CS Exit Ethiopia

Questions & Worked Answers

Computer Science Exit Examination

226 complete questions and 2 source variants

1 selected course | 18 November 2026

Selected courses

01. Fundamental of 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.

Fundamental of 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.

Fundamental of Database Systems

Questions & Worked Answers

226 questions + 2 source variants

Question 1 | Q00021

Which one of the following is not correct when mapping ER diagrams to relational tables?

A. Multi-valued attributes are mapped to new tables

B. Foreign key are used to map relationships to create relation between tables

C. Table names are directly mapped from entity names

D. Composite attribute are mapped as it is

Hint

Apply the ER mapping rule for the given attribute or entity type.

Answer D: Composite attribute are mapped as it is

Atomic relational values exclude storing an undecomposed composite attribute. Relation names need not differ from entity names.

Why each choice is right or wrong

A - Incorrect: Multivalues are separated to remove repeating groups (PDF 12, 17.)

B - Incorrect: Foreign-key mapping is explicitly described (PDF 12, 17.)

C - Incorrect: Naming a relation after its entity is compatible with the mapping (PDF 12, 17.)

D - Correct: Keeping a composite value unchanged violates atomicity (PDF 12, 17.)

Primary note proof

Study-note section FDB10

> Create one relation per strong entity type, preserve its simple attributes and select a candidate key. Expand composite attributes into their useful simple components; do not copy a composite ER structure unchanged into a flat atomic design. Derived values can usually be calculated, avoiding stale stored copies.

Study-note section FDB10

> Table names may follow entity names; naming does not invalidate a mapping.

Study-note section FDB10

> Map each multivalued attribute to a separate relation containing the owner key and one value.

Related reading (context, not primary note proof)

Study-note section FDB11

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 9-24

> 1. For each strong entity type E, create a relation R that includes all the attributes of E. 2. Choose one of the key attributes of E as the primary key for R. For each weak entity type W with owner entity type E (which is a strong entity), create a relation R & include all attributes of _W‘ as attributes of R. Also, include the primary key attribute of the relation that correspond to the owner entity type as foreign key attributes of R . 3. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Foreign Key approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. 4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as foreign key in S that represents the other entity type participating in R. Include any attributes of the 1:N relation type as attributes of S. 5. For each n-ary relationship type R, where n>2, create a new relationship S to represent R. Include the primary keys of the relations that represent the participating entity types as foreign key attributes in S. Also include any attributes of the n-ary relationship type as attributes of S.

Fundamental of Database Systems module | PDF page 17 | lines 2-9

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF).

Question sources

MOE Computer Science Exit Exam · 2016 · Question 21

Question 2 | Q00038

In DBMS-independent logical relational design, before vendor-specific refinement, which statement is NOT correct?

A. It consider a particular DBMS

B. Normalization is undertaken in logical phase

C. It is based on specific data model

D. It is independent from physical considerations

Hint

Identify the design phase from its inputs and intended deliverable.

Answer A: A. It consider a particular DBMS

Logical design maps the conceptual model into relations, keys and constraints and checks normalization. It therefore uses a particular data model while separating storage decisions from logical structure. Under the stated DBMS-independent methodology, considering a particular vendor's DBMS is the false claim. Normalization, a chosen model and independence from physical storage are all consistent with this phase.

Why each choice is right or wrong

A - Correct: False in the stated phase: particular-DBMS implementation choices are deferred.

B - Incorrect: True: normalization checks the mapped relations.

C - Incorrect: True: relational logical design uses a specific data model.

D - Incorrect: True: file layouts and indexes belong to physical design.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Study-note section FDB05

> Whether logical design is DBMS-dependent depends on the stated design methodology.

Study-note section FDB05

> Logical relational design is tied to a data model but can remain independent of a particular DBMS.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 21-22

> Logical database design (Data model mapping) The conceptual design is translated into internal model which includes mapping of all objects i.e

Fundamental of Database Systems module | PDF page 19 | lines 6-7

> data, a physical database design is needed. The goal is to create a design for storing and retrieving data that will provide adequate performance and insure database integrity, security, and

Question sources

MOE Computer Science Exit Exam · 2016 · Question 38

Question 3 | Q00040

Which SQL command would you use if you want to specify constraints to be applied on tuples?

A. GRANT

B. SELECT

C. REVOKE

D. CHECK

Hint

Read each SQL constraint's scope and enforcement rule.

Answer D: CHECK

CHECK verifies criteria on attribute values within rows; GRANT and REVOKE control permission, and SELECT retrieves data.

Why each choice is right or wrong

A - Incorrect: Permission grant is not a tuple predicate (PDF 49, 54.)

B - Incorrect: Retrieval is not constraint specification (PDF 49, 54.)

C - Incorrect: Permission removal is not a tuple predicate (PDF 49, 54.)

D - Correct: CHECK is explicitly shown verifying criteria (PDF 49, 54.)

Primary note proof

Study-note section FDB16

> Example: CREATE TABLE Enrollment (StudentID INTEGER NOT NULL REFERENCES Student(StudentID), CourseID INTEGER NOT NULL REFERENCES Course(CourseID), Grade INTEGER CHECK (Grade BETWEEN 0 AND 100), PRIMARY KEY (StudentID,CourseID)); Each column declaration gives its name then type, with commas between declarations. CHECK is a constraint clause, not a standalone GRANT-like statement. POSITIVE is not a general SQL constraint keyword; CHECK (Amount > 0) expresses positivity.

Study-note section FDB16

> A CHECK can pass when its result is unknown, so add NOT NULL if missing values must be forbidden.

Study-note section FDB26

> DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 49 | lines 19-22

● CREATE TABLE table-name (attr2 attr-type NOT NULL, forbids NULL values attrN attr-type CHECK (attrN = UPPER(attrN) verifies that the data meets certain criteria attrO attr-type DEFAULT default_value); Supplies default values

Fundamental of Database Systems module | PDF page 54 | lines 6-18

2.2.3 Authorization Rules ● Most current DBMS permit the DBA to define —access permissions‖ on a table by table basis (at least) using the GRANT and REVOKE SQL commands ● Some systems permit finer grained authorization (most use GRANT and REVOKE on variant views 2.2.4 Privileges in SQL 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

Fundamental of Database Systems module | PDF page 26 | lines 10-11

7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

MOE Computer Science Exit Exam · 2016 · Question 40

Question 4 | Q00041

Which data management approach allows application programs to define and manage their own data?

A. Manual approach

B. Database approach

C. Hierarchical approach

D. Traditional file based approach

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer D: Traditional file based approach

The file approach lacks the database approach separation between programs and data; application-owned data corresponds to traditional file processing. B reverses that distinction.

Why each choice is right or wrong

A - Incorrect: Manual records do not describe application-owned files (PDF 5, 6.)

B - Incorrect: Program/data insulation contradicts the stem (PDF 5, 6.)

C - Incorrect: Hierarchical is a model rather than this processing distinction (PDF 5, 6.)

D - Correct: Traditional file processing matches the contrast (PDF 5, 6.)

Primary note proof

Study-note section FDB01

A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

Traditional file applications define their own formats and access procedures.

Study-note section FDB04

Data abstraction hides irrelevant details; program-data insulation separates data definitions from application code.

Related reading (context, not primary note proof)

Study-note section FDB02

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 17-25

1.2 Database System versus File System The traditional file processing system is file-directory structure supported by a conventional operating system. A file system organization of data lucks a number of major features of a database system, such as Data redundancy and inconsistency, Difficulty in accessing data, Data isolation, Integrity problems, Atomicity problems, Concurrent access anomalies. 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.

Fundamental of Database Systems module | PDF page 6 | lines 2-4

Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Question sources

MOE Computer Science Exit Exam · 2016 · Question 41

Question 5 | Q00043

Which database constraints are specified in the data definition language?

A. Implicit or inherent model based constraints

B. Semantic constraints

C. Application based constraints

D. Schema based constraints

Hint

Match the validity requirement to the appropriate constraint definition.

Answer D: Schema-based constraints.

The question asks how constraints are specified, rather than which business meaning they express. FDB15 distinguishes constraints inherent in a data model from constraints explicitly declared in schema DDL. That second category is schema-based. Semantic or application rules may require triggers or application code when ordinary DDL cannot express them.

Why each choice is right or wrong

A - Incorrect: Inherent model constraints follow from the data model rather than explicit schema declarations.

B - Incorrect: Semantic rules express application meaning and may exceed what ordinary schema DDL can enforce.

C - Incorrect: Application-based constraints may require program logic rather than declarations in the database definition.

D - Correct: Schema-based constraints are explicitly specified in data definition language declarations.

Primary note proof

Study-note section FDB15

> Inherent constraints come from the data model; schema-based constraints are declared through the DDL; semantic/application rules may need triggers or application logic when DDL cannot express them. Integrity improves data validity; authorization controls who may access it. Uniqueness and domain validation are both integrity mechanisms. If a Department primary key is NULL and Manager=107 lacks a referenced employee, entity integrity certainly fails and referential integrity also fails if that foreign key is declared. One INSERT can violate multiple constraints.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 5-6

> has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language).

Fundamental of Database Systems module | PDF page 49 | lines 17-18

- This is now increasing handled by the database. In Oracle, for example, when defining a table you can specify:

Question sources

MOE Computer Science Exit Exam · 2016 · Question 43

Question 6 | Q00080

Which SQL command removes an existing relation's table object, including its definition?

A. Delete

B. Drop

C. Remove

D. UPDATE

Hint

Classify each SQL command by the kind of change it performs.

Answer B: B. Drop

Removing a relation object means removing its definition as well as its contents. DROP TABLE performs that operation. DELETE affects rows while retaining the table, and UPDATE changes values within rows. REMOVE is not the SQL table-removal command. The distinction is between schema removal and operations on data in an existing schema.

Why each choice is right or wrong

A - Incorrect: DELETE removes rows and retains the table definition.

B - Correct: DROP TABLE removes the table object and its definition.

C - Incorrect: REMOVE is not the SQL command for dropping a table.

D - Incorrect: UPDATE changes existing row values.

Primary note proof

Study-note section FDB26

DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 8-9

> Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME

Fundamental of Database Systems module | PDF page 26 | lines 11-11

> Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

MOE Computer Science Exit Exam · 2016 · Question 80

Question 7 | Q00106

Which of the following statement best describe relational data model?

A. It is a database model that allows multiple records to be linked to the same owner file.

B. It structures data in a tree like structure using parent to child relationship.

C. It is a database model to manage data as tuples grouped into relations

D. It is a model that use concepts such as entities, attributes, and relationships

E. Describe how data is stored in the computer, representing information such as record structures, record ordering, and access paths.

Hint

Compare the data models by how they organize records and relationships.

Answer C: It is a database model to manage data as tuples grouped into relations

A relational database consists of relations with tuple rows and attribute columns.

Why each choice is right or wrong

A - Incorrect: Owner-file links do not define tuple relations (PDF 13, 7.)

B - Incorrect: A tree is not the stated relational representation (PDF 13, 7.)

C - Correct: Tuple rows in relations match the definition (PDF 13, 7.)

D - Incorrect: Entities and relationships describe conceptual ER modeling (PDF 13, 7.)

E - Incorrect: Storage details define the physical model (PDF 13, 7.)

Primary note proof

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one query.

Study-note section FDB03

> Degree counts columns; cardinality counts rows.

Study-note section FDB03

> A tuple/row/record contains related attribute values; an attribute/column/field describes a property.

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 19-20

> The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple.

Fundamental of Database Systems module | PDF page 7 | lines 12-13

> 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant

Question sources

Addis Ababa University mock exam · Question 8

Question 8 | Q00107

Department(Dept_Name, Manager) has primary key Dept_Name and foreign key Manager referencing Employee(Emp_ID). Employee contains IDs 101-105 only. Consider INSERT INTO Department (Dept_Name, Manager) VALUES (NULL, 107); Which named integrity rules are violated? Select all that apply. Treat key integrity as uniqueness and domain integrity as type/allowed-value checks.

A. Domain constraint

B. Key constraint

C. Entity Integrity constraint

D. Referential Integrity constraint

Hint

Match the validity requirement to the appropriate constraint definition.

Answer C, D: C. Entity Integrity constraint; D. Referential Integrity constraint

The inserted row has two independent defects. Dept_Name is the primary key, so NULL violates entity integrity. Manager is a non-NULL foreign key, yet employee 107 is absent, violating referential integrity. These are separate requirements and the same insertion fails both. Under the question's taxonomy, neither a duplicate-key violation nor an incompatible-domain value is being asserted.

Why each choice is right or wrong

A - Incorrect: The stated domain checks are satisfied; primary-key NULL is classified separately.

B - Incorrect: No duplicate key value is inserted; this category tests uniqueness.

C - Correct: NULL in the primary key violates entity integrity.

D - Correct: Manager 107 has no referenced employee and violates referential integrity.

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 12-13

> ◦ Cannot be NULL iv. Referential Integrity

Fundamental of Database Systems module | PDF page 48 | lines 14-15

> • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key

Question sources

Addis Ababa University mock exam · Question 9

Question 9 | Q00108

Assume atomic attribute values. Consider R(A,B,C,D) with functional dependencies F = {C→D, C→B, B→C}. Which statement about R is true?

A. The relation R is in second normal form

B. The relation R is in third normal form

C. The candidate key for R is BC

D. The relation R is in Boyce-Codd normal form

E. None of the other statements is true

Hint

Derive every candidate key before testing dependency-based normal forms.

Answer E: None of the other statements is true

Every candidate key must contain A because no dependency derives it. Both AB and AC determine all attributes and are minimal; BC determines only B, C and D. The nonprime attribute D depends on B or C alone, a proper subset of either composite key, so 2NF fails. The dependency C→D also violates 3NF and BCNF because C is not a superkey and D is nonprime. None of the other statements is true.

Why each choice is right or wrong

A - Incorrect: False: D depends on a proper subset of a candidate key, violating 2NF.

B - Incorrect: False: C→D has a nonsuperkey determinant and a nonprime dependent attribute, violating 3NF.

C - Incorrect: False: $(BC)^+ = \{B,C,D\}$, so BC cannot determine A and is not a key.

D - Incorrect: False: C determines D but is not a superkey, violating BCNF.

E - Correct: Correct: each of the other four claims is false.

Primary note proof

Study-note section FDB18

> For R(A,B,C,D), let C→D, C→B and B→C. A never appears on a right-hand side, so every key needs A. $(AB)^+$ gives ABCD through B→C→D; $(AC)^+$ also gives ABCD. AB and AC are minimal keys. BC cannot derive A and is not a key. A, B and C are prime; D is nonprime.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 15 | lines 5-17

> A functional dependency is a constraint between two attributes or sets of attributes of entities in a database. A functional dependency is denoted by X → Y, between two sets of attributes X and Y that are subsets of relation R. The constraint is that, the values of Y component of a tuple in r depend on, or are determined by the values of X component. Alternatively, the values of X component of a tuple uniquely (or functionally) determine the values of Y component. 4.2 Types of Functional Dependency Partial Dependency - If an attribute which is not a member of the key is dependent on some part of the key (if we have composite/candidate key) then that attribute is partially functionally dependent on the key. Let {A, B} is the key set and C is no key attribute, then if {A, B} → C and [B → C or A → C] holds. We say C is partially functionally dependent on {A, B}. Partial dependency occurs when a non-prime attribute is functionally dependent on part of a candidate/composite key.

Fundamental of Database Systems module | PDF page 17 | lines 10-15

> Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Addis Ababa University mock exam · Question 10

Question 10 | Q00109

Which schema of the three-schema Architecture describe the structure of the whole database?

A. Internal schema

B. Conceptual schema

C. External schema

D. schema

Hint

Trace which schema levels a proposed change affects.

Answer B: Conceptual schema

The conceptual schema describes the whole database while internal describes storage and external describes each user view.

Why each choice is right or wrong

A - Incorrect: Internal covers physical storage (PDF 8.)

B - Correct: Conceptual explicitly covers the whole database (PDF 8.)

C - Incorrect: External covers a user subset (PDF 8.)

D - Incorrect: Generic schema does not identify the required level (PDF 8.)

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The

internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 25-30

2.3 Three-schema Architecture and Data Independence The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels: 1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Fundamental of Database Systems module | PDF page 8 | lines 2-7

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. 3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Question sources

Addis Ababa University mock exam · Question 11

Question 11 | Q00110

Which of the following is the disadvantage of a database approach?

A. Data consistency

B. Sharing of data

C. Improved data integrity

D. Increased concurrency

E. Enforcement of standards

F. Complexity

G. None

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer F: Complexity is a database-approach disadvantage.

FDB01 identifies complexity and administration costs as possible disadvantages of a DBMS. Consistency, sharing, integrity and centrally supported standards are benefits. Increased concurrency means useful concurrent access here; only uncontrolled concurrent access causes anomalies. Thus complexity is the sole listed disadvantage, even though database management can improve the other listed properties.

Why each choice is right or wrong

A - Incorrect: Data consistency is improved by shared definitions and controlled updates, making it a benefit.

B - Incorrect: Sharing data is a purpose and benefit of the database approach.

C - Incorrect: Improved integrity means better enforcement of valid data and is a benefit.

D - Incorrect: Concurrent access is useful; uncontrolled interference, rather than increased concurrency itself, causes problems.

E - Incorrect: Central management supports consistent definitions and standards, which is a benefit.

F - Correct: A DBMS introduces administrative and technical complexity that can outweigh benefits for small applications.

G - Incorrect: A listed disadvantage exists: the complexity of database management.

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

> Concurrent access itself is useful; uncontrolled concurrent access causes anomalies.

Study-note section FDB01

> Separate files can repeat facts, disagree after updates, isolate useful data and make shared integrity or concurrent updates difficult.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 23-24

> The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and

Fundamental of Database Systems module | PDF page 117 | lines 6-7

i. Complexity- The data replication , failure recovery , network management ...make the system more complex than the central DBMSs

Question sources

Addis Ababa University mock exam · Question 12

Question 12 | Q00111

Which of the following is a correct statement about relation? (Choose all that applies)

A. Values of each columns is a repeating group/array

B. Values in a column can be from different domain

C. A relation is in the INF

D. Two rows in a relational table can be identical

E. The sequence of columns and rows is insignificant

F. None

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer C, E: A relation is in the INF; The sequence of columns and rows is insignificant

Read INF as 1NF. Relations have atomic cells rather than repeating groups; the relational representation is independent of tuple sequence. The column sequence likewise does not alter named attributes. Distinct tuples and typed attribute domains rule out A,B,D.

Why each choice is right or wrong

A - Incorrect: Repeating arrays violate atomicity (PDF 14, 17.)

B - Incorrect: An attribute has a constrained domain (PDF 14, 17.)

C - Correct: Atomic relation values satisfy 1NF (PDF 14, 17.)

D - Incorrect: Duplicate tuples conflict with relation uniqueness (PDF 14, 17.)

E - Correct: Presentation order does not change named tuple values (PDF 14, 17.)

F - Incorrect: C and E already supply valid statements (PDF 14, 17.)

Primary note proof

Study-note section FDB03

A relation is a set of tuples, represented as a table. A tuple/row/record contains related attribute values; an attribute/column/field describes a property. Degree counts columns; cardinality counts rows. Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning. Column order has no logical meaning when attributes are identified by name. SQL tables can contain duplicate rows unless constrained.

Study-note section FDB20

1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

A mapped ER design that correctly splits composite and multivalued structures gives ordinary 1NF tables; ER drawing alone guarantees no higher form.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 14 | lines 4-20

Properties of Relations • Name of the relation is distinct from all other relations. • Each relation cell contains exactly one atomic (single) value. • Each attribute contains a distinct name. • Each tuple has no duplicate value. • Order of tuple can have a different sequence. 3.8 Relational Constraints Relational constraints in DBMS are referred to conditions which must be present for a valid relation. These Relational constraints in DBMS are derived from the rules in the system that the database represents. There are many types of Constraints in DBMS. Domain Constraints - Every domain (cell in the table or attribute value) must contain atomic values (smallest indivisible units). We perform data type check here, which means we need to assign a data type to a column and limit the values that it can contain. Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Fundamental of Database Systems module | PDF page 17 | lines 2-9

First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF).

Question sources

Addis Ababa University mock exam · Question 13

Question 13 | Q00112

Which of the following statement is correct about executing the following SQL query on the employee database given in question 2? (Choose all that applies)
SELECT Emp_ID, First_Name, Last_Name
FROM Employee E, Department D
WHERE Department <> Sales' and AVG (salary) > 35000 and
E.Department=D.Dept_Name
GROUP BY Department;

A. the query will execute successfully and will obtain all departments with an average salary higher than 35,000

B. the query will not execute because Department attribute in Group by statement was not included in the select statement list

C. the query will not execute because we can't include aggregate function in the where statement

D. the query will execute successfully and will obtain employees of all departments with an average salary higher than 35,000

Hint

Apply grouping before filtering groups and computing aggregate output.

Answer C: AVG(salary) cannot be used directly in WHERE.

The query places AVG(salary) > 35000 in WHERE. FDB32 requires ordinary aggregate predicates in HAVING. Omitting Department from SELECT is not itself an error. Selecting ungrouped employee details is another defect unless functional dependence is established. The query therefore does not successfully produce either claimed result.

Why each choice is right or wrong

A - Incorrect: The aggregate predicate in WHERE prevents this query from successfully producing department averages.

B - Incorrect: A grouping column need not appear in SELECT; that omission is not the stated error.

C - Correct: WHERE filters individual rows before grouping, so the ordinary AVG predicate belongs in HAVING.

D - Incorrect: The invalid aggregate predicate prevents execution; employee columns also require valid grouping or functional dependence.

Primary note proof

Study-note section FDB32

> COUNT(*) counts rows, COUNT(x) counts non-NULL x values; SUM and AVG summarize non-NULL numeric values. SQL's usual average function is AVG, not AVERAGE. GROUP BY creates one group per distinct grouping value; HAVING tests aggregate groups after WHERE has filtered individual rows.

Study-note section FDB32

> The grouping column need not itself appear in SELECT; check the functional dependence of other selected columns.

Study-note section FDB32

> SELECT Department, AVG(Salary) FROM Employee WHERE Department <> 'Sales' GROUP BY Department HAVING AVG(Salary) > 35000; lists qualifying department averages.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 11-11

> Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 30 | lines 4-5

> • Verify relations and attributes exist. • Verify operations

Question sources

Addis Ababa University mock exam · Question 14

Question 14 | Q00219

Which of the following keys is generally used to represent the relationships between the tables?

A. None of the choices is correct

B. Secondary key

C. Primary key

D. Foreign key

Hint

Match the validity requirement to the appropriate constraint definition.

Answer D: Foreign key

A foreign key is the child-side reference linking to an existing parent candidate key.

Why each choice is right or wrong

A - Incorrect: A named key does supply the link (PDF 48.)

B - Incorrect: Secondary key is not the child reference (PDF 48.)

C - Incorrect: Primary key is the target identifier (PDF 48.)

D - Correct: Foreign key is the explicit link (PDF 48.)

Primary note proof

Study-note section FDB15

Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB07

A primary key is one selected candidate key; remaining candidates are alternate keys.

Study-note section FDB25

Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 14-15

- A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key

Question sources

Mock exam 4 · Question 21

Question 15 | Q00220

Which of the following refers to the level of data abstraction that describes exactly how the data is actually stored?

A. Conceptual Level

B. Physical Level

C. File Level

D. Logical Level

Hint

Trace which schema levels a proposed change affects.

Answer B: Physical Level

Physical abstraction describes storage-media details.

Why each choice is right or wrong

A - Incorrect: Conceptual hides storage (PDF 7.)

B - Correct: Physical covers storage details (PDF 7.)

