes a new location while preserving the committed version.

Related reading (context, not primary note proof)

Study-note section ADB11

Study-note section ADB09

Question sources

Computer Science Exit Exam · 2018 · Question 28

Question 238 | NS0321

Which listed techniques use transaction logs for recovery under the course model? Interpret Deferred Database Modification as conventional logged deferred update with no-force. Select all that apply.

A. Recovery in Multi-database Systems

B. ARIES Recovery Algorithm

C. Shadow Paging Technique

D. Deferred Database Modification

Hint

Separate update timing from update location and follow the commit boundary.

Answer B, D: B. ARIES Recovery Algorithm; D. Deferred Database Modification

ARIES relies on WAL and log-driven analysis, redo and undo. Conventional logged deferred update also uses recorded updates for redo of committed work, with no undo of uncommitted persistent changes. Both therefore belong to the requested category. Shadow paging preserves old page mappings instead, while recovery in multidatabase systems names an environment rather than a particular log-based technique.

Why each choice is right or wrong

A - Incorrect: Multidatabase recovery describes an environment rather than one specific recovery technique.

B - Correct: ARIES explicitly uses WAL and log-driven analysis, redo and undo.

C - Incorrect: Shadow paging preserves committed page maps rather than being the conventional log-based technique.

D - Correct: Deferred modification can use logged changes for redo after commitment.

Primary note proof

Study-note section ADB29

Deferred update keeps uncommitted changes outside the persistent database; in the usual no-force setting it is NO-UNDO/REDO. Redo-only and NO-UNDO/REDO mean the same thing, so those cannot be separate unique answers. Immediate update allows persistent changes before commit and therefore may require undo; it need not flush every write_item immediately.

Study-note section ADB31

ARIES uses WAL, page log sequence numbers (pageLSN), a transaction table and a dirty-page table.

Study-note section ADB29

It is an alternative to conventional log-based recovery, with allocation, fragmentation and concurrency costs; it is not primarily a special action done only during recovery.

Question sources

Computer Science Exit Exam · 2018 · Question 46

Question 239 | NS0348

Which type of fragmentation divides a relation into subrelations containing subsets of its columns, repeating a key or row identifier so the original rows can be reconstructed?

A. Replication

B. Location

C. Horizontal

D. Vertical

Hint

Identify whether rows or columns are split and reverse the split to reconstruct the original relation.

Answer D: D. Vertical

Vertical fragmentation divides a relation by columns. Repeating a key or row identifier allows corresponding pieces to be joined back into the original rows without spurious matches. Horizontal fragmentation divides rows instead. Replication creates copies, while location transparency hides where data resides. Hiding a column split would be fragmentation transparency, but the question asks for the split operation itself.

Why each choice is right or wrong

A - Incorrect: Replication copies data rather than dividing its columns.

B - Incorrect: Location transparency hides sites rather than splitting a relation.

C - Incorrect: Horizontal fragmentation selects row subsets.

D - Correct: Vertical fragmentation projects column subsets with a reconstruction identifier.

Primary note proof

Study-note section ADB36

> Horizontal fragmentation selects row subsets by predicates; their union reconstructs the relation. Check completeness so every original row is represented and disjointness if no overlap is intended. Define an explicit fragment for NULL/other values when needed. Vertical fragmentation projects column subsets and repeats a key or row identifier so joining on it reconstructs the original rows without spurious matches.

Study-note section ADB35

> Hiding this column split from queries is fragmentation transparency; the split itself is a data-design operation.

Study-note section ADB35

> Replication transparency hides which copy is used and how copies are maintained.

Related reading (context, not primary note proof)

Study-note section ADB37

Question sources

Computer Science Exit Exam · 2018 · Question 82