C - Incorrect: File is not a three-schema level (PDF 7.)

D - Incorrect: Logical describes structure rather than layout (PDF 7.)

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 12-13

> 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant

Question sources

Mock exam 4 · Question 22

Question 16 | Q00221

Which one of the following given statements possibly contains the error?

A. select empid where empid = 1009 and Lastname = 'ADDIS';

B. select empid from emp where empid = 10006;

C. select empid from emp;

D. select * from emp where empid = 10003;

Hint

Trace how SELECT forms its result columns and rows.

Answer A: The first SELECT is missing its FROM source.

The first query reads empid and Lastname but never names a table in FROM. FDB27 allows literal-only SELECT statements without FROM, but this statement is not literal-only. The other three statements include emp as their source and use ordinary valid selection syntax, assuming those table and column names exist.

Why each choice is right or wrong

A - Correct: It refers to table columns without a FROM clause identifying their source.

B - Incorrect: It selects empid from emp with a syntactically ordinary equality predicate.

C - Incorrect: It supplies emp as the table source and validly selects its empid column.

D - Incorrect: It supplies a table source, selects all columns, and filters by empid.

Primary note proof

Study-note section FDB27

> SELECT Name,Address FROM Customer; produces only those mailing-label fields. SELECT * selects all columns; DISTINCT removes duplicate result rows; AS supplies an alias; ALL permits duplicates and does not mean all columns. SQL normally preserves duplicates without DISTINCT. A query reading table columns needs an appropriate FROM source. A literal-only query such as SELECT 1 may legally omit FROM.

Study-note section FDB27

> SELECT alone normally reads without changing the database state.

Study-note section FDB27

> Typical written clause order is SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ....

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 11-11

> Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Mock exam 4 · Question 23

Question 17 | Q00222

Which one of the following is a type of Data Manipulation Command?

A. Delete

B. Alter

C. All choices are correct

D. Create

Hint

Classify each SQL command by the kind of change it performs.

Answer A: Delete

DELETE is explicitly DML; ALTER and CREATE are DDL.

Why each choice is right or wrong

A - Correct: DELETE is in DML list (PDF 26, 26.)

B - Incorrect: ALTER is in DDL list (PDF 26, 26.)

C - Incorrect: Other choices include DDL (PDF 26, 26.)

D - Incorrect: CREATE is in DDL list (PDF 26, 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 8-9

> Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME

Fundamental of Database Systems module | PDF page 26 | lines 11-11

> Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Mock exam 4 · Question 24

Question 18 | Q00223

Which one of the following refers to the "data about data"?

A. Warehouse

B. Directory

C. Sub Data

D. Meta Data

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer D: Meta Data

The description of database structure and constraints is metadata.

Why each choice is right or wrong

A - Incorrect: Warehouse is not this term (PDF 5.)

B - Incorrect: Directory is not this term (PDF 5.)

C - Incorrect: Sub Data is not this term (PDF 5.)

D - Correct: Meta-data describes the data structure (PDF 5.)

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 28-29

> description of the database structure and constraints. This information is called meta-data, and it describes the structure of the database.

Question sources

Mock exam 4 · Question 25

Question 19 | Q00298

A language that used to formulate the schema of database structure is_____.

A. Data Manipulation language

B. Data Definition Language

C. Transaction Control Language

D. Data Control Language

Hint

Classify each SQL command by the kind of change it performs.

Answer B: Data Definition Language

DDL supplies definitions, while DML handles data operations.

Why each choice is right or wrong

A - Incorrect: Manipulates contents (PDF 26.)

B - Correct: Defines database structure (PDF 26.)

C - Incorrect: Controls transactions (PDF 26.)

D - Incorrect: Controls privileges (PDF 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 5-6

> has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language).

Question sources

Computer Science model exam · Question 29

University of Gondar model exam · Question 29

Question 20 | Q00299

A table can have only one

A. Unique Key

B. Candidate key

C. Primary key

D. Foreign key

Hint

Test uniqueness and minimality separately when classifying keys.

Answer C: Primary key

One candidate is selected as the primary key; multiple candidates and multiple inter-table references can coexist.

Why each choice is right or wrong

A - Incorrect: Uniqueness need not be the selected identifier (PDF 11, 11.)

B - Incorrect: Several candidates are explicitly allowed (PDF 11, 11.)

C - Correct: Primary key is the selected candidate (PDF 11, 11.)

D - Incorrect: Separate relationships can use separate foreign keys (PDF 11, 11.)

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Study-note section FDB07

> Prefer stable, compact identifiers when practical, but Email, SSN or EmployeeID is a candidate only if business rules guarantee uniqueness.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 10-11

> Primary Key – A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Fundamental of Database Systems module | PDF page 11 | lines 9-10

> An entity set may have more than one candidate key. Primary Key – A primary key is one of the candidate keys chosen by the database designer to

Question sources

Computer Science model exam · Question 30

University of Gondar model exam · Question 30

Question 21 | Q00301

When the values in one or more attributes being used as a foreign key must exist there must be a corresponding primary key value in another table in the database, we have

created

A. Domain Integrity constraint

B. Referential Integrity Constraint

C. Transitive Dependency

D. Entity Integrity Constraints

Hint

Match the validity requirement to the appropriate constraint definition.

Answer B: Referential Integrity Constraint

Matching foreign-key values to existing parent occurrences is the referential-integrity rule.

Why each choice is right or wrong

A - Incorrect: Domain checks legal attribute values (PDF 48.)

B - Correct: Matches the explicit parent-reference rule (PDF 48.)

C - Incorrect: Transitivity is functional dependency (PDF 48.)

D - Incorrect: Entity integrity constrains the primary key itself (PDF 48.)

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB17

> Full dependency requires all determinant attributes; partial dependency holds through a proper subset.

Study-note section FDB17

A prime attribute belongs to some candidate key; a nonprime attribute belongs to none.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 16-17

- Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Computer Science model exam · Question 32

University of Gondar model exam · Question 32

Question 22 | Q00302

Which of the following is not a feature of DBMS?

A. Support ACID Property

B. Single-user Access only

C. High Level of Security

D. Minimum Duplication and Redundancy of Data

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer B: Single-user Access only

Single-user-only contradicts the explicit multiuser DBMS requirement.

Why each choice is right or wrong

A - Incorrect: ACID is a requirement (PDF 6, 66.)

B - Correct: Multiuser support disproves only (PDF 6, 66.)

C - Incorrect: Security is supported (PDF 6, 66.)

D - Incorrect: Normalization aims to reduce redundancy (PDF 6, 66.)

Primary note proof

Study-note section FDB01

A DBMS centralizes metadata and supports multiple views, authorization, transactions, backup and recovery. It controls redundancy rather than guaranteeing zero duplication.

Study-note section FDB37

A transaction groups operations into one logical unit. Atomicity applies all its effects or none. Consistency means a correct transaction preserves declared invariants. Isolation controls interference between concurrent transactions; serializable isolation provides behavior equivalent to some serial execution. Durability preserves committed effects after a crash, typically through durable logging rather than forcing every data page immediately.

Study-note section FDB01

Concurrent access itself is useful; uncontrolled concurrent access causes anomalies.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 7-8

Sharing of data and multiuser transaction processing - A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is

Fundamental of Database Systems module | PDF page 66 | lines 3-4

To ensure data integrity, DBMS should maintain the following ACID properties: • Atomicity: A transaction is an atomic unit of processing; it is either performed in its

Question sources

Computer Science model exam · Question 33

University of Gondar model exam · Question 33

Question 23 | Q00303

Which of the following is not a function of the database?

A. Analyzing code

B. Provide Backup and recovery

C. Manipulating data

D. Managing stored data

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer A: Analyzing code

Data definition, manipulation and recovery are database responsibilities. Arbitrary program-code analysis is not a listed database function; parsing database queries is a narrower task.

Why each choice is right or wrong

A - Correct: General code analysis is outside the stated functions (PDF 5, 97.)

B - Incorrect: Recovery is expressly covered (PDF 5, 97.)

C - Incorrect: Manipulation is expressly covered (PDF 5, 97.)

D - Incorrect: DBMS manages databases (PDF 5, 97.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

> A DBMS centralizes metadata and supports multiple views, authorization, transactions, backup and recovery.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 15-16

> DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server.

Fundamental of Database Systems module | PDF page 97 | lines 4-5

> i. Purpose of Database Recovery ◦ To bring the database into the last consistent state, which existed prior to the

Question sources

Computer Science model exam · Question 34

University of Gondar model exam · Question 34

Question 24 | Q00307

Read each pair left→right. Buildings and locations are 1:1; each employee belongs to one department and a department may have many employees; a hospital serves many patients, each registered at one hospital; authors and books are M:N. Which pair is many-to-one?

A. [Building - Location]

B. [Employee - Department]

C. [Hospital - Patient]

D. [Author - Book]

Hint

Read the relationship in both directions and check participation constraints.

Answer B: B. [Employee - Department]

Many-to-one means several left-side entities may share a right-side entity while each left entity has at most one right partner. Employees share departments under the stated assignment rule. Building→Location is one-to-one, Hospital→Patient is one-to-many and Author→Book is many-to-many. Reading the direction and both maximum counts determines the answer rather than the entity names.

Why each choice is right or wrong

A - Incorrect: Building→Location is explicitly 1:1.

B - Correct: Many employees can share one department; each employee has one department.

C - Incorrect: Hospital→Patient is 1:N under the registration rule.

D - Incorrect: Author→Book is explicitly M:N.

Primary note proof

Study-note section FDB09

> Cardinality specifies maximum associated occurrences: 1:1, 1:N or M:N. Minimum cardinality specifies optional/partial participation (zero permitted) or mandatory/total participation (at least one). A maximum of one does not mean participation is compulsory. Relationship degree counts participating entity roles: unary recursive, binary or ternary.

Study-note section FDB09

> If each employee belongs to one department and each department has many employees, Employee→Department is N:1; Department→Employee is 1:N.

Study-note section FDB09

> Building→Location could also be N:1, so names alone cannot select a unique example.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 29-30

> One to many(1-M)- When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is

Question sources

Computer Science model exam · Question 38

University of Gondar model exam · Question 38

Question 25 | Q00374

_____ is a single elementary unit of information representing an attribute of an entity.

A. Entity Set

B. Record

C. File

D. Field

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer D: Field.

A field corresponds to one attribute rather than the collection of attributes describing a whole record. FDB03 equates attribute, column and field, while tuple, row and record refer to related attribute values together. An entity set groups entities, and a file groups stored records, so both are larger collections.

Why each choice is right or wrong

A - Incorrect: An entity set collects entities of one type rather than representing a single attribute.

B - Incorrect: A record contains related values for multiple attributes, rather than one elementary field.

C - Incorrect: A file groups records and is not an elementary attribute unit.

D - Correct: A field represents one attribute or property within a record.

Primary note proof

Study-note section FDB03

> A relation is a set of tuples, represented as a table. A tuple/row/record contains related attribute values; an attribute/column/field describes a property. Degree counts columns; cardinality counts rows. Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning. Column order has no logical meaning when attributes are identified by name. SQL tables can contain duplicate rows unless constrained.

Study-note section FDB06

> An entity type describes common properties; its current collection of entities is an entity set.

Study-note section FDB06

> In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes.

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 25-26

entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Question sources

Data Structures & Algorithms practice questions · Question 4

Question 26 | Q00548

An ER model has entity types A and B connected by an attribute-free relationship R. Cardinalities are read A→B. Which conditions allow R to be represented by a single B-key foreign-key column in A, permitting NULL for partial participation? Select all that apply.

A. Relationship R is one -to -many and the participation of A in R is Total.

B. Relationship R is one -to -many and the participation of A in R is Partial.

C. Relationship R is many -to -one and the participation of A in R is Total.

D. Relationship R is many --to -one and the participation of A in R is Partial.

Hint

Locate the relationship's cardinality and participation before placing a foreign key.

Answer C, D: C. Relationship R is many -to -one and the participation of A in R is Total.; D. Relationship R is many --to -one and the participation of A in R is Partial.

A single foreign-key column in A can represent R whenever each A is related to at most one B. This is exactly the many-to-one direction. Total participation makes the reference mandatory; partial participation permits NULL. One-to-many in the opposite direction needs the reference on B or another representation, because one A may have multiple related B values.

Why each choice is right or wrong

A - Incorrect: For A→B one-to-many, B normally holds A's key; total participation does not reverse that mapping.

B - Incorrect: Partial participation does not make a one-to-many collection fit a single ordinary foreign key in A.

C - Correct: Many-to-one with total A participation stores a non-NULL B reference in A.

D - Correct: Many-to-one with partial A participation stores a nullable B reference in A.

Primary note proof

Study-note section FDB11

For 1:N, place the one-side key as a foreign key in the many-side table; store relationship attributes there. Department(DeptID) and Employee(EmpID,DeptID,HireDate) implement many employees per department. If participation is total, make the reference NOT NULL. If partial, a nullable reference can represent no assignment.

Study-note section FDB11

For A→B many-to-one, A can absorb the relationship reference under either total or partial participation; requiring total participation as the only possible case overstates the rule.

Study-note section FDB11

Prefer the totally participating side to reduce NULLs; NOT NULL enforces that side's mandatory reference.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 18-19

4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as

Question sources

Computer Science model exam · 2015 · Question 17

Question 27 | Q00567

Under set relational algebra, which listed operation is not commutative? Interpret Join as an inner join, adjust its predicate when swapping inputs, and ignore attribute order.

A. Union

B. Join

C. Intersection

D. None

Hint

Apply each algebra operator to the input relations step by step.

Answer D: D. None

Union and intersection give the same tuple sets when their inputs are exchanged. An inner join also gives corresponding matches after the predicate is adjusted for the swapped inputs; the question explicitly ignores attribute order. Thus none of the three named operations lacks commutativity here. This conclusion depends on the stated inner-join interpretation; preservation of one designated outer side would use a different rule.

Why each choice is right or wrong

A - Incorrect: Union is commutative on union-compatible relations.

B - Incorrect: Inner join is commutative under the stated predicate and schema convention.

C - Incorrect: Intersection is commutative on union-compatible relations.

D - Correct: All three named operators are commutative under these assumptions.

Primary note proof

Study-note section FDB35

> Selection σ filters tuples; projection π retains attributes and removes duplicate tuples under set semantics. Rename ρ changes relation or attribute names. Union, difference and intersection require union-compatible schemas. Difference R−S is not commutative; union and intersection are. Inner joins are commutative up to attribute order when predicates are adjusted; left outer joins are not generally commutative. Identify the join variant before predicting its 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 24 | lines 15-16

> Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 24 | lines 16-17

> Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples). 6.2 Relational Calculus

Question sources

Computer Science model exam · 2015 · Question 36

Question 28 | Q00595

Which of the following is a benefit of normalization rather than a potential cost or disadvantage?

A. May require joins to reconstruct decomposed records

B. May reduce system performance

C. Time consuming

D. Reduce duplicate data

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer D: Reduce duplicate data

Normalization organizes facts according to their dependencies to reduce avoidable duplication and update anomalies. That reduction is a benefit. Decomposing a table may require joins when an application needs the original combined information, which can increase retrieval cost for some workloads. Dependency analysis and schema design also take time. These are possible costs, not claims that normalization always slows a system. Declared dependencies suffice for normalization even when the current sample data shows no errors.

Why each choice is right or wrong

A - Incorrect: A possible cost: reconstructing information from decomposed tables may require joins.

B - Incorrect: A possible cost: additional joins can reduce performance for some workloads, although this is not inevitable.

C - Incorrect: A possible cost: analyzing dependencies and designing a decomposition takes time.

D - Correct: Correct: reducing duplicate facts is a principal benefit, rather than a disadvantage.

Primary note proof

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies. It does not require seeing bad current data: declared dependencies suffice. Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design. Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 18-19

> Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the

Question sources

Computer Science model exam · 2015 · Question 64

Question 29 | Q00617

Let you have two relations A and B, each table has 200 and 100 rows of records respectively. So after applying cross product on those tables, how many rows of records you will have?

A. 300

B. 20000

C. 2000

D. 100

Hint

Apply each algebra operator to the input relations step by step.

Answer B: 20000

The Cartesian-product example explicitly creates 1000*50 tuples from inputs of sizes 1000 and 50 (PDF p.28, lines 6-8). Applying the same all-pairs construction to 200 and 100 rows yields 200*100 = 20000. Each of the 200 rows is paired with each of the 100 rows; no filtering or duplicate-eliminating projection is specified.

Why each choice is right or wrong

A - Incorrect: 300 adds input sizes instead of counting pairs (PDF p.28).

B - Correct: 20000 is 200*100, the illustrated Cartesian pair-count rule (PDF p.28).

C - Incorrect: 2000 is ten times too small for the all-pairs count (PDF p.28).

D - Incorrect: 100 counts only one input and omits its 200 pairings per row (PDF p.28).

Primary note proof

Study-note section FDB35

> A Cartesian product has |R|×|S| tuples and degree degree(R)+degree(S), renaming conflicting attributes if necessary. Thus 200×100=20,000 and 5×7=35. Join equals selection over a product; intersection can be derived from difference. A common minimal basis contains selection, projection, union, difference, product and rename. Assignment is a useful notation, not an indispensable primitive; lists of basic operations vary across textbooks.

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 11-14

> 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.

Fundamental of Database Systems module | PDF page 28 | lines 6-8

> ◦ Requires (1000+50) disk accesses to read from Staff and Branch relations ◦ Creates temporary relation of Cartesian Product (1000*50) tuples ◦ Requires (1000*50) disk access to read in temporary relation and test predicate

Question sources

Computer Science model exam · 2015 · Question 86

Question 30 | Q00675

_____It defines how and where data are organized in physical data storage

A. Internal Schema

B. Conceptual Schema

C. External Schema

D. External and Conceptual Schema

Hint

Trace which schema levels a proposed change affects.

Answer A: Internal Schema

Internal schema specifies physical structure and access paths.

Why each choice is right or wrong

A - Correct: Exact storage schema (PDF 7.)

B - Incorrect: Conceptual hides physical detail (PDF 7.)

C - Incorrect: External is user view (PDF 7.)

D - Incorrect: Neither listed user-level schema defines storage (PDF 7.)

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 28-29

> 1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of

Question sources

Computer Science model exam 2 · Question 16

Question 31 | Q00676

Which database-design stage primarily builds an ER model from the users' requirements?

A. Applications and security

B. Physical refinement

C. Conceptual database

D. Schema refinement

Hint

Identify the design phase from its inputs and intended deliverable.

Answer C: Conceptual database

Conceptual design represents the required entities, attributes and relationships in an ER model. Logical design later maps that conceptual model into relations, keys and constraints and checks normalization. Physical refinement chooses storage organizations and access paths, while application and security work concerns behavior and protection rather than the modeling stage requested. The stem asks for the primary stage building the ER model, not every later activity that might consult it.

Why each choice is right or wrong

A - Incorrect: Incorrect: application and security work addresses behavior and protection, rather than building the conceptual ER model.

B - Incorrect: Incorrect: physical refinement chooses storage organization and access paths.

C - Correct: Correct: conceptual design turns requirements into the ER model.

D - Incorrect: Incorrect: schema refinement checks or improves a logical schema, rather than naming the initial ER-model stage.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 4-5

> High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main

Question sources

Computer Science model exam 2 · Question 17

Question 32 | Q00678

Employee(empid, ...) already exists, has no primary key, and empid values are unique and non-NULL. Which SQL statement adds a primary key on empid?

A. ADD TABLE Employee ADD PRIMARY KEY (empid)

B. ATTACH TABLE Employee ADD PRIMARY KEY (empid)

C. ADD TABLE Employee ATTACH PRIMARY KEY (empid)

D. ALTER TABLE Employee ADD PRIMARY KEY (empid);

Hint

Distinguish table definition changes from changes to stored rows.

Answer D: D. ALTER TABLE Employee ADD PRIMARY KEY (empid);

Adding a constraint to an existing table uses ALTER TABLE, followed by ADD PRIMARY KEY and the parenthesized key column. The existing unique, non-NULL empid values satisfy the prerequisite for the constraint. Option D has this structure. ADD TABLE and ATTACH TABLE do not perform this alteration, and ATTACH PRIMARY KEY is not the required constraint clause.

Why each choice is right or wrong

A - Incorrect: ADD TABLE does not modify an existing table's constraints.

B - Incorrect: ATTACH TABLE is not the required table-alteration syntax.

C - Incorrect: Neither ADD TABLE nor ATTACH PRIMARY KEY forms this SQL alteration.

D - Correct: The corrected ALTER TABLE statement adds the primary-key constraint.

Primary note proof

Study-note section FDB30

> CREATE TABLE Employee (EmpID INTEGER PRIMARY KEY, Name VARCHAR(80)); declares two columns. The name and its type stay together; the comma separates whole column declarations. To add a primary key later: ALTER TABLE Employee ADD PRIMARY KEY (EmpID); existing data must be unique and non-NULL. A table cannot gain a second primary key without replacing the existing constraint.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 8-9

> Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME

Question sources

Computer Science model exam 2 · Question 19

Question 33 | Q00679

While checking normalization process, the problem that doesn't exist on transitive dependency is

A. Modification Anomaly

B. Revoke Anomaly

C. Insertion Anomaly

D. Deletion Anomaly

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer B: Revoke Anomaly

The defined anomalies are insertion, deletion and update; revoke concerns privileges.

Why each choice is right or wrong

A - Incorrect: Modification means update (PDF 16.)

B - Correct: Revoke is not a normalization anomaly (PDF 16.)

C - Incorrect: Insertion is defined (PDF 16.)

D - Incorrect: Deletion is defined (PDF 16.)

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency 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 16 | lines 8-9

data. Types of database anomalies are Insertion anomaly, Deletion anomaly, and Update anomaly. Insertion Anomaly - occurs when the entire primary key is not known and the database cannot

Question sources

Computer Science model exam 2 · Question 20

Question 34 | Q00680

You need to produce a report for mailing labels for customers. The mailing label must have only the customer name and address. The CUSTOMERS table has these columns, ID ,NAME,ADDRESS,PHONE ,the SELECT statement to accomplishes this task?
*

A. SELECT name, address FROM customers;

B. SELECT*FROM customers;

C. SELECT id, name, address, phone FROM customers;

D. SELECT cust_name, cust_address FROM customers;

Hint

Trace how SELECT forms its result columns and rows.

Answer A: SELECT name, address FROM customers;

The required output contains exactly NAME and ADDRESS from the listed schema. FDB27 distinguishes naming specific columns from selecting all columns with *. A names the two available columns explicitly. B and C include unwanted ID and PHONE fields, while D uses column names not supplied by the question.

Why each choice is right or wrong

A - Correct: It selects exactly the two existing columns required for the mailing label.

B - Incorrect: The wildcard also includes ID and PHONE, exceeding the required output.

C - Incorrect: It explicitly includes ID and PHONE in addition to the requested label fields.

D - Incorrect: The supplied schema names NAME and ADDRESS, not CUST_NAME and CUST_ADDRESS.

Primary note proof

Study-note section FDB27

SELECT Name,Address FROM Customer; produces only those mailing-label fields. SELECT * selects all columns; DISTINCT removes duplicate result rows; AS supplies an alias; ALL permits duplicates and does not mean all columns. SQL normally preserves duplicates without DISTINCT. A query reading table columns needs an appropriate FROM source. A literal-only query such as SELECT 1 may legally omit FROM.

Study-note section FDB27

> Typical written clause order is SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ....

Study-note section FDB27

> SELECT alone normally reads without changing the database state.

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 12-13

> Projection (π) - Deletes unwanted columns from relation. Cross-product (x) - Allows us to combine two relations.

Fundamental of Database Systems module | PDF page 26 | lines 11-11

> Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Computer Science model exam 2 · Question 21

Question 35 | Q00681

If the maximum cardinality is equal to 1 in both directions of a relationship, then it is classified as:

A. Optional

B. Many-to-many

C. One-to-one

D. One-to-many

Hint

Read the relationship in both directions and check participation constraints.

Answer C: One-to-one

At most one related entity on each side is the one-to-one case.

Why each choice is right or wrong

A - Incorrect: Optional describes minimum participation (PDF 11.)

B - Incorrect: Many-to-many allows multiple (PDF 11.)

C - Correct: Both maxima match one-to-one (PDF 11.)

D - Incorrect: One-to-many permits multiple on one side (PDF 11.)

Primary note proof

Study-note section FDB09

> Cardinality specifies maximum associated occurrences: 1:1, 1:N or M:N. Minimum cardinality specifies optional/partial participation (zero permitted) or mandatory/total participation (at least one). A maximum of one does not mean participation is compulsory. Relationship degree counts participating entity roles: unary recursive, binary or ternary.

Study-note section FDB09

> Read business rules in both directions.

Study-note section FDB09

> If each employee belongs to one department and each department has many employees, Employee→Department is N:1; Department→Employee is 1:N.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 27-28

> One to one (1-1) - When each entity in each entity set can take part only once in the relationship, the cardinality is one to one.

Question sources

Computer Science model exam 2 · Question 22

Question 36 | Q00682

Which SQL statement sets PRICE to 600 in BOOK for rows whose ID is 'B001'?

A. CHANGE BOOK SET PRICE =600 Where ID ='B001';

B. UPDATE TO CHANGE BOOK SET PRICE =600 Where ID ='B001';

C. UPDATE BOOK SET PRICE =600 Where ID ='B001';

D. GRANT BOOK SET PRICE =600 Where ID ='B001';

Hint

Track which rows each data-changing command affects.

Answer C: UPDATE BOOK SET PRICE =600 Where ID ='B001';

UPDATE changes existing row values. The table name follows UPDATE, SET supplies the assignment and WHERE restricts the rows. C therefore assigns 600 only where ID equals the quoted string B001. CHANGE is not the row-update command, the extra TO CHANGE words break the required UPDATE structure, and GRANT controls privileges rather than performing a row assignment. These malformed alternatives are definite incorrect selections.

Why each choice is right or wrong

A - Incorrect: Incorrect: the required row-change statement begins UPDATE, not CHANGE BOOK SET.

B - Incorrect: Incorrect: UPDATE must be followed by its target table, not TO CHANGE.

C - Correct: Correct: the statement assigns 600 to PRICE for precisely the matching ID rows.

D - Incorrect: Incorrect: GRANT manages privileges; it cannot perform this SET assignment.

Primary note proof

Study-note section FDB29

> UPDATE Book SET Price = 600 WHERE ID = 'B001'; changes matching rows. UPDATE without WHERE targets every row. DELETE FROM Publisher WHERE Name = 'XWZ'; deletes matching publisher rows, subject to reference policies. DELETE FROM r; removes every row but leaves r defined. Both clear relation entries and delete rows describe that effect. Several UPDATE statements can all be valid while performing different tasks; most appropriate requires a requested change and a specified quoting dialect.

Study-note section FDB26

> DCL manages privileges through GRANT and REVOKE.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 5-6

> has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language).

Question sources

Computer Science model exam 2 · Question 23

Question 37 | Q00683

While listing limitation of File Processing System identify the wrong statement?

A. Difficult in accessing data

B. Data isolation and integrity.

C. Data redundancy & inconsistency available.

D. Concurrent access is possible.

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer D: Concurrent access is possible.

The file-system limitation is concurrent-access anomalies; merely allowing concurrent access is not itself the listed limitation.

Why each choice is right or wrong

A - Incorrect: Access difficulty is listed (PDF 5.)

B - Incorrect: Isolation and integrity problems are listed (PDF 5.)

C - Incorrect: Redundancy and inconsistency are listed (PDF 5.)

D - Correct: Possibility of access does not name the anomaly (PDF 5.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

> Concurrent access itself is useful; uncontrolled concurrent access causes anomalies.

Study-note section FDB01

> Traditional file applications define their own formats and access procedures.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 21-22

> Integrity problems, Atomicity problems, Concurrent access anomalies. 1.3 Characteristics of the Database Approach

Question sources

Computer Science model exam 2 · Question 24

Question 38 | Q00684

The Entity Relationship (E-R) Model

A. Entities are represented by circle

B. Attributes are represented by rectangle

C. It represents conceptual view

D. It corresponds to row

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer C: It represents conceptual view

ER depicts the conceptual database schema and entity rectangles.

Why each choice is right or wrong

A - Incorrect: Entity uses rectangle (PDF 9, 10.)

B - Incorrect: Attribute is a property rather than an entity rectangle (PDF 9, 10.)

C - Correct: Conceptual depiction matches ER (PDF 9, 10.)

D - Incorrect: A model is not one tuple (PDF 9, 10.)

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Study-note section FDB05

> Conceptual design turns them into an ER/EER model.

Study-note section FDB05

> Logical design maps the conceptual model to relations, keys and constraints and checks normalization.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 28-29

> ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being

Fundamental of Database Systems module | PDF page 10 | lines 5-6

> or a university course). In ER diagrams, an entity is placed in a rectangle. Each entity has attributes, the particular properties that describe it. Entities that do not have key

Question sources

Computer Science model exam 2 · Question 25

Question 39 | Q00685

Removal of transitivity property should be full filled in

A. First Normal form

B. Fourth Normal form

C. Third Normal form

D. Second Normal form

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer C: Third Normal form

3NF requires nonprime attributes to depend nontransitively on keys after reaching 2NF.

Why each choice is right or wrong

A - Incorrect: 1NF requires atomicity (PDF 18.)

B - Incorrect: 4NF is not the stated rule (PDF 18.)

C - Correct: 3NF explicitly removes transitive dependence (PDF 18.)

D - Incorrect: 2NF removes partial dependence (PDF 18.)

Primary note proof

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key. For Enrollment(SID,CID,SName,CTitle,Grade), assume key (SID,CID), SID→SName and CID→CTitle. Split into Student(SID,SName), Course(CID,CTitle) and Enrollment(SID,CID,Grade). Grade still depends on the whole pair. A relation whose every candidate key is single-attribute has no nonprime partial-key dependency. Higher forms inherit 2NF, so no partial dependencies establishes the minimum guaranteed stage only when 1NF is also assumed.

Study-note section FDB21

> With atomic values the highest form is 1NF, even though a transitive dependency also exists.

Study-note section FDB21

> The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 18 | lines 7-8

> _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science model exam 2 · Question 26

Question 40 | Q00686

If relation R has 5 tuples and relation S has also 7 tuples, then Cartesian product of R and S will have

A. 5+7 tuples

B. 7-5 tuples

C. 5X7 tuples

D. 5/7 tuples

Hint

Apply each algebra operator to the input relations step by step.

Answer C: 5X7 tuples

The module explicitly multiplies input tuple counts for a Cartesian product (PDF p.28, lines 6-8). Hence 5 rows paired with 7 rows give 5*7 = 35 distinct concatenated tuples, before any filtering/projection.

Why each choice is right or wrong

A - Incorrect: 5+7 is the sum, not all pair combinations (PDF p.28).

B - Incorrect: 7-5 is the size difference, not the pair count (PDF p.28).

C - Correct: 5*7 gives the 35 combinations required by the product construction (PDF p.28).

D - Incorrect: 5/7 is not even an integer tuple count and is not pair enumeration (PDF p.28).

Primary note proof

Study-note section FDB35

> A Cartesian product has |R|×|S| tuples and degree degree(R)+degree(S), renaming conflicting attributes if necessary. Thus 200×100=20,000 and 5×7=35. Join equals selection over a product; intersection can be derived from difference. A common minimal basis contains selection, projection, union, difference, product and rename. Assignment is a useful notation, not an indispensable primitive; lists of basic operations vary across textbooks.

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 11-14

> 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.

Fundamental of Database Systems module | PDF page 28 | lines 6-8

◦ Requires (1000+50) disk accesses to read from Staff and Branch relations ◦ Creates temporary relation of Cartesian Product (1000*50) tuples ◦ Requires (1000*50) disk access to read in temporary relation and test predicate

Question sources

Computer Science model exam 2 · Question 27

Question 41 | Q00688

The language associated with a database management system that is employed by end users and programmers to alter data values in the database is the:

A. Data translation language.

B. Data definition language.

C. Data manipulation language.

D. Data presentation language.

Hint

Classify each SQL command by the kind of change it performs.

Answer C: Data manipulation language.

The language for INSERT, UPDATE and DELETE is DML.

Why each choice is right or wrong

A - Incorrect: Not a defined language category (PDF 26.)

B - Incorrect: DDL defines structure (PDF 26.)

C - Correct: DML changes stored data (PDF 26.)

D - Incorrect: Not a defined language category (PDF 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 6-7

> language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language)

Question sources

Computer Science model exam 2 · Question 29

Question 42 | Q00730

Using standard SQL string literals, which update changes only the name of the student whose ID is '123' to 'abel'?

A. UPDATE student ID='123';

B. UPDATE student SET ID='123';

C. UPDATE student SET name='abel' WHERE ID='123';

D. UPDATE student SET name='abel';

Hint

Track which rows each data-changing command affects.

Answer C: C. UPDATE student SET name='abel' WHERE ID='123';

UPDATE needs SET to state the assignment and WHERE to restrict the target rows. C assigns the desired name only to the student with ID '123'. B changes IDs in every row, while D changes every student's name because it has no filter. A omits SET and is malformed. Correct syntax and the requested scope both matter.

Why each choice is right or wrong

A - Incorrect: SET is missing before the assignment.

B - Incorrect: This sets every student's ID and does not change the name.

C - Correct: SET changes name and WHERE restricts the operation to ID '123'.

D - Incorrect: Without WHERE, every name is changed.

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB29

> UPDATE Book SET Price = 600 WHERE ID = 'B001'; changes matching rows.

Study-note section FDB28

> Use single quotes for SQL string literals: WHERE ID = 'B001'.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 5-6

> has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language).

Question sources

Computer Science model exam 2 · Question 71

Question 43 | Q00899

A weak entity is identified by its partial key together with the key of an independently keyed associated entity. Which listed terms describe that associated entity set? Select all that apply.

A. Identifying set

B. Strong entity set

C. Neighbor set

D. Owner set

Hint

Check how a dependent entity obtains its complete identity.

Answer B, D: B. Strong entity set; D. Owner set

A weak entity cannot identify itself using its own attributes alone. Its owner contributes the missing key component, and in the specified case that owner has an independent key, so it is a strong entity set. Both owner set and strong entity set describe it. The identifying relationship connects the sets; neighbor does not identify the required ownership role.

Why each choice is right or wrong

A - Incorrect: Identifying normally names the owner relationship rather than the defined owner entity-set role.

B - Correct: The owner supplying the independent key is normally a strong entity set.

C - Incorrect: Neighbor does not express the ownership and identification needed for a weak entity.

D - Correct: Owner set names the entity set whose key contributes to weak-entity identification.

Primary note proof

Study-note section FDB08

> A weak entity lacks a complete independent key. Its owner is normally a strong entity, so owner entity set and strong entity set may both describe the associated set. Its partial key, also called discriminator, distinguishes weak entities of the same owner. Identification combines the owner key and partial key.

Study-note section FDB08

> The weak entity participates totally in its identifying relationship.

Study-note section FDB08

> Chen notation uses a double rectangle, double identifying diamond and dashed underline for a partial key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 6-10

> Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities in combination with one of their attribute values. A weak entity type has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity.

Fundamental of Database Systems module | PDF page 12 | lines 10-13

> 2. Choose one of the key attributes of E as the primary key for R. For each weak entity type W with owner entity type E (which is a strong entity), create a relation R & include all attributes of ‗W‘ as attributes of R. Also, include the primary key attribute of the relation that correspond to the owner entity type as foreign key attributes of R .

Question sources

Computer Science Exit Exam · 2018 · Question 12

Question 44 | Q00901

A language that allows the DBA or user to describe and name the entities, attributes, and relationships required for the application is

A. Transaction control language (TCL)

B. Data control language (DCL)

C. .
Data Manipulation Language (DML)

D. Data Definition Language (DDL)

Hint

Classify each SQL command by the kind of change it performs.

Answer D: Data Definition Language (DDL)

Defining schema objects belongs to DDL.

Why each choice is right or wrong

A - Incorrect: Transaction control is a different task (PDF 26.)

B - Incorrect: Privileges are a different task (PDF 26.)

C - Incorrect: Manipulation operates on contents (PDF 26.)

D - Correct: DDL supplies definitions (PDF 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 5-6

> has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language).

Question sources

Computer Science Exit Exam · 2018 · Question 14

Question 45 | Q00907

In relational-table terminology, which listed terms name a unit of related field values treated together by an application? Select all that apply.

A. Rows

B. Field

C. Record

D. Database

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer A, C: A. Rows; C. Record

A relational tuple groups related attribute values into one unit. Row and record are names for that same unit, so both options qualify even though one is plural. A field refers to one property rather than the whole grouping. A database is the larger coherent collection containing tables and their data, rather than one individual tuple.

Why each choice is right or wrong

A - Correct: A relational row groups related field values and is synonymous with a record.

B - Incorrect: A field represents one attribute, not the collection of related fields in the definition.

C - Correct: A record is the unit containing related fields described by the stem.

D - Incorrect: A database contains a larger coherent collection, rather than this single field grouping.

Primary note proof

Study-note section FDB03

> A relation is a set of tuples, represented as a table. A tuple/row/record contains related attribute values; an attribute/column/field describes a property. Degree counts columns; cardinality counts rows. Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning. Column order has no logical meaning when attributes are identified by name. SQL tables can contain duplicate rows unless constrained.

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application.

Study-note section FDB01

> It controls redundancy rather than guaranteeing zero duplication.

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 25-26

entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Question sources

Computer Science Exit Exam · 2018 · Question 20

Question 46 | Q00927

Which template correctly creates a table with one column? Treat table_name, column_name and column_type as placeholders for valid identifiers and a data type.

A. CREATE table_name (column_name, column type);

B. CREATE table_name (column_type, column name);

C. CREATE TABLE table_name (column_name column_type);

D. CREATE TABLE table_name (column_type, column_name);

Hint

Distinguish table definition changes from changes to stored rows.

Answer C: C. CREATE TABLE table_name (column_name column_type);

A column declaration places its name immediately before its data type. Commas separate complete declarations when more than one column is present. The corrected C uses CREATE TABLE and a properly paired name and type. A and B omit TABLE, while D separates and reverses the name/type parts rather than declaring the intended column.

Why each choice is right or wrong

A - Incorrect: TABLE is missing and the name/type pair is split by a comma.

B - Incorrect: TABLE is missing and the type/name order is wrong.

C - Correct: CREATE TABLE is followed by one correctly ordered name/type declaration.

D - Incorrect: A comma separates type from name and no valid intended declaration is formed.

Primary note proof

Study-note section FDB30

> CREATE TABLE Employee (EmpID INTEGER PRIMARY KEY, Name VARCHAR(80)); declares two columns. The name and its type stay together; the comma separates whole column declarations. To add a primary key later: ALTER TABLE Employee ADD PRIMARY KEY (EmpID); existing data must be unique and non-NULL. A table cannot gain a second primary key without replacing the existing constraint.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 49 | lines 19-20

> - CREATE TABLE table-name (attr2 attr-type NOT NULL, forbids NULL values

Question sources

Computer Science Exit Exam · 2018 · Question 40

Question 47 | Q00932

From the following alternatives which one is FALSE about database system?

A. " I is very difficult to protect a file under the system

B. * iy database system the user is not required to write the procedures

C. « tn database system, DBMS provides a good protection mechanism.

D. ° i contains a wide variety of sophisticated techniques to store and retrieve the data.

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer A: " I is very difficult to protect a file under the system

DBMS permission and authentication support contradicts the claim of poor protection; B is interpreted as nonprocedural querying rather than no application code ever.

Why each choice is right or wrong

A - Correct: Contradicts described protection (PDF 53.)

B - Incorrect: Nonprocedural retrieval is supported (PDF 53.)

C - Incorrect: Matches security support (PDF 53.)

D - Incorrect: Storage and indexes support retrieval (PDF 53.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB25

> B/B+ trees support ordered and range search; hashing targets equality; bitmap indexes suit some analytic workloads.

Study-note section FDB01

> Complexity, administration, storage and licensing can outweigh benefits for a tiny fixed application; lightweight embedded DBMSs also exist.

Related reading (context, not primary note proof)

Study-note section FDB36

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 53 | lines 15-16

- DBMS can provide some security ◦ Each user has an account, username and password

Question sources

Computer Science Exit Exam · 2018 · Question 45

Question 48 | Q00943

Consider a directed line(->) from the relationship set advisor to both entity sets instructor and student. This indicates cardinality

A. Many to one

B. Many to many

C. ¢.
One to one

D. One to many

Hint

Read the relationship in both directions and check participation constraints.

Answer C: One-to-one, under the common relationship-arrow convention.

FDB09 explicitly states the convention used for this example: arrows from advisor toward both Instructor and Student impose maximum-one at both ends. Each instructor can therefore be associated with at most one student and each student with at most one instructor. That is 1:1, without implying mandatory participation. A different notation would need an explicit legend.

Why each choice is right or wrong

A - Incorrect: Many-to-one constrains only one end to maximum one, whereas both ends have arrows here.

B - Incorrect: Many-to-many permits multiple associations at both ends, contrary to both maximum-one arrows.

C - Correct: Maximum one at each end yields a one-to-one relationship under the stated common convention.

D - Incorrect: One-to-many allows multiple associations at one end, which the second arrow rules out.

Primary note proof

Study-note section FDB09

Read business rules in both directions. If each employee belongs to one department and each department has many employees, Employee→Department is N:1; Department→Employee is 1:N. Building→Location could also be N:1, so names alone cannot select a unique example. Under the common arrow convention, arrows from advisor to both Instructor and Student impose maximum-one at both ends, giving 1:1. Without a notation convention, arrows are unsafe evidence.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 27-28

> One to one (1-1) - When each entity in each entity set can take part only once in the relationship, the cardinality is one to one.

Question sources

Computer Science Exit Exam · 2018 · Question 56

Question 49 | Q00957

What is the basic goal of Normalization?

A. To decrease data integrity

B. To reduce redundancy

C. ¢.
To maximize transitive dependency

D. To increase dependency

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer B: To reduce redundancy

Normalization reduces repeated data and resulting anomalies.

Why each choice is right or wrong

A - Incorrect: Integrity is protected (PDF 16.)

B - Correct: Redundancy reduction is explicit (PDF 16.)

C - Incorrect: 3NF removes transitive dependence (PDF 16.)

D - Incorrect: Increasing dependency is not the goal (PDF 16.)

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Related reading (context, not primary note proof)

Study-note section FDB21

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 26-27

> database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship.

Question sources

Computer Science Exit Exam · 2018 · Question 70

Question 50 | Q00964

For a relation with attributes ID, CITY and NAME, ID is guaranteed unique and non-NULL; CITY and NAME are not unique. Which listed attribute sets are superkeys? Select all that apply.

A. NAME

B. ID

C. CITY

D. CITY, ID

Hint

Test uniqueness and minimality separately when classifying keys.

Answer B, D: B. ID; D. CITY, ID

A superkey is any attribute set guaranteed to identify a tuple; it need not be minimal. ID has the stated uniqueness guarantee, and adding CITY cannot destroy it. Thus ID and (CITY, ID) are superkeys. CITY and NAME individually lack uniqueness guarantees. Only ID is a minimal candidate key among the two qualifying choices, but minimality is not required here.

Why each choice is right or wrong

A - Incorrect: Repeated names are permitted, so NAME is not a guaranteed identifier.

B - Correct: ID is guaranteed unique and identifies each tuple.

C - Incorrect: Repeated cities are permitted, so CITY alone is not a superkey.

D - Correct: The set includes unique ID and therefore retains uniqueness.

Primary note proof

Study-note section FDB07

A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

If ID is unique, both {ID} and {ID,City} are superkeys, but only {ID} is minimal.

Study-note section FDB07

Prefer stable, compact identifiers when practical, but Email, SSN or EmployeeID is a candidate only if business rules guarantee uniqueness.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-8

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key.

Question sources

Computer Science Exit Exam · 2018 · Question 77

Question 51 | Q00982

is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

A. Super key

B. Candidate key

C. Social security key

D. Primary key

Hint

Test uniqueness and minimality separately when classifying keys.

Answer D: Primary key

A primary key is the candidate selected by the designer.

Why each choice is right or wrong

A - Incorrect: Superkey need not be minimal (PDF 11.)

B - Incorrect: Candidate need not be selected (PDF 11.)

C - Incorrect: Example identifier is not this role (PDF 11.)

D - Correct: Exact selected-key definition (PDF 11.)

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 10-11

> Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Computer Science Exit Exam · 2018 · Question 95

Question 52 | Q01151

12.
is a collection of concepts thatcan be used to describe the structure of a database.

A. a | Data Architecture

B. i? | Data Model

C. oO Data client

D. | All of the above

Hint

Compare the data models by how they organize records and relationships.

Answer B: i? | Data Model

The defined collection of concepts describing database structure is a data model.

Why each choice is right or wrong

A - Incorrect: Architecture is not the defined term (PDF 7.)

B - Correct: Exact definition (PDF 7.)

C - Incorrect: Client is an application role (PDF 7.)

D - Incorrect: Other choices do not match (PDF 7.)

Primary note proof

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one 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 7 | lines 4-5

> A data model a collection of concepts that can be used to describe the structure of a database and provides the necessary means to achieve data abstraction. By structure of a database we mean the

Question sources

Computer Science Exit Exam practice questions · Question 12

Question 53 | Q01152

The description of a database is called

A. a | Database snapshot

B. i? | Database instances

C. @) Database schema

D. | All of the above

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer C: @) Database schema

Schema describes the designed database; instance and snapshot describe current data.

Why each choice is right or wrong

A - Incorrect: Current contents (PDF 7.)

B - Incorrect: Current occurrences (PDF 7.)

C - Correct: Designed structure (PDF 7.)

D - Incorrect: Other choices are different concepts (PDF 7.)

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 21-22

> The description of a database is called the database schema, which is specified during database design and is not expected to change frequently. The data in the database at a particular moment

Question sources

Computer Science Exit Exam practice questions · Question 13

Question 54 | Q01153

Which listed terms denote database contents at one particular moment, rather than the database definition? Select all that apply.

A. Database instance

B. Database state / snapshot

C. Database schema

D. None of the above

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer A, B: A. Database instance; B. Database state / snapshot

An instance, state or snapshot describes the actual data at a selected time. Those contents can change after an insertion or update without changing the database definition. A schema describes names, structures and constraints instead. Therefore both the instance option and the state/snapshot option fit, while none of the above cannot be selected because these matching terms are present.

Why each choice is right or wrong

A - Correct: A database instance denotes its data at a particular time.

B - Correct: State and snapshot are synonyms for the same time-specific database contents.

C - Incorrect: A schema describes structure and constraints rather than current data values.

D - Incorrect: Both instance and state/snapshot are provided and correctly fit the definition.

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 22-23

> design and is not expected to change frequently. The data in the database at a particular moment of time is called a database state or snapshot. It is also called the current set of occurrences or

Question sources

Computer Science Exit Exam practice questions · Question 14

Question 55 | Q01154

What is the capacity to change the conceptual schema while preserving affected external schemas and application interfaces?

A. Logical data independence

B. Physical data independence

C. Conceptual database

D. All of the above

Hint

Trace which schema levels a proposed change affects.

Answer A: Logical data independence

Logical data independence allows changes to the conceptual schema while preserving the external views and application interfaces that depend on it. Physical data independence instead concerns changing internal storage without changing the conceptual schema. A conceptual database or schema describes the logical structure itself; naming that structure does not express the ability to preserve external interfaces during a change. Consequently A names the requested property and the all-of-the-above choice is false.

Why each choice is right or wrong

A - Correct: Correct: logical data independence preserves external interfaces across conceptual-schema changes.

B - Incorrect: Incorrect: physical data independence concerns internal-storage changes while preserving the conceptual schema.

C - Incorrect: Incorrect: conceptual database names the logical structure, not the property preserving external interfaces when that structure changes.

D - Incorrect: Incorrect: physical independence and the conceptual database are not the requested change-preservation property.

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints.

Study-note section FDB04

> Physical data independence permits changing internal storage without changing the conceptual schema, for example adding an index. Logical data independence permits conceptual changes while preserving affected external views and application interfaces, for example splitting a table while maintaining a compatible view.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 8 | lines 12-13

1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.

Question sources

Computer Science Exit Exam practice questions · Question 15

Question 56 | Q01155

is language used to define the database conceptual schema

A. a | Data Manipulation Language (DML) x

B. ? | Data Definition Langugae (DDL) v

C. /@| Transaction Query Language (TQL) x

D. | None of the above x

Hint

Classify each SQL command by the kind of change it performs.

Answer B: ? | Data Definition Langugae (DDL) v

Defining a schema is definition rather than manipulation. DDL commands define structure, while DML commands manipulate data.

Why each choice is right or wrong

A - Incorrect: DML covers retrieval and updates, not schema definition. (PDF pp. 26.)

B - Correct: DDL is the definition-language category. (PDF pp. 26.)

C - Incorrect: TQL is not the module's schema language. (PDF pp. 26.)

D - Incorrect: The definition-language choice is present. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Computer Science Exit Exam practice questions · Question 16

Question 57 | Q01157

Common functions applied to collections of numeric values: SUM, AVERAGE, COUNT. This function belongs to

A. Ww Grouping x

B. i? | Aggregate functions 4

C. |@| All of the above x

D. | None of the above ><

Hint

Apply grouping before filtering groups and computing aggregate output.

Answer B: Aggregate functions.

SUM, average and COUNT summarize collections into values, which is aggregation. GROUP BY instead partitions rows into groups for those summaries. SQL normally spells the average function AVG; the conceptual use of AVERAGE here still describes aggregation.

Why each choice is right or wrong

A - Incorrect: Grouping partitions rows; it is not the function category containing SUM and COUNT.

B - Correct: These functions summarize a collection of values into a count, total or average.

C - Incorrect: Grouping and aggregate functions are distinct concepts, so all listed categories cannot be correct.

D - Incorrect: Aggregate functions is present and correctly describes the functions in the stem.

Primary note proof

Study-note section FDB32

> COUNT(*) counts rows, COUNT(x) counts non-NULL x values; SUM and AVG summarize non-NULL numeric values. SQL's usual average function is AVG, not AVERAGE. GROUP BY creates one group per distinct grouping value; HAVING tests aggregate groups after WHERE has filtered individual rows.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Computer Science Exit Exam practice questions · Question 18

Question 58 | Q01158

Entity types that do not have key attributes of their own are called

A. a | Entity

B. i? | Strong Entity

C. @) Weak Entity

D. | Foreign Entity

Hint

Check how a dependent entity obtains its complete identity.

Answer C: @) Weak Entity

Weak entities lack their own key attributes, while strong entities have their own keys. Their partial keys identify them only within an owner.

Why each choice is right or wrong

A - Incorrect: Entity is the broader category and does not identify the missing-key subtype. (PDF pp. 10.)

B - Incorrect: Strong entities have their own key, the opposite condition. (PDF pp. 10.)

C - Correct: Weak entity matches the exact definition. (PDF pp. 10.)

D - Incorrect: Foreign entity is not the defined subtype; foreign keys concern table references. (PDF pp. 10.)

Primary note proof

Study-note section FDB08

> A weak entity lacks a complete independent key. Its owner is normally a strong entity, so owner entity set and strong entity set may both describe the associated set. Its partial key, also called discriminator, distinguishes weak entities of the same owner. Identification combines the owner key and partial key.

Study-note section FDB08

> A weak entity is not merely any table with a foreign key.

Study-note section FDB08

> Then (EmployeeID,DependentName) is the key, and EmployeeID is a foreign key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 6-10

> Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities in combination with one of their attribute values. A weak entity type has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity.

Question sources

Computer Science Exit Exam practice questions · Question 19

Question 59 | Q01159

The partial key is sometimes called

A. a | Dominant Entity

B. i? | Child Entity Type

C. @) Subordinate Entity Type

D. | Discriminator

Hint

Check how a dependent entity obtains its complete identity.

Answer D: Discriminator.

A partial key distinguishes weak entities belonging to the same owner; the owner key is still needed for global identification. FDB08 explicitly calls that partial key a discriminator. The other choices describe entity roles or types rather than the identifying attribute.

Why each choice is right or wrong

A - Incorrect: An entity role is not the name of the identifying partial-key attribute.

B - Incorrect: A child entity type describes a type of entity rather than its partial identifier.

C - Incorrect: A subordinate entity type is not a synonym for a partial key.

D - Correct: Discriminator is the partial key that distinguishes weak entities within the same owner.

Primary note proof

Study-note section FDB08

> A weak entity lacks a complete independent key. Its owner is normally a strong entity, so owner entity set and strong entity set may both describe the associated set. Its partial key, also called discriminator, distinguishes weak entities of the same owner. Identification combines the owner key and partial key.

Study-note section FDB08

The weak entity participates totally in its identifying relationship.

Study-note section FDB08

Chen notation uses a double rectangle, double identifying diamond and dashed underline for a partial key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 6-13

Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities in combination with one of their attribute values. A weak entity type has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity. In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.

Question sources

Computer Science Exit Exam practice questions · Question 20

Question 60 | Q01162

A real-world physical or logical object is called:

A. Attributes

B. Relationship

C. Entity

D. All of the above

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer C: Entity

An entity may have physical existence or conceptual existence. Attributes describe an entity; relationships associate entity types. The stem asks for the object itself.

Why each choice is right or wrong

A - Incorrect: An attribute is a property of the object. (PDF pp. 10, 11.)

B - Incorrect: A relationship associates objects rather than naming the object. (PDF pp. 10, 11.)

C - Correct: Entity covers both physical and conceptual objects. (PDF pp. 10, 11.)

D - Incorrect: The other categories have different roles, so all is false. (PDF pp. 10, 11.)

Primary note proof

Study-note section FDB06

An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 3-8

An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). In ER diagrams, an entity is placed in a rectangle. Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities

Fundamental of Database Systems module | PDF page 11 | lines 13-15

A relationship type represents the association between entity types. For example, _Enrolled in‘ is a relationship type that exists between entity Student and Course. In ER diagram, relationship is represented by a diamond and connecting the entities with lines.

Question sources

Computer Science Exit Exam practice questions · Question 23

Question 61 | Q01163

In the usual normalization sequence, which is the first normal form that excludes partial dependencies of nonprime attributes on any candidate key?

A. First Normal Form (1NF)

B. Third Normal Form (3NF)

C. Second Normal Form (2NF)

D. Boyce-Codd Normal Form (BCNF)

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer C: Second Normal Form (2NF)

Second normal form is the first stage requiring atomic relations to have no nonprime attribute partially dependent on any candidate key. First normal form addresses atomic values, so it does not impose that dependency rule. Third normal form and BCNF are stronger stages that inherit the absence of partial dependencies, but they are not the first stage introducing it. BCNF specifically requires every nontrivial functional-dependency determinant to be a superkey.

Why each choice is right or wrong

A - Incorrect: Incorrect: 1NF requires atomic values but does not exclude partial dependencies.

B - Incorrect: Incorrect: 3NF inherits 2NF and imposes its additional determinant/prime-attribute condition; it is not the first stage.

C - Correct: Correct: 2NF is the first stage excluding nonprime partial dependencies on candidate keys.

D - Incorrect: Incorrect: BCNF requires every nontrivial determinant to be a superkey; it is stronger than the first stage imposing the partial-dependency rule.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent

attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science Exit Exam practice questions · Question 24

Question 62 | Q01164

In the usual normalization sequence, which is the first normal form that excludes transitive dependence of nonprime attributes on a key through a nonkey determinant?

A. Third Normal Form (3NF)

B. Fourth Normal Form (4NF)

C. First Normal Form (1NF)

D. All of the above

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer A: Third Normal Form (3NF)

In the stated single-key teaching case, 2NF excludes partial dependencies but can still allow a nonprime fact to depend on the key through a nonkey determinant. The next defining stage is 3NF, summarized by the whole-key-and-nothing-but-the-key mnemonic. BCNF strengthens the determinant rule, and 4NF addresses multivalued dependencies. Stronger forms inherit earlier conditions, so the explicit first-stage scope makes 3NF the unique selection rather than treating inherited properties as rival answers.

Why each choice is right or wrong

A - Correct: Correct: 3NF is the first stage imposing the specified nonprime transitive-dependency restriction in this teaching case.

B - Incorrect: Incorrect: 4NF addresses multivalued dependencies and is a later stage; it is not the first stage imposing the 3NF condition.

C - Incorrect: Incorrect: 1NF concerns atomic values and does not eliminate the specified transitive dependencies.

D - Incorrect: Incorrect: Not all options identify the requested first stage: 1NF does not eliminate the specified transitive dependency and 4NF is later.

Primary note proof

Study-note section FDB20

1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Study-note section FDB21

4NF addresses nontrivial multivalued dependencies with nonsuperkey determinants; 5NF addresses join dependencies.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-9

First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF).

Fundamental of Database Systems module | PDF page 18 | lines 5-8

Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science Exit Exam practice questions · Question 25

Question 63 | Q01217

A conceptual level refers to:

A. Describes the represents global view of the entire database

B. Describes the part of the database that a particular user group views of the data environment

C. Describes how data is actually stored on disk

D. Describes the physical storage structure of the database

Hint

Trace which schema levels a proposed change affects.

Answer A: Describes the represents global view of the entire database

The conceptual schema describes the whole database while hiding storage details. External views show a particular group's part, and internal schema describes storage.

Why each choice is right or wrong

A - Correct: Whole-database structure is conceptual. (PDF pp. 7, 8.)

B - Incorrect: A user-group subset is external. (PDF pp. 7, 8.)

C - Incorrect: Actual storage is internal. (PDF pp. 7, 8.)

D - Incorrect: Physical storage structure is internal. (PDF pp. 7, 8.)

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 25-30

> 2.3 Three-schema Architecture and Data Independence The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels: 1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Fundamental of Database Systems module | PDF page 8 | lines 2-7

> 2. The conceptual level has a conceptual schema, which describes the structure of the whole database for users. The conceptual schema hides the details of physical storage structures and

concentrates on describing entities, data types, relationships, user operations, and constraints. 3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Question sources

Computer Science model exam 2 · Question 38

Question 64 | Q01226

Suppose you want to design a database for a certain company after you know the need of all users of the database. What is the next step you are expected to do?

A. Logical design

B. Normalization

C. Query Processing

D. Conceptual design

Hint

Identify the design phase from its inputs and intended deliverable.

Answer D: Conceptual design

After collecting users' information requirements, conceptual design develops the ER/EER model that represents those needs. Logical design then maps that model to relations, keys and constraints and checks normalization. Normalization refines dependency structure rather than replacing the initial conceptual modeling step. Query processing evaluates database requests instead of defining the initial enterprise model. D is therefore the next step under the stated sequence, rather than a claim that design can never iterate.

Why each choice is right or wrong

A - Incorrect: Incorrect: logical design maps an already developed conceptual model into relations and constraints.

B - Incorrect: Incorrect: normalization refines relation dependencies during logical design; the initial conceptual model must first be developed.

C - Incorrect: Incorrect: query processing evaluates requests rather than creating the initial model from requirements.

D - Correct: Correct: conceptual design follows the collection of requirements in the stated sequence.

Primary note proof

Study-note section FDB05

First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Related reading (context, not primary note proof)

Study-note section FDB19

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 10-26

Requirement Analysis The most important step in implementing a database system is to find out what is needed i.e what type of a database is required for the business organization, daily volume of data, how much data needs to be stored in the master files etc. In order to collect all this information, a database analyst spends a lot of time within the business organization talking to people, end users and getting acquainted with the day-to-day process. Conceptual Database Design In this phase the database designers will make a decision on the database model that perfectly suits the organization‘s requirement. The database designers will study the documents prepared by the analysis in the requirement analysis stage and then start development of a system model that fulfils the needs. Logical database design (Data model mapping) The conceptual design is translated into internal model which includes mapping of all objects i.e design of tables, indexes, views, transaction, access privileges etc. Physical Design Physical design or Database implementation needs the formation of special storage related constructs. These constructs consist of storage groups, table spaces, data files, tables etc.

Question sources

Computer Science model exam 2 · Question 47

Question 65 | Q01278

All are true about the Database except:

A. Program data independence.

B. Self-describing nature

C. Sharing of data and multi-user transaction processing

D. Support of single views of the data

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer D: Support of single views of the data

The listed characteristics include insulation between data and programs, self-description, sharing, and multiple views. The exception is single views, interpreted as support restricted to one view; a DBMS can of course also display a single view.

Why each choice is right or wrong

A - Incorrect: Program/data insulation is expressly a characteristic. (PDF pp. 5, 6.)

B - Incorrect: Self-description is expressly listed. (PDF pp. 5, 6.)

C - Incorrect: Sharing and multiuser transactions are expressly listed. (PDF pp. 5, 6.)

D - Correct: The stated characteristic is multiple views, rather than restriction to a single view. (PDF pp. 5, 6.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

> A DBMS centralizes metadata and supports multiple views, authorization, transactions, backup and recovery.

Study-note section FDB04

> The external level gives each user group a selected view.

Related reading (context, not primary note proof)

Study-note section FDB03

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 22-29

> 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.
> Self-describing nature - A fundamental characteristic of the database approach is that the database system contains not only the database itself (data) but also a complete definition or description of the database structure and constraints. This information is called meta-data, and it describes the structure of the database.

Fundamental of Database Systems module | PDF page 6 | lines 2-9

> Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps

in achieving data independence which is used to enhance the security of data. Support of multiple views of the data - A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Sharing of data and multiuser transaction processing - A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database.

Question sources

Computer Science model exam 2 · Question 99

Question 66 | Q01326

Which one of the following provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database?

A. DML(Data Manipulation Langauge)

B. DDL(Data Definition Langauge)

C. Query

D. Relational Schema

Hint

Classify each SQL command by the kind of change it performs.

Answer A: DML(Data Manipulation Langauge)

DML includes SELECT, INSERT, DELETE, UPDATE and therefore covers both retrieval and tuple changes. The second list is misprinted as Basic DDL commands but sits under the DML heading; the heading and separation resolve that typo.

Why each choice is right or wrong

A - Correct: DML is the category covering all named operations. (PDF pp. 26.)

B - Incorrect: DDL describes structural commands. (PDF pp. 26.)

C - Incorrect: A query is a request, not the complete language category for all modifications. (PDF pp. 26.)

D - Incorrect: A relational schema describes structure rather than performing operations. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 1

Question 67 | Q01327

Which command is used to modifies an existing database object, such as a table?

A. Create

B. Alter

C. Drop

D. Modify

Hint

Distinguish table definition changes from changes to stored rows.

Answer B: ALTER.

ALTER changes the definition of an existing database object. FDB30 demonstrates adding or removing columns and adding constraints with ALTER TABLE. CREATE establishes an object and DROP removes it. MODIFY can occur within some vendor-specific alteration syntax, but it is not the general top-level command requested here.

Why each choice is right or wrong

A - Incorrect: CREATE defines a new object rather than altering the definition of an existing one.

B - Correct: ALTER is the command used to change an existing database object definition.

C - Incorrect: DROP removes the object instead of modifying its existing definition.

D - Incorrect: MODIFY is not the general standalone command; some dialects use it inside ALTER syntax.

Primary note proof

Study-note section FDB30

> ALTER TABLE Employee ADD COLUMN Email VARCHAR(120); adds a column in PostgreSQL-style syntax. ALTER TABLE Employee DROP COLUMN Email; removes it. ALTER TABLE Employee ADD

CONSTRAINT uq_email UNIQUE (Email); adds a named constraint. Use ALTER rather than DALTER, and do not separate column_name and column_type with a comma; validate the resulting SQL before execution.

Study-note section FDB26

DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications.

Study-note section FDB26

Some texts separate SELECT as DQL.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 2

Question 68 | Q01328

Which command is used to modifies records?

A. Insert

B. Update

C. Delete

D. Modify

Hint

Track which rows each data-changing command affects.

Answer B: Update

The module puts UPDATE in DML (PDF p.26). Its UPDATE(Salary) privilege example explicitly permits the Finance user to change Salary values (PDF pp.54-55), grounding UPDATE as the command for modifying existing record values.

Why each choice is right or wrong

A - Incorrect: INSERT is a separate DML operation, illustrated as adding VALUES rows on PDF p.77.

B - Correct: UPDATE is explicitly tied to changing Salary values on PDF p.55.

C - Incorrect: DELETE is a separate operation illustrated with a row predicate on PDF p.77.

D - Incorrect: MODIFY is not the command used for the demonstrated change-value privilege on PDF pp.54-55.

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB29

> UPDATE without WHERE targets every row.

Study-note section FDB29

> DELETE FROM r; removes every row but leaves r defined.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 10-11

> 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 54 | lines 16-18

> • <privileges> is a list of SELECT <columns>, INSERT <columns>, DELETE, and UPDATE <columns>, or simply ALL

Fundamental of Database Systems module | PDF page 55 | lines 2-5

> UPDATE(Salary) ON Employee TO Finance The user _Finance‘ can view the entire Employee table, and can change Salary values, but cannot change other values or pass on their privilege

Fundamental of Database Systems module | PDF page 77 | lines 14-24

> Example 1: Commit Transaction BEGIN TRANSACTION; DELETE FROM HumanResources.JobCandidate WHERE JobCandidateID = 13; COMMIT; Example 2: Rollback Transaction CREATE TABLE ValueTable (id int); BEGIN TRANSACTION; INSERT INTO ValueTable VALUES(1); INSERT INTO ValueTable VALUES(2); ROLLBACK;

Question sources

Database Fundamentals practice questions · Question 3

Question 69 | Q01329

In the SQLite command-line shell, which command lists the names and files of attached databases?

A. db

B. use

C. .databases

D. database

Hint

Classify each SQL command by the kind of change it performs.

Answer C: C. .databases

The command belongs to the SQLite shell rather than ordinary SQL statement syntax. Its documented spelling is .databases, including the leading dot and final s, and it lists attached database names and files. The corrected C has that spelling. The alternatives db, use and database do not form this SQLite attached-database listing command.

Why each choice is right or wrong

A - Incorrect: db is not the SQLite shell listing command.

B - Incorrect: use does not list SQLite attached names and files.

C - Correct: The .databases command performs the requested listing.

D - Incorrect: database lacks the required shell-command spelling.

Primary note proof

Study-note section FDB26

> DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 4

Question 70 | Q01330

What is an RDBMS?

A. Database that stores data elements that are not linked

B. Database that accesses data elements that are not linked

C. Database that stores and allows access to data elements that are linked

D. None of the mentioned

Hint

Compare the data models by how they organize records and relationships.

Answer C: Database that stores and allows access to data elements that are linked

A relational DBMS manages a database represented by relations/tables, with references maintaining relationships between tuples. Among these broad descriptions C is closest, although linked data alone is not a complete definition distinguishing relational from other models.

Why each choice is right or wrong

A - Incorrect: A description excluding links omits the relationships modeled by references. (PDF pp. 5, 13, 14.)

B - Incorrect: Accessing only unlinked elements also omits those relationships. (PDF pp. 5, 13, 14.)

C - Correct: Storage and access to related elements agrees with tables and relational constraints. (PDF pp. 5, 13, 14.)

D - Incorrect: C supplies the compatible broad description, though it is not a formal definition. (PDF pp. 5, 13, 14.)

Primary note proof

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one query.

Study-note section FDB03

> A relation is a set of tuples, represented as a table.

Study-note section FDB03

Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning.

Related reading (context, not primary note proof)

Study-note section FDB15

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-16

Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server.

Fundamental of Database Systems module | PDF page 13 | lines 18-27

3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Fundamental of Database Systems module | PDF page 14 | lines 21-26

Referential Integrity Constraints - The Referential integrity constraints is specified between two relations or tables and used to maintain the consistency among the tuples in two relations. Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Question sources

Database Fundamentals practice questions · Question 5

Question 71 | Q01331

What is a relation in RDBMS?

A. Key

B. Table

C. Row

D. Data Types

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer B: Table

The relational model represents relations in table format. A tuple is one row; a key identifies tuples rather than being the relation itself.

Why each choice is right or wrong

A - Incorrect: A key is an identifying attribute set. (PDF pp. 13.)

B - Correct: A relation is represented as a table. (PDF pp. 13.)

C - Incorrect: A row is a tuple within the relation. (PDF pp. 13.)

D - Incorrect: Data types constrain values rather than constituting a relation. (PDF pp. 13.)

Primary note proof

Study-note section FDB03

> A relation is a set of tuples, represented as a table. A tuple/row/record contains related attribute values; an attribute/column/field describes a property. Degree counts columns; cardinality counts rows. Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning. Column order has no logical meaning when attributes are identified by name. SQL tables can contain duplicate rows unless constrained.

Study-note section FDB07

> A superkey uniquely identifies a tuple.

Study-note section FDB07

> A primary key is one selected candidate key; remaining candidates are alternate keys.

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 18-27

> 3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Database Fundamentals practice questions · Question 6

Question 72 | Q01332

Relational Algebra is a _____ query language that takes two relations as input and produces another relation as an output of the query.

A. Relational

B. Structural

C. Procedural

D. Fundamental

Hint

Distinguish a declarative condition from a prescribed evaluation procedure.

Answer C: Procedural

Relational algebra is procedural because it specifies a sequence of operations. The stem overgeneralizes arity: operations can take one or more input relations, not always two.

Why each choice is right or wrong

A - Incorrect: Relational describes the model but misses the procedural/calculus distinction. (PDF pp. 24, 24.)

B - Incorrect: Structural is not the stated classification. (PDF pp. 24, 24.)

C - Correct: Procedural is expressly justified by the required sequence. (PDF pp. 24, 24.)

D - Incorrect: Fundamental is not the procedural classification used in the definition. (PDF pp. 24, 24.)

Primary note proof

Study-note section FDB36

> Relational algebra specifies a sequence of operations and is procedural at the logical level. Relational calculus specifies a predicate describing desired results and is declarative/nonprocedural. A tuple variable ranges over tuples; a domain variable ranges over individual attribute values. Safe/range-restricted expressions produce results bounded by the database's active domain.

Study-note section FDB35

> Rename ρ changes relation or attribute 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 24 | lines 5-9

> Relational algebra is the basic set of operations for the relational model. These operations enable a user to specify basic retrieval requests (or queries). The result of an operation is a new relation, which may have been formed from one or more input relations. The algebra operations thus

produce new relations. Since each operation returns a relation, operations can be composed (Algebra is —closed‖.)

Fundamental of Database Systems module | PDF page 24 | lines 21-26

In a calculus expression, there is no order of operations to specify how to retrieve the query result. A calculus expression specifies only what information the result should contain. This is the main distinguishing feature between relational algebra and relational calculus. Relational calculus is considered to be a nonprocedural language. This differs from relational algebra, where we must write a sequence of operations to specify a retrieval request; hence relational algebra can be considered as a procedural way of stating a query.

Question sources

Database Fundamentals practice questions · Question 7

Question 73 | Q01333

Which of the following is a fundamental operation in relational algebra?

A. Set intersection

B. Natural join

C. Assignment

D. None of the mentioned

Hint

Use the operator convention stated in the question and distinguish membership from derivability.

Answer A: Set intersection

Under the stated Basic operations list, intersection is basic. A smaller primitive basis may instead treat intersection as derived; the answer uses the stated list.

Why each choice is right or wrong

A - Correct: Intersection is explicitly under Basic operations. (PDF pp. 24.)

B - Incorrect: Natural join is absent from this basic list. (PDF pp. 24.)

C - Incorrect: Assignment is absent from this basic list. (PDF pp. 24.)

D - Incorrect: False because intersection is listed as basic. (PDF pp. 24.)

Primary note proof

Study-note section FDB-S01

A list of basic relational-algebra operators is a teaching convention, distinct from a claim that every listed operation is logically indispensable. The course module uses selection, projection, Cartesian product, set difference, union and intersection as its basic-operation list. Natural join and assignment are not members of that particular list.

Study-note section FDB-S01

A smaller primitive basis may therefore omit intersection even when it is introduced as a basic operation.

Study-note section FDB-S01

Under set semantics, union includes tuples from either compatible input, intersection retains tuples common to both, and difference keeps first-input tuples absent from the second.

Related reading (context, not primary note proof)

Study-note section FDB35

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 10-16

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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Question sources

Database Fundamentals practice questions · Question 8

Question 74 | Q01334

Which of the following is used to denote the selection operation in relational algebra?

A. Pi (Greek)

B. Sigma (Greek)

C. Lambda (Greek)

D. Omega (Greek)

Hint

Apply each algebra operator to the input relations step by step.

Answer B: Sigma (Greek)

Selection filters tuples and is conventionally denoted by sigma, σ. Projection keeps attributes and is denoted by pi, π. Lambda and omega are different Greek symbols and therefore cannot identify the requested sigma operation under this notation. This is a question about the operator's conventional symbol, rather than whether another symbol could be assigned a new meaning in a custom notation. B is the unique selection symbol among the choices.

Why each choice is right or wrong

A - Incorrect: Incorrect: pi, π, denotes projection, which retains attributes.

B - Correct: Correct: sigma, σ, denotes selection, which filters tuples.

C - Incorrect: Incorrect: lambda, λ, is a different symbol from the conventional selection symbol σ.

D - Incorrect: Incorrect: omega, ω, is a different symbol from the conventional selection symbol σ.

Primary note proof

Study-note section FDB35

> Selection σ filters tuples; projection π retains attributes and removes duplicate tuples under set semantics.

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Question sources

Database Fundamentals practice questions · Question 9

Question 75 | Q01335

For union-compatible relations, which listed names denote the minus operation R−S, returning tuples in R but not S? Select all that apply.

A. Union

B. Set-difference

C. Difference

D. Intersection

Hint

Use the operator convention stated in the question and distinguish membership from derivability.

Answer B, C: B. Set-difference; C. Difference

R−S tests membership asymmetrically: a tuple must occur in R and must not occur in S. Difference and set-difference are names for this same operator, so both should be selected. Union combines membership from either input, whereas intersection requires membership in both. Swapping the inputs to difference can change the result because the exclusion direction matters.

Why each choice is right or wrong

A - Incorrect: Union combines compatible input tuples rather than retaining only first-input tuples absent from the second.

B - Correct: Set-difference returns tuples in the first relation that do not occur in the second.

C - Correct: Difference is the same minus operator as set-difference.

D - Incorrect: Intersection keeps tuples common to both relations, the opposite membership condition.

Primary note proof

Study-note section FDB-S01

> Under set semantics, union includes tuples from either compatible input, intersection retains tuples common to both, and difference keeps first-input tuples absent from the second. Intersection can be expressed using difference, while a matching join can be expressed as selection over a Cartesian product. A smaller primitive basis may therefore omit intersection even when it is introduced as a basic operation.

Study-note section FDB35

> Union, difference and intersection require union-compatible schemas.

Study-note section FDB35

> Rename ρ changes relation or attribute 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 24 | lines 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Question sources

Database Fundamentals practice questions · Question 10

Question 76 | Q01336

Which one of the following is used to define the structure of the relation, deleting relations and relating schemas?

A. DML(Data Manipulation Langauge)

B. DDL(Data Definition Langauge)

C. Query

D. Relational Schema

Hint

Classify each SQL command by the kind of change it performs.

Answer B: DDL(Data Definition Langauge)

Defining and deleting relation structures are schema operations in DDL. DML operates on their data. Relating schemas here is interpreted as defining schema structure rather than retrieving joins.

Why each choice is right or wrong

A - Incorrect: DML is for data operations. (PDF pp. 26.)

B - Correct: DDL is the schema-definition category. (PDF pp. 26.)

C - Incorrect: A query request is not the schema language category. (PDF pp. 26.)

D - Incorrect: A schema is a description, not a language for changing it. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 11

Question 77 | Q01337

CREATE TABLE employee (name VARCHAR(80), id INTEGER); What kind of SQL statement is this?

A. DML

B. DDL

C. View

D. Integrity constraint

Hint

Classify each SQL command by the kind of change it performs.

Answer B: DDL

CREATE TABLE defines a new table and its columns, so it is a data-definition-language statement. DML instead retrieves or changes stored rows. A view is a query-based logical object, while this statement creates a base table. Integrity constraints can be included inside a table definition, but this whole statement is not merely a constraint expression. VARCHAR has the stated length in this table definition.

Why each choice is right or wrong

A - Incorrect: Incorrect: DML acts on stored data, whereas this statement defines a table.

B - Correct: Correct: creating a table and declaring its columns is DDL.

C - Incorrect: Incorrect: CREATE TABLE defines a table rather than a view.

D - Incorrect: Incorrect: the statement defines an entire table, not just an integrity-constraint expression.

Primary note proof

Study-note section FDB26

DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Study-note section FDB30

CREATE TABLE Employee (EmpID INTEGER PRIMARY KEY, Name VARCHAR(80)); declares two columns. The name and its type stay together; the comma separates whole column declarations.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 12

Question 78 | Q01338

SELECT * FROM employee
What type of statement is this?

A. DML

B. DDL

C. View

D. Integrity constraint

Hint

Classify each SQL command by the kind of change it performs.

Answer A: DML

SELECT is in the list under the DML heading. The line's repeated DDL word is a typographical error in that DML subsection, as shown by the preceding separate DDL list.

Why each choice is right or wrong

A - Correct: SELECT is classified under DML. (PDF pp. 26.)

B - Incorrect: SELECT is not in the preceding DDL list. (PDF pp. 26.)

C - Incorrect: No view is defined by this standalone retrieval. (PDF pp. 26.)

D - Incorrect: The retrieval is not an integrity constraint. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Study-note section FDB33

CREATE VIEW CSStudent AS SELECT StudentID,Name FROM Student WHERE Major = 'CS'; stores a query definition and presents a virtual table.

Study-note section FDB33

An ordinary view does not necessarily store its result; a materialized view does.

Related reading (context, not primary note proof)

Study-note section FDB27

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 13

Question 79 | Q01339

The basic data type char(n) is a _____ length character string and varchar(n) is _____ length character.

A. Fixed, equal

B. Equal, variable

C. Fixed, variable

D. Variable, equal

Hint

Relate the value domain and access requirements to storage and type choices.

Answer C: Fixed, variable

The attribute-type section directly defines CHAR as fixed-length and VARCHAR as variable-length. Neither blank means equal length.

Why each choice is right or wrong

A - Incorrect: Fixed is right for CHAR but equal is wrong for VARCHAR. (PDF pp. 19.)

B - Incorrect: Variable is right for VARCHAR but equal is not the stated CHAR class. (PDF pp. 19.)

C - Correct: Fixed followed by variable matches both definitions. (PDF pp. 19.)

D - Incorrect: Both descriptions miss the given definitions. (PDF pp. 19.)

Primary note proof

Study-note section FDB23

> CHAR(n) is fixed length; VARCHAR(n) has variable length up to a limit. DATE represents a date and BLOB binary content such as images. INTEGER represents whole numbers; DECIMAL(p,s) represents exact decimal values, making it suitable for money when scale is specified. FLOAT/REAL/DOUBLE are approximate; names, widths and precision depend on the DBMS. Numeric type widths depend on the DBMS and type; do not assume one universal width.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 19 | lines 11-15

> 5.2 Choosing Attribute Data Types CHAR - fixed-length character VARCHAR() - variable-length characters DATE - actual date BLOB - binary large object (good for graphics images, audio, sound clips, etc.)

Question sources

Database Fundamentals practice questions · Question 14

Question 80 | Q01340

To remove a relation from an SQL database, we use the _____ comman
d.

A. Delete

B. Purge

C. Remove

D. Drop table

Hint

Classify each SQL command by the kind of change it performs.

Answer D: DROP TABLE.

Removing a relation means removing the table object, rather than just its tuples. FDB26 distinguishes DROP TABLE from DELETE: DELETE preserves the table definition while removing rows. DROP TABLE removes the object itself. PURGE and REMOVE are not the general SQL command specified here for this task.

Why each choice is right or wrong

A - Incorrect: DELETE removes tuples while preserving the relation definition, so it does not remove the relation.

B - Incorrect: PURGE is not the general table-removal command established here.

C - Incorrect: REMOVE is not the SQL command used here to remove a relation object.

D - Correct: DROP TABLE removes the table object, including its relation definition.

Primary note proof

Study-note section FDB26

> DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 15

Question 81 | Q01341

Assume constraints permit the deletion. Which descriptions apply to executing DELETE FROM r; on an existing table r? Select all that apply.

A. Remove relation

B. Clear relation entries

C. Delete fields

D. Delete rows

Hint

Track which rows each data-changing command affects.

Answer B, D: B. Clear relation entries; D. Delete rows

Without WHERE, DELETE targets every row in r. If the operation succeeds, the table's entries are cleared but its column definitions and table object remain. Clear relation entries and delete rows describe this same effect. Removing the relation itself requires a schema operation such as DROP TABLE, while deleting fields would change columns rather than row contents.

Why each choice is right or wrong

A - Incorrect: The table definition remains; DELETE does not remove the relation object.

B - Correct: Without WHERE, every relation entry is removed while the table remains available.

C - Incorrect: Fields are columns; DELETE removes rows rather than column definitions.

D - Correct: Every row is targeted because the statement contains no filtering WHERE clause.

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB29

> Both clear relation entries and delete rows describe that effect.

Study-note section FDB29

> DELETE FROM r; removes every row but leaves r defined.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 10-11

> 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 16

Question 82 | Q01342

INSERT INTO instructor VALUES (10211, 'Smith', 'Biology', 66000);
What type of statement is this?

A. Query

B. DML

C. Relational

D. DDL

Hint

Classify each SQL command by the kind of change it performs.

Answer B: DML

INSERT belongs to the DML list. Classification follows from the operation name; it does not require assuming an instructor schema or proving that these particular values satisfy its constraints.

Why each choice is right or wrong

A - Incorrect: Query is a generic request rather than the displayed language category. (PDF pp. 26.)

B - Correct: INSERT is expressly a DML command. (PDF pp. 26.)

C - Incorrect: Relational names a model, not this statement class. (PDF pp. 26.)

D - Incorrect: INSERT is not in the DDL list. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Study-note section FDB29

> Without a column list, values must follow the table's declared column order.

Study-note section FDB29

INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 17

Question 83 | Q01343

Which joins discard unmatched tuples from both inputs? Interpret Natural join as an ordinary natural inner join. Select all that apply.

A. Left outer join

B. Right outer join

C. Inner join

D. Natural join

Hint

Compute matching pairs, then account for any preserved unmatched rows.

Answer C, D: C. Inner join; D. Natural join

An inner join returns only pairs that satisfy its matching condition, so unmatched tuples from either input disappear. An ordinary natural join is an inner equality join using all common column names and has the same preservation behavior. Left and right outer joins instead preserve unmatched tuples from their designated side by adding NULLs for the other side.

Why each choice is right or wrong

A - Incorrect: A left outer join preserves unmatched left rows, filling right-side columns with NULL.

B - Incorrect: A right outer join preserves unmatched right rows, filling left-side columns with NULL.

C - Correct: An inner join returns matching pairs only, so unmatched tuples do not survive.

D - Correct: An ordinary natural join is an inner equality join and discards unmatched tuples.

Primary note proof

Study-note section FDB31

An inner join returns matching pairs satisfying ON. An equijoin uses equality; a theta join permits another comparison. A natural join is ordinarily an inner equality join on every same-named column and merges those duplicate columns. JOIN ... USING (DeptID) specifies shared join columns explicitly. CROSS JOIN forms every pair without a matching condition.

Study-note section FDB31

LEFT OUTER JOIN also preserves unmatched left rows, padding right columns with NULL; RIGHT preserves right rows and FULL preserves both.

Study-note section FDB31

Therefore inner and natural inner joins both discard unmatched rows.

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 10-16

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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 28 | lines 17-24

Query 2 (Better) (position=„Manager“)^(city=„London“) (Staff Staff.branchNo = Branch.branchNo Branch) - Again requires (1000+50) disk accesses to read from Staff and Branch - Joins Staff and Branch on branchNo with 1000 tuples (1 employee : 1 branch) - Requires (1000) disk access to read in joined relation and check predicate Total Work = (1000+50) + 2*(1000) = 3050 I/O operations

Question sources

Database Fundamentals practice questions · Question 18

Question 84 | Q01344

What type of join is needed when you wish to include rows that do not have matching values?

A. Equi-join

B. Natural join

C. Outer join

D. All of the mentioned

Hint

Compute matching pairs, then account for any preserved unmatched rows.

Answer C: Outer join.

An outer join retains unmatched rows from one or both inputs and fills the missing side with NULL. FDB31 contrasts that behavior with ordinary matching-only joins. Equijoin describes an equality condition, and an ordinary natural join is an inner equality join; neither alone guarantees unmatched-row preservation. Therefore all of the mentioned is false.

Why each choice is right or wrong

A - Incorrect: An equality condition alone does not retain rows that fail to match.

B - Incorrect: An ordinary natural join uses matching shared-column values and discards unmatched rows.

C - Correct: Outer joins preserve unmatched rows from the left, right or both inputs.

D - Incorrect: The other named joins do not all guarantee preservation of unmatched rows.

Primary note proof

Study-note section FDB31

> An inner join returns matching pairs satisfying ON. An equijoin uses equality; a theta join permits another comparison. A natural join is ordinarily an inner equality join on every same-named column and merges those duplicate columns. JOIN ... USING (DeptID) specifies shared join columns explicitly. CROSS JOIN forms every pair without a matching condition.

Study-note section FDB31

> Therefore inner and natural inner joins both discard unmatched rows.

Study-note section FDB31

> LEFT OUTER JOIN also preserves unmatched left rows, padding right columns with NULL; RIGHT preserves right rows and FULL preserves both.

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 28 | lines 17-24

> Query 2 (Better) (position=„Manager“)^(city=„London“) (Staff Staff.branchNo = Branch.branchNo Branch) - Again requires (1000+50) disk accesses to read from Staff and Branch - Joins Staff and Branch on branchNo with 1000 tuples (1 employee : 1 branch) - Requires (1000) disk access to

read in joined relation and check predicate Total Work = (1000+50) + 2*(1000) = 3050 I/O operations

Question sources

Database Fundamentals practice questions · Question 19

Question 85 | Q01345

How many input relations does one ordinary binary join operator combine? Count operator inputs, not distinct base tables used by a complete query.

A. One

B. Two

C. Three

D. All of the mentioned

Hint

Compute matching pairs, then account for any preserved unmatched rows.

Answer B: B. Two

One binary join operator combines two input relations. Either input may itself be the output of an earlier join, so the complete query can involve more base tables. A self-join can refer twice to one base table, but still supplies two relation occurrences to the operator. The question fixes the count at the operator level, making two the unique choice.

Why each choice is right or wrong

A - Incorrect: One distinct base table can support a self-join, but the operator still has two inputs.

B - Correct: A binary join operator has exactly two inputs.

C - Incorrect: Three base tables require a chain of binary joins, not one operator.

D - Incorrect: The input count of the specified binary operator is fixed, so all counts do not apply.

Primary note proof

Study-note section FDB31

An inner join returns matching pairs satisfying ON. An equijoin uses equality; a theta join permits another comparison. A natural join is ordinarily an inner equality join on every same-named column and merges those duplicate columns. JOIN ... USING (DeptID) specifies shared join columns explicitly. CROSS JOIN forms every pair without a matching condition.

Study-note section FDB31

Each ordinary join operator combines two inputs, but its inputs can be earlier joins, so one query can join many tables.

Study-note section FDB31

Distinguish base-table count, table occurrences and binary operator arity.

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 5-9

Relational algebra is the basic set of operations for the relational model. These operations enable a user to specify basic retrieval requests (or queries). The result of an operation is a new relation, which may have been formed from one or more input relations. The algebra operations thus produce new relations. Since each operation returns a relation, operations can be composed (Algebra is —closed‖.)

Fundamental of Database Systems module | PDF page 28 | lines 17-24

Query 2 (Better) (position=„Manager“)^(city=„London“) (Staff Staff.branchNo = Branch.branchNo Branch) - Again requires (1000+50) disk accesses to read from Staff and Branch - Joins Staff and Branch on branchNo with 1000 tuples (1 employee : 1 branch) - Requires (1000) disk access to read in joined relation and check predicate Total Work = (1000+50) + 2*(1000) = 3050 I/O operations

Question sources

Database Fundamentals practice questions · Question 20

Question 86 | Q01346

Which option gives the complete set of valid SQL join forms among the listed alternatives? Choose the most complete single answer.

A. Cross join

B. Natural join

C. Join with USING clause

D. All of the mentioned

Hint

Compute matching pairs, then account for any preserved unmatched rows.

Answer D: All of the mentioned

CROSS JOIN forms every pair of input rows without a matching predicate. NATURAL JOIN matches all same-named columns and merges the duplicate join columns. JOIN USING explicitly names shared equality columns. All three are valid join forms, although they do not all express a join condition in the same way. The revised stem requests the complete single answer, so the aggregate D is selected; each individually true alternative is incomplete on its own.

Why each choice is right or wrong

A - Incorrect: Incorrect as the complete selection: CROSS JOIN is a valid join form without a matching condition, but B and C also qualify.

B - Incorrect: Incorrect as the complete selection: NATURAL JOIN is valid and matches same-named columns, but A and C also qualify.

C - Incorrect: Incorrect as the complete selection: JOIN USING is valid and names shared equality columns, but A and B also qualify.

D - Correct: Correct: all three named constructs are valid SQL join forms, so the aggregate is the complete answer.

Primary note proof

Study-note section FDB31

> An inner join returns matching pairs satisfying ON. An equijoin uses equality; a theta join permits another comparison. A natural join is ordinarily an inner equality join on every same-named column and merges those duplicate columns. JOIN ... USING (DeptID) specifies shared join columns explicitly. CROSS JOIN forms every pair without a matching condition.

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 28 | lines 17-24

> Query 2 (Better) (position=„Manager“)^(city=„London“) (Staff Staff.branchNo = Branch.branchNo Branch) - Again requires (1000+50) disk accesses to read from Staff and Branch - Joins Staff and Branch on branchNo with 1000 tuples (1 employee : 1 branch) - Requires (1000) disk access to read in joined relation and check predicate Total Work = (1000+50) + 2*(1000) = 3050 I/O operations

Question sources

Database Fundamentals practice questions · Question 21

Question 87 | Q01347

The operation which is not considered a basic operation of relational algebra is

A. Join

B. Selection

C. Union

D. Cross product

Hint

Use the operator convention stated in the question and distinguish membership from derivability.

Answer A: Join

The stated basic-operation list contains selection, union, cross-product, projection, difference, and intersection. Join is not included, even though it appears in later execution-plan examples.

Why each choice is right or wrong

A - Correct: Join is absent from the basic list and is the requested exception. (PDF pp. 24.)

B - Incorrect: Selection is explicitly basic. (PDF pp. 24.)

C - Incorrect: Union is explicitly basic. (PDF pp. 24.)

D - Incorrect: Cross-product is explicitly basic. (PDF pp. 24.)

Primary note proof

Study-note section FDB-S01

> A list of basic relational-algebra operators is a teaching convention, distinct from a claim that every listed operation is logically indispensable. The course module uses selection, projection, Cartesian product, set difference, union and intersection as its basic-operation list. Natural join and assignment are not members of that particular list.

Study-note section FDB-S01

> A smaller primitive basis may therefore omit intersection even when it is introduced as a basic operation.

Study-note section FDB-S01

> Intersection can be expressed using difference, while a matching join can be expressed as selection over a Cartesian product.

Related reading (context, not primary note proof)

Study-note section FDB35

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Question sources

Database Fundamentals practice questions · Question 22

Question 88 | Q01348

A many to many relationship between two entities usually results in how many tables?

A. One

B. Two

C. Three

D. Four

Hint

Identify relationship degree and cardinality before choosing a relational mapping.

Answer C: Three tables: two entity tables and one association table.

Each of the two entity types normally has its own relation. Their many-to-many relationship requires an additional association relation holding references to both entities and any relationship attributes. FDB12 illustrates Student, Course and Enrollment. Thus the usual total is three, rather than only the one new association table.

Why each choice is right or wrong

A - Incorrect: One counts only the association table and omits the two participating entity tables.

B - Incorrect: Two entity tables alone do not record arbitrary many-to-many pairs.

C - Correct: The two entity tables plus their association table produce the usual total of three.

D - Incorrect: No fourth table is required by the ordinary binary many-to-many mapping.

Primary note proof

Study-note section FDB12

> A binary M:N relationship needs an association table in addition to the two entity tables. Enrollment(StudentID,CourseID,Grade) references Student and Course; (StudentID,CourseID) is a key if one enrollment per pair is allowed. Repeated attempts require Term or AttemptNo in the key. This ordinarily produces three tables.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 2-3

> Many to many (M-M) - When entities in all entity sets can take part more than once in the relationship cardinality is many to many.

Fundamental of Database Systems module | PDF page 12 | lines 8-24

> 3.4 Mapping ER-models to relational tables 1. For each strong entity type E, create a relation R that includes all the attributes of E. 2. Choose one of the key attributes of E as the primary key for R. For each weak entity type W with owner entity type E (which is a strong entity), create a relation R & include all attributes of ‗W‘ as attributes of R. Also, include the primary key attribute

of the relation that correspond to the owner entity type as foreign key attributes of R . 3. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Foreign Key approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. 4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as foreign key in S that represents the other entity type participating in R. Include any attributes of the 1:N relation type as attributes of S. 5. For each n-ary relationship type R, where n>2, create a new relationship S to represent R. Include the primary keys of the relations that represent the participating entity types as foreign key attributes in S. Also include any attributes of the n-ary relationship type as attributes of S.

Question sources

Database Fundamentals practice questions · Question 23

Question 89 | Q01349

An _____ is a set of entities of the same type that share the same properties, or attributes.

A. Entity set

B. Attribute set

C. Relation set

D. Entity model

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer A: Entity set.

FDB06 distinguishes an individual entity from the type describing its properties and the set containing the current entities of that type. The stem asks for that collection of similarly described entities, which is an entity set. Attributes describe properties; a model describes representation rather than the current collection itself.

Why each choice is right or wrong

A - Correct: An entity set is the collection of entities belonging to one common entity type.

B - Incorrect: An attribute set groups properties, rather than the entities possessing those properties.

C - Incorrect: Relation set is not the ER term for the stated collection of entities.

D - Incorrect: An entity model describes structures, rather than naming the current set of entity instances.

Primary note proof

Study-note section FDB06

An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Study-note section FDB03

A relation is a set of tuples, represented as a table.

Study-note section FDB03

A tuple/row/record contains related attribute values; an attribute/column/field describes a property.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 2-9

An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes. An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely. Key Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key.

Fundamental of Database Systems module | PDF page 11 | lines 16-18

Degree of a relationship The number of different entity sets participating in a relationship set is called as degree of a relationship set.

Question sources

Database Fundamentals practice questions · Question 24

Question 90 | Q01350

Entity is a _____

A. Object of relation

B. Present working model

C. Thing in real world

D. Model of relation

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer C: Thing in real world

An entity is an object with physical or conceptual existence. It need not be a physical thing; the real-world thing option includes conceptual objects such as a course.

Why each choice is right or wrong

A - Incorrect: Object of relation is not the module's definition. (PDF pp. 10.)

B - Incorrect: A present working model is not an entity definition. (PDF pp. 10.)

C - Correct: Thing in the real world matches physical or conceptual objects. (PDF pp. 10.)

D - Incorrect: A model describes entities rather than being the object described. (PDF pp. 10.)

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 3-8

> An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). In ER diagrams, an entity is placed in a rectangle. Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities

Question sources

Database Fundamentals practice questions · Question 25

Question 91 | Q01351

E-R modeling technique is a :

A. Top-down approach

B. Bottom-up approach

C. Left-right approach

D. None of the above

Hint

Identify the design phase from its inputs and intended deliverable.

Answer A: Top-down is the usual conceptual ER approach.

FDB05 describes conceptual design as usually starting with an enterprise view and working downward into entity and relationship details. This is the usual top-down classification of ER modeling. Bottom-up refinement is associated here with normalization from dependencies. This is a useful conventional approach, rather than a claim that every ER modeling activity must follow one direction.

Why each choice is right or wrong

A - Correct: Conceptual ER modeling usually moves from an overall enterprise view toward detailed entities and relationships.

B - Incorrect: Bottom-up dependency refinement describes normalization here, rather than the usual ER conceptual approach.

C - Incorrect: Left-right is not a database design methodology identified here.

D - Incorrect: Top-down is an available and appropriate description of the usual conceptual ER approach.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Study-note section FDB05

> Conceptual design usually proceeds from a whole enterprise view downward; normalization can refine structures from dependencies upward.

Study-note section FDB05

> These are useful approaches rather than mandatory directions for every modeling activity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 3-6

> 3.1 Database Modeling High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships.

Fundamental of Database Systems module | PDF page 9 | lines 27-31

> 3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Question sources

Database Fundamentals practice questions · Question 26

Question 92 | Q01352

The number of entities to which another entity can be associated via a relationship set is expressed as :

A. Entity

B. Cardinality

C. Schema

D. Attributes

Hint

Read the relationship in both directions and check participation constraints.

Answer B: Cardinality

Cardinality measures how often an entity participates and distinguishes one-to-one, one-to-many, and many-to-many. That is the number of possible associations asked about.

Why each choice is right or wrong

A - Incorrect: Entity is a participating object. (PDF pp. 11, 12.)

B - Correct: Cardinality is the association-count constraint. (PDF pp. 11, 12.)

C - Incorrect: Schema describes structure rather than the count. (PDF pp. 11, 12.)

D - Incorrect: Attributes are properties, not the association-count classification. (PDF pp. 11, 12.)

Primary note proof

Study-note section FDB09

Cardinality specifies maximum associated occurrences: 1:1, 1:N or M:N. Minimum cardinality specifies optional/partial participation (zero permitted) or mandatory/total participation (at least one). A maximum of one does not mean participation is compulsory. Relationship degree counts participating entity roles: unary recursive, binary or ternary.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 24-31

Cardinality The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types: One to one (1-1) - When each entity in each entity set can take part only once in the relationship, the cardinality is one to one. One to many(1-M)- When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.

Fundamental of Database Systems module | PDF page 12 | lines 2-7

Many to many (M-M) - When entities in all entity sets can take part more than once in the relationship cardinality is many to many. The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the minimum number of relationship instances that each entity can participate in and is sometimes called the minimum cardinality constraint. There are two types of participation constraints, total and partial.

Question sources

Database Fundamentals practice questions · Question 27

Question 93 | Q01353

Relations produced from an E-R model will always be in :

A. First normal form

B. Second normal form

C. Third normal form

D. Fourth normal form

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer A: First normal form, assuming correct ER-to-relational mapping.

FDB20 states that properly splitting composite and multivalued ER structures produces ordinary 1NF relations. Nothing in an ER drawing alone proves absence of partial, transitive or multivalued dependencies needed for higher forms. Thus A is the guaranteed level under correct relational mapping; the word always must not be read as excusing an incorrect mapping.

Why each choice is right or wrong

A - Correct: Correctly mapped atomic relations are in 1NF; this is the guaranteed level described here.

B - Incorrect: An ER model alone does not rule out partial dependencies on a composite candidate key.

C - Incorrect: An ER model alone does not establish the full dependency conditions required for 3NF.

D - Incorrect: An ER model alone does not rule out the nontrivial multivalued dependencies addressed by 4NF.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells. It does not mean merely declaring a primary key. A mapped ER design that correctly splits composite and multivalued structures gives ordinary 1NF tables; ER drawing alone guarantees no higher form.

Study-note section FDB20

> Higher forms inherit 2NF, so no partial dependencies establishes the minimum guaranteed stage only when 1NF is also assumed.

Study-note section FDB20

> A relation whose every candidate key is single-attribute has no nonprime partial-key dependency.

Related reading (context, not primary note proof)

Study-note section FDB21

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 8-24

> 3.4 Mapping ER-models to relational tables 1. For each strong entity type E, create a relation R that includes all the attributes of E. 2. Choose one of the key attributes of E as the primary key for R. For each weak entity type W with owner entity type E (which is a strong entity), create a relation R & include all attributes of ‗W‘ as attributes of R. Also, include the primary key attribute of the relation that correspond to the owner entity type as foreign key attributes of R . 3. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Foreign Key approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. 4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as foreign key in S that represents the other entity type participating in R. Include any attributes of the 1:N relation type as attributes of S. 5. For each

n-ary relationship type R, where n>2, create a new relationship S to represent R. Include the primary keys of the relations that represent the participating entity types as foreign key attributes in S. Also include any attributes of the n-ary relationship type as attributes of S.

Fundamental of Database Systems module | PDF page 17 | lines 2-15

First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Database Fundamentals practice questions · Question 28

Question 94 | Q01354

In ER model the details of the entities are hidden from the user. This process is called :

A. generalization

B. specialization

C. abstraction

D. none of these above

Hint

Trace which schema levels a proposed change affects.

Answer C: abstraction

Abstraction hides unwanted or irrelevant details. Generalization groups subclasses; specialization divides a superclass. Thus hiding entity details describes abstraction.

Why each choice is right or wrong

A - Incorrect: Generalization combines common characteristics, rather than the general detail-hiding process. (PDF pp. 6, 13.)

B - Incorrect: Specialization separates narrower classes. (PDF pp. 6, 13.)

C - Correct: Abstraction is explicitly the process of hiding details. (PDF pp. 6, 13.)

D - Incorrect: The required term is present. (PDF pp. 6, 13.)

Primary note proof

Study-note section FDB04

> The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Study-note section FDB13

> Specialization divides Student into Undergraduate and Graduate; generalization combines common properties into Student.

Study-note section FDB13

> EER extends ER with superclass/subclass structure.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 2-4

> Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Fundamental of Database Systems module | PDF page 13 | lines 9-14

> Specialization is the process of dividing a parent-level entity (super-class) into narrower categories (sub-classes) accordingly to all the possible child categories. Specialization requires the separation of entities based on certain uncommon attributes. The process of extracting common characteristics from two or more classes (sub-classes) and combining them into a generalized superclass is called Generalization. It is the process that does the grouping entities into broader or distributed categories based on certain common attributes.

Question sources

Database Fundamentals practice questions · Question 29

Question 95 | Q01355

The attributes made up of more than one single attributes are called

A. Composite attribute

B. Derived attribute

C. Single value attribute

D. Multi value attribute

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer A: Composite attribute

Composite attributes have smaller meaningful subparts. This differs from multiple values of one attribute and from calculating a value from another.

Why each choice is right or wrong

A - Correct: Composite is the stated divisible structure. (PDF pp. 10, 10.)

B - Incorrect: Derived concerns computation from stored attributes. (PDF pp. 10, 10.)

C - Incorrect: Single-valued concerns count of values, not internal composition. (PDF pp. 10, 10.)

D - Incorrect: Multivalued concerns a set of values rather than components of one value. (PDF pp. 10, 10.)

Primary note proof

Study-note section FDB06

> Simple versus composite asks whether a value has meaningful components: Address may split into City and Street. Single-valued versus multivalued asks how many values one entity may have: a student may have several phone numbers. These dimensions differ: one address may be composite and single-valued. Age derived from birth date is derived, even if also single-valued. Attribute names alone do not establish multiplicity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 15-24

> Composite and Simple (Atomic) Attributes Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called simple or atomic attributes. Multivalued Attributes Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity. For instance, a College_degrees attribute for a person. Similarly, one person may not have any college degrees, another person may have one, and a third person may have two or more degrees; therefore, different people can have different numbers of values for the College_degrees attribute. Such attributes are called multivalued.

Fundamental of Database Systems module | PDF page 10 | lines 25-30

> Derived Attributes In some cases, two (or more) attribute values are related. For example, the Age and Birth_date attributes of a person. For a particular person entity, the value of Age can be determined from the current (today‘s) date and the value of that person‘s Birth_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute.

Question sources

Database Fundamentals practice questions · Question 30

Question 96 | Q01356

The attribute AGE is calculated from DATE_OF_BIRTH. The attribute AGE is

A. Single valued

B. Multi valued

C. Composite

D. Derived

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer D: Derived

The exact Age/Birth_date example identifies Age as derived. Age can also be single-valued, but the stem asks the classification indicated by calculating it from another attribute; D describes that dependency, while A describes a different axis.

Why each choice is right or wrong

A - Incorrect: Age can be single-valued, but that does not express the calculation relationship asked for. (PDF pp. 10, 10.)

B - Incorrect: No set of ages per person is described. (PDF pp. 10, 10.)

C - Incorrect: No meaningful component attributes of Age are specified. (PDF pp. 10, 10.)

D - Correct: Derived directly matches the module's example and the calculation clause. (PDF pp. 10, 10.)

Primary note proof

Study-note section FDB06

> Simple versus composite asks whether a value has meaningful components: Address may split into City and Street. Single-valued versus multivalued asks how many values one entity may have: a student may have several phone numbers. These dimensions differ: one address may be composite and single-valued. Age derived from birth date is derived, even if also single-valued. Attribute names alone do not establish multiplicity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 19-20

> Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set

Fundamental of Database Systems module | PDF page 10 | lines 25-30

> Derived Attributes In some cases, two (or more) attribute values are related. For example, the Age and Birth_date attributes of a person. For a particular person entity, the value of Age can be determined from the current (today‘s) date and the value of that person‘s Birth_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth_date attribute, which is called a stored attribute.

Question sources

Database Fundamentals practice questions · Question 31

Question 97 | Q01357

A student has one official Name and one Date_of_birth, but may have several Phone_number values. Which listed attribute is multivalued for a student?

A. Phone_number

B. Name

C. Date_of_birth

D. All of the mentioned

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer A: A. Phone_number

Multiplicity asks how many values of one attribute may belong to the same entity. The student may have several phone numbers, making Phone_number multivalued. The stated rules allow one official name and one birth date, so those attributes are single-valued. These classifications follow the supplied requirements rather than the names or internal structure of the values.

Why each choice is right or wrong

A - Correct: Several phone values are permitted for one student.

B - Incorrect: The model permits one official name per student.

C - Incorrect: One birth date is permitted per student.

D - Incorrect: Name and birth date are single-valued under the stated rules.

Primary note proof

Study-note section FDB06

Simple versus composite asks whether a value has meaningful components: Address may split into City and Street. Single-valued versus multivalued asks how many values one entity may have: a student may have several phone numbers. These dimensions differ: one address may be composite and single-valued. Age derived from birth date is derived, even if also single-valued. Attribute names alone do not establish multiplicity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 15-24

Composite and Simple (Atomic) Attributes Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called simple or atomic attributes. Multivalued Attributes Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity. For instance, a College_degrees attribute for a person. Similarly, one person may not have any college degrees, another person may have one, and a third person may have two or more degrees; therefore, different people can have different numbers of values for the College_degrees attribute. Such attributes are called multivalued.

Question sources

Database Fundamentals practice questions · Question 32

Question 98 | Q01358

For each student, Register_number has one value; Address records all residences and permits several values; SUBJECT_TAKEN permits several subjects; Reference permits several referees. Which listed attribute is single-valued?

A. Register_number

B. Address

C. SUBJECT_TAKEN

D. Reference

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer A: A. Register_number

Single-valued means one value per entity, independently of whether that value can be broken into components. Register_number is single-valued under the stated student rule. Address, SUBJECT_TAKEN and Reference each explicitly permit multiple values for one student. An address can be composite without necessarily being multivalued, but this particular model permits multiple residences.

Why each choice is right or wrong

A - Correct: One register number is assigned per student.

B - Incorrect: Several residence addresses are permitted.

C - Incorrect: Several subjects can be taken by one student.

D - Incorrect: Several referee references are permitted.

Primary note proof

Study-note section FDB06

> Simple versus composite asks whether a value has meaningful components: Address may split into City and Street. Single-valued versus multivalued asks how many values one entity may have: a student may have several phone numbers. These dimensions differ: one address may be composite and single-valued. Age derived from birth date is derived, even if also single-valued. Attribute names alone do not establish multiplicity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 15-24

> Composite and Simple (Atomic) Attributes Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called simple or atomic attributes. Multivalued Attributes Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity. For instance, a College_degrees attribute for a person. Similarly, one person may not have any college degrees, another person may have one, and a third person may have two or more degrees; therefore, different people can have different numbers of values for the College_degrees attribute. Such attributes are called multivalued.

Question sources

Database Fundamentals practice questions · Question 33

Question 99 | Q01359

An association of several entities in an Entity-Relation model is called :

A. Tuple

B. Record

C. Relationship

D. Field

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer C: Relationship

A relationship represents an association between entity types. Tuples and records instead represent individual table rows.

Why each choice is right or wrong

A - Incorrect: Tuple is a row. (PDF pp. 11, 13.)

B - Incorrect: Record is row data, not the ER association category. (PDF pp. 11, 13.)

C - Correct: Relationship is the explicit association term. (PDF pp. 11, 13.)

D - Incorrect: Field is an attribute component, not the association. (PDF pp. 11, 13.)

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Study-note section FDB03

> A tuple/row/record contains related attribute values; an attribute/column/field describes a property.

Study-note section FDB03

> A relation is a set of tuples, represented as a table.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 13-15

> A relationship type represents the association between entity types. For example, _Enrolled in‘ is a relationship type that exists between entity Student and Course. In ER diagram, relationship is represented by a diamond and connecting the entities with lines.

Fundamental of Database Systems module | PDF page 13 | lines 24-27

> Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Database Fundamentals practice questions · Question 34

Question 100 | Q01360

To include integrity constraint in an existing relation use:

A. Modify table

B. Drop table

C. Alter table

D. Create table

Hint

Distinguish table definition changes from changes to stored rows.

Answer C: Alter table

Adding an integrity constraint to an existing table changes its definition, so ALTER TABLE is the appropriate command family. It can add a named constraint or a primary key, provided existing data satisfies the rule. DROP TABLE removes the object, and CREATE TABLE defines a new one. MODIFY TABLE does not follow the ALTER TABLE command form required for this structural change. The remaining constraint clause depends on the integrity rule to be enforced.

Why each choice is right or wrong

A - Incorrect: Incorrect: the required command family for this definition change is ALTER TABLE, not MODIFY TABLE.

B - Incorrect: Incorrect: DROP TABLE removes the existing object instead of adding a constraint.

C - Correct: Correct: ALTER TABLE can add a constraint to an existing table definition.

D - Incorrect: Incorrect: CREATE TABLE defines a new table rather than changing this existing one.

Primary note proof

Study-note section FDB30

> ALTER TABLE Employee ADD COLUMN Email VARCHAR(120); adds a column in PostgreSQL-style syntax. ALTER TABLE Employee DROP COLUMN Email; removes it. ALTER TABLE Employee ADD CONSTRAINT uq_email UNIQUE (Email); adds a named constraint. Use ALTER rather than DALTER, and do not separate column_name and column_type with a comma; validate the resulting SQL before execution.

Study-note section FDB26

DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 49 | lines 16-28

Data and Domain Integrity ● This is now increasing handled by the database. In Oracle, for example, when defining a table you can specify: ● CREATE TABLE table-name (attr2 attr-type NOT NULL, forbids NULL values attrN attr-type CHECK (attrN = UPPER(attrN) verifies that the data meets certain criteria attrO attr-type DEFAULT default_value); Supplies default values Referential Integrity ● Ensures that dependent relationships in the data are maintained. In Oracle, for example: ● CREATE TABLE table-name (attr1 attr-type PRIMARY KEY, attr2 attr-type NOT NULL, ..., attrM attr-type REFERENCES owner.tablename(attrname) ON DELETE CASCADE, ...

Question sources

Database Fundamentals practice questions · Question 35

Question 101 | Q01361

Which option is not a general SQL integrity-constraint keyword or clause? Distinguish a constraint clause from a business condition that could be expressed inside one.

A. Not null

B. Positive

C. Unique

D. Check predicate

Hint

Read each SQL constraint's scope and enforcement rule.

Answer B: Positive

NOT NULL forbids missing values, UNIQUE enforces uniqueness under the DBMS's NULL rules, and CHECK evaluates a declared predicate. POSITIVE by itself is not a general SQL constraint keyword. A positivity requirement is still a valid business rule and can be written as a CHECK predicate such as CHECK (Amount > 0). The question therefore distinguishes clause syntax from the meaning of a possible rule, making B the unique incorrect keyword rather than rejecting positivity as an integrity requirement.

Why each choice is right or wrong

A - Incorrect: Incorrect selection: NOT NULL is a constraint that forbids NULL values.

B - Correct: Correct selection: POSITIVE is not the standalone constraint keyword; a CHECK predicate can enforce positivity.

C - Incorrect: Incorrect selection: UNIQUE is a constraint enforcing uniqueness.

D - Incorrect: Incorrect selection: CHECK is the clause for enforcing a declared predicate.

Primary note proof

Study-note section FDB16

> Example: CREATE TABLE Enrollment (StudentID INTEGER NOT NULL REFERENCES Student(StudentID), CourseID INTEGER NOT NULL REFERENCES Course(CourseID), Grade INTEGER CHECK (Grade BETWEEN 0 AND 100), PRIMARY KEY (StudentID,CourseID)); Each column declaration gives its name then type, with commas between declarations. CHECK is a constraint clause, not a standalone GRANT-like statement. POSITIVE is not a general SQL constraint keyword; CHECK (Amount > 0) expresses positivity.

Study-note section FDB16

> PRIMARY KEY implies uniqueness and non-NULL. UNIQUE does not generally imply NOT NULL; NULL treatment differs by DBMS.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 14 | lines 17-20

> Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Fundamental of Database Systems module | PDF page 47 | lines 20-27

> • Five types i. Required data ii. attribute domain constraints iii. entity integrity iv. referential integrity v. enterprise constraints i. Required Data • Some attributes must always contain a value -- they cannot have a NULL value

Fundamental of Database Systems module | PDF page 49 | lines 19-22

> • CREATE TABLE table-name (attr2 attr-type NOT NULL, forbids NULL values attrN attr-type CHECK (attrN = UPPER(attrN) verifies that the data meets certain criteria attrO attr-type DEFAULT default_value); Supplies default values

Question sources

Database Fundamentals practice questions · Question 36

Question 102 | Q01362

Foreign key is the one in which the _____ of one relation is referenced in another relation.

A. Foreign key

B. Primary key

C. References

D. Check constraint

Hint

Match the validity requirement to the appropriate constraint definition.

Answer B: Primary key

A foreign key references a matching candidate, usually primary, key in the parent. B is the offered conventional primary-key case; the definition also permits a candidate key, so primary is not an exclusive universal requirement.

Why each choice is right or wrong

A - Incorrect: An arbitrary foreign key is not the required parent unique target. (PDF pp. 48.)

B - Correct: Primary key is the listed usual referenced key. (PDF pp. 48.)

C - Incorrect: References names a mechanism rather than the key category. (PDF pp. 48.)

D - Incorrect: A CHECK predicate is not the referenced key. (PDF pp. 48.)

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Database Fundamentals practice questions · Question 37

Question 103 | Q01363

Which are uses of data integrity constraints, rather than access authorization? Select all that apply.

A. Control who is allowed access to the data

B. Ensure that duplicate records are not entered into the table

C. Improve the quality of data entered for a specific property (i.e., table column)

D. Prevent users from changing the values stored in the table

Hint

Match the validity requirement to the appropriate constraint definition.

Answer B, C: B. Ensure that duplicate records are not entered into the table; C. Improve the quality of data entered for a specific property (i.e., table column)

Integrity constraints test whether data is valid according to the schema's rules. Uniqueness constraints can reject duplicate keyed records, while domain constraints reject inappropriate values for a column. These support the two selected purposes. Authorization controls who may access the data. Integrity rejects invalid changes rather than prohibiting all changes, so it does not imply that every stored value is immutable.

Why each choice is right or wrong

A - Incorrect: Controlling who may access data is authorization rather than data integrity itself.

B - Correct: Key or uniqueness constraints prevent duplicate keyed records.

C - Correct: Domain constraints validate types and allowed values for a specific column.

D - Incorrect: Integrity permits valid updates; it does not prohibit every change to stored values.

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 47 | lines 4-19

> ● Security vs Integrity ◦ Database security makes sure that the user is authorised to access information ◦ Database integrity makes sure that (authorised) users use that information correctly ● Integrity constraints ◦ Domain constraints apply to data types ◦ Attribute constraints apply to columns ◦ Relation constraints apply to rows in a single table ◦ Database constraints apply between tables ● Intrarecord integrity (enforcing constraints on contents of fields, etc.) ● Referential Integrity (enforcing the validity of references between records in the database) ●

Concurrency control (ensuring the validity of database updates in a shared multiuser environment) ● The constraints we wish to impose in order to protect the database from becoming inconsistent.

Fundamental of Database Systems module | PDF page 48 | lines 4-12

ii. Attribute domain constraints ● Every attribute has a domain, that is a set of values that are legal for it to use ● For example: ◦ The domain of sex in the employee relation is —M‖ or —F‖ ● Domain ranges can be used to validate input to the database iii. Entity integrity ● The primary key of any entity: ◦ Must be Unique ◦ Cannot be NULL

Question sources

Database Fundamentals practice questions · Question 38

Question 104 | Q01364

The relationship between the two tables are created using _____

A. Candidate Key

B. Primary Key

C. Foreign Key

D. Check Constraint

Hint

Match the validity requirement to the appropriate constraint definition.

Answer C: Foreign Key

A foreign key links child occurrences to the matching parent candidate key. Primary/candidate keys identify the target, but foreign-key referencing creates the intertable link.

Why each choice is right or wrong

A - Incorrect: Candidate keys identify potential reference targets rather than expressing the link alone. (PDF pp. 48.)

B - Incorrect: A primary key identifies a row; a foreign key expresses the reference. (PDF pp. 48.)

C - Correct: The foreign key is explicitly the linking mechanism. (PDF pp. 48.)

D - Incorrect: CHECK restricts values rather than defining this intertable reference. (PDF pp. 48.)

Primary note proof

Study-note section FDB15

Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB07

A primary key is one selected candidate key; remaining candidates are alternate keys.

Study-note section FDB16

POSITIVE is not a general SQL constraint keyword; CHECK (Amount > 0) expresses positivity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Database Fundamentals practice questions · Question 39

Question 105 | Q01365

Customer has candidate keys CustId, Email and ContactNo. CustId is an immutable compact generated identifier; Email and ContactNo may change. A policy prefers stable compact candidate keys. Which is the best primary key under that policy?

A. CustId

B. Email

C. ContactNo

D. CustId and ContactNo

Hint

Test uniqueness and minimality separately when classifying keys.

Answer A: A. CustId

Every candidate key is eligible for primary-key selection, but the policy gives a reason to prefer one. CustId is explicitly stable and compact, while the contact attributes may change. Choosing CustId follows that policy without denying the uniqueness of the other candidates. Combining CustId and ContactNo is nonminimal because CustId alone already identifies each customer.

Why each choice is right or wrong

A - Correct: Its stated stability and compactness satisfy the preference policy.

B - Incorrect: Email is eligible but may change, so it loses under this policy.

C - Incorrect: ContactNo is eligible but may change, so it loses under this policy.

D - Incorrect: Adding ContactNo to an already unique CustId is unnecessary and nonminimal.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> Prefer stable, compact identifiers when practical, but Email, SSN or EmployeeID is a candidate only if business rules guarantee uniqueness.

Study-note section FDB07

> A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Database Fundamentals practice questions · Question 40

Question 106 | Q01366

Which of the following statement(s) is/are FALSE about primary key?

A. There can be more than one primary key for a table

B. Columns with string data types cannot be made Primary Key because they are large

C. Both A and B

D. None of the above

Hint

Test uniqueness and minimality separately when classifying keys.

Answer C: Both A and B

The designer selects one candidate key as the primary key; a composite primary key is still one key. No numeric-only restriction appears: a key is defined by uniqueness, not by a prohibition on string type. Thus both A and B are false claims.

Why each choice is right or wrong

A - Incorrect: False: several candidate keys do not mean several primary keys. (PDF pp. 11, 14.)

B - Incorrect: False: the definition imposes uniqueness, not a ban on strings. (PDF pp. 11, 14.)

C - Correct: Both preceding claims conflict with the key definition. (PDF pp. 11, 14.)

D - Incorrect: There are false claims, so none is not correct. (PDF pp. 11, 14.)

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Study-note section FDB16

> PRIMARY KEY implies uniqueness and non-NULL.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Fundamental of Database Systems module | PDF page 14 | lines 17-20

> Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Question sources

Database Fundamentals practice questions · Question 41

Question 107 | Q01367

Which is the subset of SQL commands used to manipulate Oracle Database structures, including tables?

A. Data Definition Language(DDL)

B. Data Manipulation Language(DML)

C. DML and DDL

D. None of the Mentioned

Hint

Classify each SQL command by the kind of change it performs.

Answer A: Data Definition Language(DDL)

Schema-structure manipulation falls under definition language, despite the everyday word manipulate. DML is the separate category for data manipulation.

Why each choice is right or wrong

A - Correct: DDL contains the structural commands. (PDF pp. 26.)

B - Incorrect: DML concerns data operations. (PDF pp. 26.)

C - Incorrect: The asked subset for structures is DDL, not a combined category. (PDF pp. 26.)

D - Incorrect: DDL is offered. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 42

Question 108 | Q01368

Using the course's textbook SQL command categories, with RENAME treated as DDL, which listed commands are outside DDL? Select all that apply.

A. RENAME

B. REVOKE

C. GRANT

D. UPDATE

Hint

Classify each SQL command by the kind of change it performs.

Answer B, C, D: B. REVOKE; C. GRANT; D. UPDATE

DDL concerns structural definitions and includes RENAME in this course classification. GRANT and REVOKE manage privileges, placing them in DCL, while UPDATE changes existing data and belongs to DML. All three are consequently outside DDL. Classifying by each statement's effect distinguishes schema operations, access permissions and data changes even though all are expressed using SQL.

Why each choice is right or wrong

A - Incorrect: RENAME is commonly classified as DDL in the note's textbook taxonomy.

B - Correct: REVOKE removes privileges and is DCL rather than DDL.

C - Correct: GRANT assigns privileges and is DCL rather than DDL.

D - Correct: UPDATE changes data and is DML rather than DDL.

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 54 | lines 6-10

2.2.3 Authorization Rules • Most current DBMS permit the DBA to define —access permissions‖ on a table by table basis (at least) using the GRANT and REVOKE SQL commands • Some systems permit finer grained authorization (most use GRANT and REVOKE on variant views

Question sources

Database Fundamentals practice questions · Question 43

Question 109 | Q01369

Which command empties an existing table while retaining its definition, using DBMS-specific table-emptying mechanisms rather than asserting that rollback is impossible?

A. TRUNCATE command

B. Create command

C. Drop table command

D. Alter table command

Hint

Classify each SQL command by the kind of change it performs.

Answer A: A. TRUNCATE command

TRUNCATE empties a table while retaining the table object. Its implementation, logging and transaction behavior depend on the DBMS, so the answer does not rely on a claim that rollback is impossible. CREATE defines an object, DROP TABLE removes it, and ALTER changes its definition. Only TRUNCATE has the requested table-emptying role among these choices.

Why each choice is right or wrong

A - Correct: TRUNCATE empties the table and keeps its definition.

B - Incorrect: CREATE defines objects rather than emptying an existing table.

C - Incorrect: DROP TABLE removes the definition as well as the table.

D - Incorrect: ALTER changes the table definition rather than serving as the table-emptying command.

Primary note proof

Study-note section FDB26

DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 52 | lines 2-3

> • Part or all of a transaction can be undone using ROLLBACK • COMMIT;

Question sources

Database Fundamentals practice questions · Question 44

Question 110 | Q01370

Which SQL command form removes a database object together with its contained database objects?

A. DROP DATABASE

B. DELETE DATABASE

C. REMOVE DATABASE

D. None of the mentioned

Hint

Classify each SQL command by the kind of change it performs.

Answer A: DROP DATABASE

DROP removes a database object, whereas DELETE removes rows from an existing table. DROP DATABASE is consequently the correct command form for removing a database and its contained objects. DELETE DATABASE does not express that structural operation, and REMOVE DATABASE does not follow the taught DROP syntax. Since the correct DROP DATABASE form is one of the options, the none-of-the-mentioned alternative is also incorrect.

Why each choice is right or wrong

A - Correct: Correct: DROP DATABASE removes the database object and its contained objects.

B - Incorrect: Incorrect: DELETE is row removal; deleting rows does not remove the database object.

C - Incorrect: Incorrect: the database-removal command form uses DROP, not REMOVE.

D - Incorrect: Incorrect: DROP DATABASE is present and performs the requested structural operation.

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Study-note section FDB26

> SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 45

Question 111 | Q01371

In SQL Server, which statement drops the two databases named Sales and suppliers? Assume permissions and database state allow the operation.

A. DROP DATABASE Sales, NewSales

B. DROP DATABASE Sales, suppliers

C. DROP DATABASE Sales and suppliers

D. DROP DATABASE Sales-suppliers

Hint

Classify each SQL command by the kind of change it performs.

Answer B: B. DROP DATABASE Sales, suppliers

SQL Server permits a comma-separated list of database names after DROP DATABASE. B names exactly Sales and suppliers using that syntax. A names a different second database, while AND and a hyphen do not form the required two-name list. The answer is specific to SQL Server and does not assert that the same multi-database syntax works in every SQL dialect.

Why each choice is right or wrong

A - Incorrect: The second target is NewSales rather than suppliers.

B - Correct: The supported comma list names both required databases.

C - Incorrect: AND is not the separator in this DROP DATABASE syntax.

D - Incorrect: A hyphen does not separate these two database names.

Primary note proof

Study-note section FDB26

DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 46

Question 112 | Q01372

Command that is used to add attributes to an existing relation, is said to be

A. Alter

B. Modify

C. Tailor

D. Eliminate

Hint

Distinguish table definition changes from changes to stored rows.

Answer A: ALTER.

Adding attributes changes a relation schema. FDB30 demonstrates ALTER TABLE Employee ADD COLUMN Email VARCHAR(120), keeping the column name next to its type. ALTER is therefore the required command family. The remaining alternatives do not name the general SQL schema-change command shown by the complete note.

Why each choice is right or wrong

A - Correct: ALTER changes an existing relation definition and supports adding a column.

B - Incorrect: MODIFY alone is not the general command for adding attributes to an existing relation.

C - Incorrect: TAILOR is not an SQL schema-change command established here.

D - Incorrect: ELIMINATE is not the SQL command for adding a relation attribute.

Primary note proof

Study-note section FDB30

> ALTER TABLE Employee ADD COLUMN Email VARCHAR(120); adds a column in PostgreSQL-style syntax. ALTER TABLE Employee DROP COLUMN Email; removes it. ALTER TABLE Employee ADD CONSTRAINT uq_email UNIQUE (Email); adds a named constraint. Use ALTER rather than DALTER, and do not separate column_name and column_type with a comma; validate the resulting SQL before execution.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 47

Question 113 | Q01373

Which keyword pair changes a table definition to remove a specified column, rather than changing or deleting row values? Treat the comma in an option as separating keyword names.

A. UPDATE TABLE

B. TRUNCATE COLUMN

C. ALTER, DROP

D. DELETE COLUMN

Hint

Distinguish table definition changes from changes to stored rows.

Answer C: ALTER, DROP

Removing a column changes the table's definition. The appropriate form is ALTER TABLE followed by DROP COLUMN, so the keyword pair represented by C is correct. Its comma separates the two keyword names rather than belonging in executable SQL. UPDATE changes stored row values, DELETE removes rows, and TRUNCATE empties a table while retaining its definition; none of those operations removes the specified column definition.

Why each choice is right or wrong

A - Incorrect: Incorrect: UPDATE changes row values rather than removing a column definition.

B - Incorrect: Incorrect: TRUNCATE empties a table; TRUNCATE COLUMN is not the column-removal form.

C - Correct: Correct: ALTER TABLE ... DROP COLUMN removes a column definition.

D - Incorrect: Incorrect: DELETE removes rows, not a column from the schema.

Primary note proof

Study-note section FDB30

> ALTER TABLE Employee ADD COLUMN Email VARCHAR(120); adds a column in PostgreSQL-style syntax. ALTER TABLE Employee DROP COLUMN Email; removes it. ALTER TABLE Employee ADD CONSTRAINT uq_email UNIQUE (Email); adds a named constraint. Use ALTER rather than DALTER, and do not separate column_name and column_type with a comma; validate the resulting SQL before execution.

Study-note section FDB26

> DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition.

Study-note section FDB26

> TRUNCATE empties a table using implementation-specific mechanisms.

Study-note section FDB29

UPDATE Book SET Price = 600 WHERE ID = 'B001'; changes matching rows. UPDATE without WHERE targets every row.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 48

Question 114 | Q01374

Which one of these is used with SELECT clause to fetch all columns from a table?

A. ALL

B. *

C. DISTINCT

D. AS

Hint

Trace how SELECT forms its result columns and rows.

Answer B: The asterisk (*).

FDB27 explicitly says SELECT * selects all columns. ALL and DISTINCT instead govern duplicate result rows, so neither means every column. AS introduces an alias. A query still needs the appropriate FROM source for table columns; the question asks only which SELECT-list symbol requests them all.

Why each choice is right or wrong

A - Incorrect: ALL permits duplicate result rows and does not expand the result to every column.

B - Correct: The wildcard * requests every column from the selected table source.

C - Incorrect: DISTINCT removes duplicate result rows rather than selecting all available columns.

D - Incorrect: AS supplies an alias for a result expression or source, not a wildcard column list.

Primary note proof

Study-note section FDB27

SELECT Name,Address FROM Customer; produces only those mailing-label fields. SELECT * selects all columns; DISTINCT removes duplicate result rows; AS supplies an alias; ALL permits duplicates and does not mean all columns. SQL normally preserves duplicates without DISTINCT.

A query reading table columns needs an appropriate FROM source. A literal-only query such as SELECT 1 may legally omit FROM.

Study-note section FDB27

WHERE alone is a clause rather than a complete read-only statement, so it cannot be compared as an equivalent executable command.

Study-note section FDB27

Typical written clause order is SELECT … FROM … WHERE … GROUP BY … HAVING … ORDER BY ….

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 16-21

Example: Identify all managers who work in a London branch SELECT * FROM Staff s, Branch b WHERE s.branchNo = b.branchNo AND s.position = _Manager‘ AND b.city = _london‘;

Fundamental of Database Systems module | PDF page 52 | lines 4-6

● SET TRANSACTION READ ONLY; ● SELECT NAME, ADDRESS FROM WORKERS; ● SELECT MANAGER, ADDRESS FROM PLACES;

Question sources

Database Fundamentals practice questions · Question 49

Question 115 | Q01375

A parent table has a DATE column, a nullable data column and a primary-key column referenced by existing child rows with ON UPDATE RESTRICT. Which column's change to a different otherwise valid value is blocked specifically because the old key is still referenced? No other constraints or triggers block changes.

A. DATE type columns in the table

B. Columns which allows NULL values in the table

C. A primary key column which also serves as foreign key reference in another table

D. All of the above

Hint

Read each SQL constraint's scope and enforcement rule.

Answer C: C. A primary key column which also serves as foreign key reference in another table

The stated restrictive action prevents changing the parent key while existing children still refer to its old value. This is a consequence of the particular reference state and action, rather than inherent immutability of every primary key. DATE and nullable columns can accept otherwise valid changes. Therefore only the referenced primary-key column has the specified update block.

Why each choice is right or wrong

A - Incorrect: DATE values can be changed to other permitted dates.

B - Incorrect: Allowing NULL does not itself block valid updates.

C - Correct: Existing child references plus ON UPDATE RESTRICT block this key change.

D - Incorrect: The date and nullable columns are not blocked under the given conditions.

Primary note proof

Study-note section FDB16

> PRIMARY KEY implies uniqueness and non-NULL. UNIQUE does not generally imply NOT NULL; NULL treatment differs by DBMS. A CHECK can pass when its result is unknown, so add NOT NULL if missing values must be forbidden. Parent updates/deletes may use NO ACTION/RESTRICT, CASCADE or SET NULL. A referenced key is not inherently unchangeable: legality depends on matching children and the declared action.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity ● A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key ● Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 49 | lines 4-11

> Deletion rules ● A row should not be deleted from the referenced table (parent) if there are matching rows in the referencing table (child) ● Three ways to handle this ◦ Restrict -- disallow the delete ◦ Nullify -- reset the foreign keys in the child to some NULL or dummy value ◦ Cascade -- Delete all rows in the child where there is a foreign key matching the key in the parent row being deleted

Fundamental of Database Systems module | PDF page 26 | lines 10-11

> 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 50

Question 116 | Q01376

Using the course classification that includes SELECT in DML, which listed language terms can describe application requests to retrieve database data? Select all that apply.

A. DML

B. DDL

C. Query language

D. All of the Mentioned

Hint

Classify each SQL command by the kind of change it performs.

Answer A, C: A. DML; C. Query language

A query language expresses requests for data. The course also includes SELECT retrieval within DML, alongside insertion, update and deletion. Both terms can therefore describe the application's retrieval request at different levels of terminology. DDL changes structural definitions instead of expressing the requested retrieval. Since DDL does not fit, the aggregate all-of-the-mentioned choice is also false.

Why each choice is right or wrong

A - Correct: DML includes retrieval with SELECT in the note's classification.

B - Incorrect: DDL changes schema structure rather than expressing ordinary data retrieval requests.

C - Correct: A query language expresses requests for data, directly matching the stem.

D - Incorrect: DDL does not fit the retrieval role, so the inclusive choice is false.

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 24 | lines 5-9

> Relational algebra is the basic set of operations for the relational model. These operations enable a user to specify basic retrieval requests (or queries). The result of an operation is a new relation, which may have been formed from one or more input relations. The algebra operations thus produce new relations. Since each operation returns a relation, operations can be composed (Algebra is —closed‖.)

Question sources

Database Fundamentals practice questions · Question 51

Question 117 | Q01377

Which of the following is not included in DML (Data Manipulation Language)?

A. Insert

B. Update

C. Delete

D. Alter

Hint

Classify each SQL command by the kind of change it performs.

Answer D: Alter

INSERT, UPDATE and DELETE are all in the DML subsection. ALTER is instead in the preceding DDL list.

Why each choice is right or wrong

A - Incorrect: INSERT is listed as DML. (PDF pp. 26.)

B - Incorrect: UPDATE is listed as DML. (PDF pp. 26.)

C - Incorrect: DELETE is listed as DML. (PDF pp. 26.)

D - Correct: ALTER is the DDL exception. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 52

Question 118 | Q01378

In the course's SQL command taxonomy, how is TRUNCATE classified?

A. DDL

B. DML

C. DCL

D. TCL

Hint

Classify each SQL command by the kind of change it performs.

Answer A: DDL

The course groups TRUNCATE with DDL commands. It empties a table through mechanisms that depend on the implementation, but that fact does not turn it into a privilege-management or transaction-boundary command. DML covers commands such as INSERT, UPDATE and DELETE; DCL covers GRANT and REVOKE; TCL covers COMMIT, ROLLBACK and SAVEPOINT. The requested classification is the course convention, not a claim that TRUNCATE has identical recovery behavior in every DBMS.

Why each choice is right or wrong

A - Correct: Correct: the course's textbook classification includes TRUNCATE under DDL.

B - Incorrect: Incorrect: DML includes row operations such as INSERT, UPDATE and DELETE; TRUNCATE is assigned to DDL in this taxonomy.

C - Incorrect: Incorrect: DCL manages privileges through commands such as GRANT and REVOKE.

D - Incorrect: Incorrect: TCL controls transaction boundaries or rollback points, unlike TRUNCATE.

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 53

Question 119 | Q01379

A type of query that is placed within a WHERE or HAVING clause of another query is called

A. Master query

B. Sub query

C. Super query

D. Multi-query

Hint

Distinguish a stored query definition from stored data and trace subquery scope.

Answer B: Sub query

The optimization section identifies nested queries as separate query blocks, and the view example places a SELECT inside an outer WHERE. Subquery is the offered name matching this inner/nested query; WHERE or HAVING placement does not define every possible subquery location.

Why each choice is right or wrong

A - Incorrect: Master describes neither the inner query nor the module's nesting distinction. (PDF pp. 34, 58.)

B - Correct: Subquery corresponds to the nested inner SELECT demonstrated. (PDF pp. 34, 58.)

C - Incorrect: Super query does not name the inner block. (PDF pp. 34, 58.)

D - Incorrect: Multi-query does not identify the nesting relationship. (PDF pp. 34, 58.)

Primary note proof

Study-note section FDB33

> A subquery is a SELECT nested inside another statement. It may appear in WHERE, HAVING or FROM, not only the first two. IN tests membership; EXISTS tests whether at least one qualifying row exists. Correlated subqueries refer to the outer row. A scalar subquery must produce an appropriate single value. Beware NOT IN when its subquery can return NULL; NOT EXISTS often expresses the intended absence more safely.

Study-note section FDB33

> CREATE VIEW CSStudent AS SELECT StudentID,Name FROM Student WHERE Major = 'CS'; stores a query definition and presents a virtual table.

Study-note section FDB33

> An ordinary view does not necessarily store its result; a materialized view does.

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 12-16

> • Query block: The basic unit that can be translated into the algebraic operators and optimized. • A query block contains a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clause if these are part of the block. • Nested queries within a query are identified as separate query blocks.

Fundamental of Database Systems module | PDF page 58 | lines 15-20

> View example CREATE VIEW OwnDept AS SELECT Name, Phone FROM Employee WHERE Department = (SELECT Department FROM Employee WHERE name = USER)

Question sources

Database Fundamentals practice questions · Question 54

Question 120 | Q01380

A data manipulation command the combines the records from one or more tables is called

A. SELECT

B. PROJECT

C. JOIN

D. PRODUCT

Hint

Compute matching pairs, then account for any preserved unmatched rows.

Answer C: JOIN

The DML list contains JOIN, and the query-plan example describes joining Staff and Branch into a joined relation. This directly matches combining records rather than selecting columns alone.

Why each choice is right or wrong

A - Incorrect: SELECT can retrieve data, but JOIN names the combining operation illustrated. (PDF pp. 26, 28.)

B - Incorrect: PROJECT selects columns rather than linking records by the illustrated condition. (PDF pp. 26, 28.)

C - Correct: JOIN is explicitly used to combine the example relations. (PDF pp. 26, 28.)

D - Incorrect: PRODUCT combines all pairs; it is the broader Cartesian operation rather than the illustrated join command. (PDF pp. 26, 28.)

Primary note proof

Study-note section FDB31

An inner join returns matching pairs satisfying ON. An equijoin uses equality; a theta join permits another comparison. A natural join is ordinarily an inner equality join on every same-named column and merges those duplicate columns. JOIN ... USING (DeptID) specifies shared join columns explicitly. CROSS JOIN forms every pair without a matching condition.

Study-note section FDB35

Rename ρ changes relation or attribute names.

Study-note section FDB35

Identify the join variant before predicting its 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 26 | lines 10-11

7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 28 | lines 17-24

Query 2 (Better) (position=„Manager“)^(city=„London“) (Staff Staff.branchNo = Branch.branchNo Branch) - Again requires (1000+50) disk accesses to read from Staff and Branch - Joins Staff and Branch on branchNo with 1000 tuples (1 employee : 1 branch) - Requires (1000) disk access to read in joined relation and check predicate Total Work = (1000+50) + 2*(1000) = 3050 I/O operations

Question sources

Database Fundamentals practice questions · Question 55

Question 121 | Q01382

Data manipulation language (DML) includes statements that modify the _____ of a database.

A. Structure

B. Data

C. User

D. Size

Hint

Classify each SQL command by the kind of change it performs.

Answer B: Data

Data-definition commands are distinct from data-manipulation commands such as INSERT, DELETE and UPDATE. Their category concerns the stored data, while SELECT is also included as a retrieval command that need not modify state.

Why each choice is right or wrong

A - Incorrect: Structure belongs to definition language. (PDF pp. 26.)

B - Correct: Data is the manipulated content. (PDF pp. 26.)

C - Incorrect: User administration is not the target category here. (PDF pp. 26.)

D - Incorrect: Size can change as a consequence, not the category manipulated. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 57

Question 122 | Q01383

Which listed term names the standard SQL expression form for converting a value to a requested data type, rather than vendor-specific conversion syntax?

A. Cast

B. Drop type

C. Alter type

D. Convert

Hint

Check set membership, duplicates and compatible result types.

Answer A: A. Cast

CAST is the standard expression form that requests conversion of a value to a target type. CONVERT can also perform conversions in supporting vendor dialects, but the question specifically excludes vendor-specific syntax. DROP TYPE and ALTER TYPE act on type definitions rather than converting one expression value. The target value must still be representable in the requested type.

Why each choice is right or wrong

A - Correct: CAST supplies the requested standard expression conversion.

B - Incorrect: DROP TYPE removes a type definition.

C - Incorrect: ALTER TYPE changes a type definition.

D - Incorrect: CONVERT is vendor-specific in the supplied course distinction.

Primary note proof

Study-note section FDB34

> CAST('42' AS INTEGER) converts an expression to a requested type if the value permits it. CONVERT is vendor-specific and can also perform type conversion; specify the SQL dialect when comparing it with CAST. In SELECT xyz FROM table1 UNION xyz FROM table2, the second SELECT is missing. In the corrected query, xyz is a selected column/expression, not a table name.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 19 | lines 11-25

> 5.2 Choosing Attribute Data Types CHAR - fixed-length character VARCHAR() - variable-length characters DATE - actual date BLOB - binary large object (good for graphics images, audio, sound clips, etc.) NUMERIC - positive/negative numbers Exact • SMALLINT → 16 bit • INT → 32 bit • LONG → 64 bit • DECIMAL → 128 bit ... Approximate • FLOAT (single precision) → 32 bit with ≈ 6-9 digit precision... • REAL(variation of double) → 32 bit with ≈ 15 digit precision... • DOUBLE (double precision) → 64 bit with ≈ 32 bit precision...

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 58

Question 123 | Q01384

Which SQL keyword combines the rows returned by two compatible SELECT statements into one result set, removing duplicates by default?

A. COMBINE

B. CONCAT

C. JOIN

D. UNION

Hint

Check set membership, duplicates and compatible result types.

Answer D: UNION

UNION combines the rows of compatible query results and removes duplicates by default. It is written between the SELECT statements. JOIN instead combines input rows through matching or product behavior within a query, rather than taking the set union of completed results. CONCAT concerns concatenation of values, and COMBINE does not form the required SQL set-operation syntax. D is therefore the unique correct selection for this result-set operation.

Why each choice is right or wrong

A - Incorrect: Incorrect: COMBINE is not the UNION set-operation keyword required between these SELECT statements.

B - Incorrect: Incorrect: concatenating values does not perform the requested union of query-result rows.

C - Incorrect: Incorrect: JOIN forms combinations of input rows rather than this union of compatible SELECT results.

D - Correct: Correct: UNION combines compatible query results and removes duplicates by default.

Primary note proof

Study-note section FDB34

SELECT ID FROM A UNION SELECT ID FROM B; combines compatible result columns and removes duplicate rows by default. UNION ALL retains duplicates. Explicit UNION DISTINCT is accepted in some dialects, including PostgreSQL; it is not a separate algebra rule. INTERSECT returns common rows; EXCEPT returns first-result rows absent from the second. Inputs need the same column count and compatible corresponding types.

Related reading (context, not primary note proof)

Study-note section FDB31

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 10-16

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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 59

Question 124 | Q01385

What is xyz in the following SQL statement? SELECT xyz FROM table1 UNION xyz FROM table2;

A. row name

B. column name

C. table name

D. database name

Hint

Check set membership, duplicates and compatible result types.

Answer B: xyz denotes a selected column/expression; the second SELECT is missing.

The printed statement is malformed because the second branch says UNION xyz rather than UNION SELECT xyz. Nevertheless, the role of xyz in the first SELECT list is clear: it is a column or expression, not a row, table or database name. FDB34 explicitly identifies that role and the required repair, so the terminology question remains answerable.

Why each choice is right or wrong

A - Incorrect: SQL does not select a row by placing its row name in this SELECT-list position.

B - Correct: xyz occupies the selected-column position; the second branch needs its missing SELECT keyword.

C - Incorrect: The table sources are table1 and table2, which follow FROM rather than SELECT.

D - Incorrect: No database identifier occupies xyz's SELECT-list position in this statement.

Primary note proof

Study-note section FDB34

> SELECT ID FROM A UNION SELECT ID FROM B; combines compatible result columns and removes duplicate rows by default. UNION ALL retains duplicates. Explicit UNION DISTINCT is accepted in some dialects, including PostgreSQL; it is not a separate algebra rule. INTERSECT returns common rows; EXCEPT returns first-result rows absent from the second. Inputs need the same column count and compatible corresponding types.

Study-note section FDB34

> In the corrected query, xyz is a selected column/expression, not a table name.

Study-note section FDB34

> In SELECT xyz FROM table1 UNION xyz FROM table2, the second SELECT is missing.

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 60

Question 125 | Q01386

In a dialect that accepts explicit UNION DISTINCT, which listed modifier requests duplicate elimination when used with UNION?

A. ALL

B. NARROW

C. STRICT

D. DISTINCT

Hint

Check set membership, duplicates and compatible result types.

Answer D: DISTINCT

UNION eliminates duplicate result rows by default. In the stated dialect, DISTINCT explicitly requests that same behavior, while ALL retains duplicates. NARROW and STRICT are not modifiers in the UNION [ALL | DISTINCT] form being tested. This classification does not imply that every DBMS accepts the explicit DISTINCT spelling: the question states that support, and bare UNION remains sufficient to request duplicate elimination.

Why each choice is right or wrong

A - Incorrect: Incorrect: ALL retains duplicates, the opposite of the requested behavior.

B - Incorrect: Incorrect: NARROW is not an ALL or DISTINCT modifier in the stated UNION form.

C - Incorrect: Incorrect: STRICT is not an ALL or DISTINCT modifier in the stated UNION form.

D - Correct: Correct: DISTINCT explicitly requests duplicate elimination in the stated dialect.

Primary note proof

Study-note section FDB34

> SELECT ID FROM A UNION SELECT ID FROM B; combines compatible result columns and removes duplicate rows by default. UNION ALL retains duplicates. Explicit UNION DISTINCT is accepted in some dialects, including PostgreSQL; it is not a separate algebra rule. INTERSECT returns common rows; EXCEPT returns first-result rows absent from the second. Inputs need the same column count and compatible corresponding types.

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 10-16

> 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. Union (U) - Tuples in relation 1 and in relation 2. Intersection (∩) - Only tuples in relation 1 and in relation 2 (only the common tuples).

Fundamental of Database Systems module | PDF page 26 | lines 4-11

SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 61

Question 126 | Q01387

The database design prevents some data from being stored due to _____.

A. Deletion anomalies

B. Insertion anomalies

C. Update anomalies

D. Selection anomalies

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer B: Insertion anomalies

Insertion anomaly is explicitly the inability to insert a new record properly when the whole primary key is unknown. The other anomalies concern losing information on deletion or inconsistent/multiple updates.

Why each choice is right or wrong

A - Incorrect: Deletion anomaly loses existing related information. (PDF pp. 16.)

B - Correct: Insertion anomaly prevents a proper new insertion. (PDF pp. 16.)

C - Incorrect: Update anomaly concerns changing existing values. (PDF pp. 16.)

D - Incorrect: Selection anomaly is not among the module's stated three types. (PDF pp. 16.)

Primary note proof

Study-note section FDB19

Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies. It does not require seeing bad current data: declared dependencies suffice. Adding a generated row ID may make insertion technically possible, but does not remove

repeated facts or fix the dependency design. Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 5-16

4.3 Database Anomalies Database Anomalies are the flaws in databases which occurs because of poor database design and storing everything in a single flat database. They are problems that are caused usually by redundant data. Types of database anomalies are Insertion anomaly, Deletion anomaly, and Update anomaly. Insertion Anomaly - occurs when the entire primary key is not known and the database cannot insert a new record properly. This would violate entity integrity. It can be avoided by using a sequence number for the primary key. Deletion Anomaly - happens when a record is deleted that results in the loss of other related data. It can be avoided by normalizing tables in a database to minimize their dependency. Update Anomaly - occurs when a change to one attribute value causes the database to either contain inconsistent data or causes multiple records to need changing. It may be prevented by making sure tables are in third normal form.

Question sources

Database Fundamentals practice questions · Question 62

Question 127 | Q01388

In the course terminology, which labels can name a drawing of relational tables and their primary-key/foreign-key references? Select all that apply. ER Diagram means a conceptual entity/relationship drawing; Relational Schema means a textual table definition here.

A. ER Diagram

B. Database Diagram

C. Schema Diagram

D. Relational Schema

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer B, C: B. Database Diagram; C. Schema Diagram

A schema diagram draws relational tables and their key references. Database diagram is also used for that depiction in the course terminology, so both labels qualify. The question explicitly distinguishes a conceptual ER drawing and a textual relational schema from the requested table-reference drawing. This makes the two accepted names compatible without treating every representation of a database as the same artifact.

Why each choice is right or wrong

A - Incorrect: A conceptual ER diagram represents entities and relationships rather than specifically depicting mapped table key references.

B - Correct: Database diagram can denote the table-and-key depiction also called a schema diagram.

C - Correct: A schema diagram explicitly draws tables and their primary/foreign-key references.

D - Incorrect: A relational schema defines structures and constraints but need not itself be a diagram.

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course.

Study-note section FDB06

> In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 27-31

> 3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Fundamental of Database Systems module | PDF page 14 | lines 2-3

> Relation Schema: A relation schema represents the name of the relation with its attributes (columns).

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity ● A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key ● Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Database Fundamentals practice questions · Question 63

Question 128 | Q01389

R(A,B,C) has atomic attributes and exactly the dependencies implied by A→B and B→C. A is the sole candidate key; B is not a superkey and C is nonprime. Moving upward through 1NF, 2NF, 3NF and BCNF, which listed normal form is the first that R fails?

A. BCNF

B. 1NF

C. 2NF

D. 3NF

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer D: D. 3NF

The atomic relation satisfies 1NF. Its only candidate key is the single attribute A, so no nonprime dependency can involve a proper part of that key, and 2NF holds. However, B→C has a determinant that is not a superkey and a dependent that is nonprime. This violates the stated 3NF test, so 3NF is the listed form it fails.

Why each choice is right or wrong

A - Incorrect: BCNF also fails, but 3NF is the earlier failed level in the stated hierarchy.

B - Incorrect: Atomic values satisfy 1NF.

C - Incorrect: A single-attribute candidate key has no nonprime partial-key dependency.

D - Correct: B→C violates 3NF because B is not a superkey and C is nonprime.

Primary note proof

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key. For Enrollment(SID,CID,SName,CTitle,Grade), assume key (SID,CID), SID→SName and CID→CTitle. Split into Student(SID,SName), Course(CID,CTitle) and Enrollment(SID,CID,Grade). Grade still depends on the whole pair. A relation whose every candidate key is single-attribute has no nonprime partial-key dependency. Higher forms inherit 2NF, so no partial dependencies establishes the minimum guaranteed stage only when 1NF is also assumed.

Study-note section FDB17

> Full dependency requires all determinant attributes; partial dependency holds through a proper subset.

Study-note section FDB21

With atomic values the highest form is 1NF, even though a transitive dependency also 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 15 | lines 25-28

Transitive Dependency - In mathematics and logic, a transitive relationship is a relationship of the following form: —If A → B, and if also B → C, then A → C.‖ Example: If X >Y and Y >Z then X >Z. Generalized way of describing transitive dependency is that:

Fundamental of Database Systems module | PDF page 16 | lines 2-4

If A functionally governs B, AND If B functionally governs C THEN _A‘ functionally governs _C‘, provided that neither C nor B determines A. A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Database Fundamentals practice questions · Question 64

Question 129 | Q01390

Consider the relation Sale(Date, Customer, Product, Vendor, VendorCity, SalesRep){Date, Customer, Product} is the composite candidate key and the following functional dependencies are also given:Vendor => VendorCity, Product => VendorWhat is the highest normal form of the sale relation?

A. 0NF

B. 1NF

C. 2NF

D. 3NF

Hint

Test determinants and prime attributes against the stated normal-form rules.

Answer B: 1NF

Assuming atomic values as customary for a relation, Product is a proper subset of the stated composite key and determines non-prime Vendor. Vendor→VendorCity adds another non-key dependency. Product→Vendor alone violates 2NF, so the highest supported form is 1NF. No stated dependency makes these non-prime attributes alternative keys.

Why each choice is right or wrong

A - Incorrect: 0NF is not warranted when attributes are atomic. (PDF pp. 15, 17.)

B - Correct: 1NF is the highest form under the explicit atomicity assumption. (PDF pp. 15, 17.)

C - Incorrect: 2NF fails because Vendor depends on Product alone. (PDF pp. 15, 17.)

D - Incorrect: 3NF requires 2NF, which already fails. (PDF pp. 15, 17.)

Primary note proof

Study-note section FDB21

> In Sale(Date,Customer,Product,Vendor,VendorCity,SalesRep), assume key (Date,Customer,Product), Product→Vendor and Vendor→VendorCity. Product is a proper subset of the key, so Vendor and VendorCity violate 2NF. With atomic values the highest form is 1NF, even though a transitive dependency also exists. 4NF addresses nontrivial multivalued dependencies with nonsuperkey determinants; 5NF addresses join dependencies. Do not confuse a multivalued dependency with a list stored inside a cell.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 15 | lines 11-17

> Partial Dependency - If an attribute which is not a member of the key is dependent on some part of the key (if we have composite/candidate key) then that attribute is partially functionally dependent on the key. Let {A, B} is the key set and C is no key attribute, then if {A, B} → C and [B → C or A → C] holds. We say C is partially functionally dependent on {A, B}. Partial dependency occurs when a non-prime attribute is functionally dependent on part of a candidate/composite key.

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Question sources

Database Fundamentals practice questions · Question 65

Question 130 | Q01391

Third normal form is based on the concept of _____

A. Closure Dependency

B. Transitive Dependency

C. Normal Dependency

D. Functional Dependency

Hint

Test determinants and prime attributes against the stated normal-form rules.

Answer B: Transitive Dependency

The defining extra condition of 3NF over 2NF is non-transitive dependence of non-prime attributes on keys. Functional dependency is the broader framework, but transitive dependency is the particular concept distinguishing this stage.

Why each choice is right or wrong

A - Incorrect: Closure dependency is not the stated criterion. (PDF pp. 18.)

B - Correct: Transitive dependency is the specific concept addressed. (PDF pp. 18.)

C - Incorrect: Normal dependency is not a defined dependency category. (PDF pp. 18.)

D - Incorrect: Functional dependency is broader and does not identify the distinctive 3NF condition. (PDF pp. 18.)

Primary note proof

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Study-note section FDB21

> With atomic values the highest form is 1NF, even though a transitive dependency also exists.

Study-note section FDB21

> Do not confuse a multivalued dependency with a list stored inside a cell.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Database Fundamentals practice questions · Question 66

Question 131 | Q01392

Which of the following statements does not change the state of the database?

A. Where

B. Select

C. Insert

D. Drop

Hint

Trace how SELECT forms its result columns and rows.

Answer B: Select

The read-only transaction example uses SELECT to retrieve rows from two tables. SELECT is the complete retrieval statement among these choices; WHERE is only a clause and not a standalone SQL statement. The question is read as asking about valid statements.

Why each choice is right or wrong

A - Incorrect: WHERE alone is not a complete statement, so it cannot be the requested statement. (PDF pp. 52, 26.)

B - Correct: SELECT is demonstrated in a read-only transaction. (PDF pp. 52, 26.)

C - Incorrect: INSERT is a manipulation command, not the demonstrated read-only request. (PDF pp. 52, 26.)

D - Incorrect: DROP is a structural DDL command, not the demonstrated retrieval. (PDF pp. 52, 26.)

Primary note proof

Study-note section FDB27

> SELECT Name,Address FROM Customer; produces only those mailing-label fields. SELECT * selects all columns; DISTINCT removes duplicate result rows; AS supplies an alias; ALL permits duplicates and does not mean all columns. SQL normally preserves duplicates without DISTINCT. A query reading table columns needs an appropriate FROM source. A literal-only query such as SELECT 1 may legally omit FROM.

Study-note section FDB27

> WHERE alone is a clause rather than a complete read-only statement, so it cannot be compared as an equivalent executable command.

Study-note section FDB27

Typical written clause order is SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ....

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 4-11

• SET TRANSACTION READ ONLY; • SELECT NAME, ADDRESS FROM WORKERS; • SELECT MANAGER, ADDRESS FROM PLACES; • COMMIT; • Freezes the data for the user in both tables before either select retrieves any rows, so that changes that occur concurrently will not show up • Commits before and after ensure any uncompleted transactions are finish, and then release the frozen data when done

Fundamental of Database Systems module | PDF page 26 | lines 7-11

7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Database Fundamentals practice questions · Question 67

Question 132 | Q01500

A relation is in 1NF and has no nonprime attribute partially dependent on any candidate key. What is the strongest normal form guaranteed by these facts alone?

A. First

B. BCNF

C. Third

D. Second

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer D: D. Second

The two stated conditions are exactly the requirements for 2NF: 1NF plus the absence of nonprime partial dependencies on candidate keys. First normal form is also true but is a weaker guarantee. Third normal form and BCNF need additional conditions on other functional dependencies. The relation might satisfy those higher forms, but the given facts alone establish only 2NF.

Why each choice is right or wrong

A - Incorrect: 1NF is given, but 2NF is the stronger guaranteed form.

B - Incorrect: BCNF additionally requires all nontrivial determinants to be superkeys.

C - Incorrect: 3NF needs further checks on determinants and prime dependents.

D - Correct: The conditions directly establish 2NF as the strongest guaranteed form.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells. It does not mean merely declaring a primary key. A mapped ER design that correctly splits composite and multivalued structures gives ordinary 1NF tables; ER drawing alone guarantees no higher form.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB20

> A relation whose every candidate key is single-attribute has no nonprime partial-key dependency.

Related reading (context, not primary note proof)

Study-note section FDB21

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of ‗R' are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Mock exam 1 · Question 16

Question 133 | Q01502

The minimal set of super key is called

A. Foreign key

B. None of the choices is correct

C. Candidate key

D. Primary key

Hint

Test uniqueness and minimality separately when classifying keys.

Answer C: Candidate key

A candidate key is a minimal superkey. Minimal means no removable attribute while retaining uniqueness, not necessarily the fewest attributes among all keys.

Why each choice is right or wrong

A - Incorrect: Foreign key links relations, not the minimal-superkey definition. (PDF pp. 11.)

B - Incorrect: A matching definition is offered. (PDF pp. 11.)

C - Correct: Candidate key is the exact definition. (PDF pp. 11.)

D - Incorrect: Primary key is the selected candidate key rather than the category of every minimal superkey. (PDF pp. 11.)

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Mock exam 1 · Question 18

Question 134 | Q01581

Assume every foreign-key column is NOT NULL and references a declared primary key. Which statement expresses referential integrity under these assumptions?

A. Every primary-key value must match a primary-key value in an associated table

B. Every primary-key value must match a foreign-key value in an associated table

C. Every foreign-key value must match a primary-key value in an associated table

D. Every foreign-key value must match a foreign-key value in an associated table

Hint

Match the validity requirement to the appropriate constraint definition.

Answer C: C. Every foreign-key value must match a primary-key value in an associated table

A foreign key is a reference from a child row to an eligible key in the parent. Because the question makes every reference non-NULL and targets primary keys, each foreign-key value must match a primary-key value. A parent row need not have children, and primary keys do not themselves have to reference another table. Merely matching a foreign-key column is not sufficient.

Why each choice is right or wrong

A - Incorrect: A primary key can exist without referencing another table.

B - Incorrect: A parent can have no child references.

C - Correct: Every non-NULL child reference must match its targeted parent primary key.

D - Incorrect: The target must be an eligible referenced key, not an arbitrary foreign-key column.

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 49 | lines 2-3

- Sometimes a special NULL value may be used to create child entries without a parent or with a —dummy‖ parent

Question sources

Mock exam 3 · Question 13

Question 135 | Q01582

Which of the following is generally used for performing tasks like creating the structure of the relations, deleting relation?

A. DML(Data Manipulation Language)

B. Query

C. Relational Schema

D. DDL(Data Definition Language)

Hint

Classify each SQL command by the kind of change it performs.

Answer D: DDL(Data Definition Language)

Creating/deleting relation structure falls under DDL's structural command category. It is distinguished from data manipulation and from the schema description itself.

Why each choice is right or wrong

A - Incorrect: DML is the data-operation category. (PDF pp. 26.)

B - Incorrect: Query is not the schema-definition category. (PDF pp. 26.)

C - Incorrect: A relational schema is a description, not a command language. (PDF pp. 26.)

D - Correct: DDL is the named structural-language category. (PDF pp. 26.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Mock exam 3 · Question 14

Question 136 | Q01583

Which of the following is used in the application programs to request data from the database management system?

A. Data Manipulation language

B. Data Definition Language

C. Data Control Language

D. All choices are correct

Hint

Classify each SQL command by the kind of change it performs.

Answer A: Data Manipulation language

DML is the offered language category that includes retrieval requests such as SELECT. DDL handles definition, while authorization is separate from retrieval commands.

Why each choice is right or wrong

A - Correct: DML includes retrieval. (PDF pp. 26, 24.)

B - Incorrect: DDL is definition rather than ordinary data requests. (PDF pp. 26, 24.)

C - Incorrect: Access-control language is not the category that retrieves the requested data. (PDF pp. 26, 24.)

D - Incorrect: DDL and control are not all retrieval categories. (PDF pp. 26, 24.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 24 | lines 5-9

Relational algebra is the basic set of operations for the relational model. These operations enable a user to specify basic retrieval requests (or queries). The result of an operation is a new relation, which may have been formed from one or more input relations. The algebra operations thus produce new relations. Since each operation returns a relation, operations can be composed (Algebra is —closed‖.)

Question sources

Mock exam 3 · Question 15

Question 137 | Q01331B

Which single choice completely identifies the relational DBMS products in the list?

A. Oracle Database

B. Microsoft SQLServer

C. IBM Db2

D. All of the mentioned

Hint

Compare the data models by how they organize records and relationships.

Answer D: D. All of the mentioned

Oracle Database, Microsoft SQL Server and IBM Db2 are all relational DBMS products named in the note. Replacing the vendor-only IBM label with IBM Db2 makes every product entry well-defined. The question asks for the complete set, so All of the mentioned is the best answer. Each individual product is genuinely relational but identifies only part of the full set.

Why each choice is right or wrong

A - Incorrect: Oracle Database is relational, but this alone is an incomplete set.

B - Incorrect: SQL Server is relational, but this alone is an incomplete set.

C - Incorrect: IBM Db2 is relational, but this alone is an incomplete set.

D - Correct: The aggregate choice includes all three relational products.

Primary note proof

Study-note section FDB02

PostgreSQL, MySQL, Oracle Database, Microsoft SQL Server and IBM Db2 are relational DBMS products. MongoDB is a document DBMS. Java is a programming language. IBM alone names a vendor, not a database product. Distinguish a vendor such as IBM from a specific database product.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-16

Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server.

Fundamental of Database Systems module | PDF page 13 | lines 18-27

3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Database Fundamentals practice questions · Question 6

Question 138 | Q01388B

In ER diagram Diamond/Ellipses represent

A. Relationship sets/Attributes

B. Entity sets/Attributes

C. Attributes/Relationships sets

D. Relationship sets/Entity sets

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer A: Diamonds represent relationship sets; ellipses represent attributes.

FDB06 maps rectangles to entity types, diamonds to relationships and ellipses to attributes in Chen notation. The question asks for the diamond/ellipse pair in that order. A preserves that order and meaning; the other choices either substitute entity sets or reverse the symbol meanings. A diagram-specific legend would override the generic convention.

Why each choice is right or wrong

A - Correct: The two shapes map to relationships and attributes respectively in Chen notation.

B - Incorrect: Entity types use rectangles, so a diamond does not represent the first item listed.

C - Incorrect: This reverses the requested meanings: diamonds indicate relationships, while ellipses indicate attributes.

D - Incorrect: The first meaning fits, but ellipses represent attributes rather than entity sets.

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 10 | lines 3-13

> An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). In ER diagrams, an entity is placed in a rectangle. Each entity has attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities in combination with one of their attribute values. A weak entity type has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity. In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.

Fundamental of Database Systems module | PDF page 11 | lines 13-15

> A relationship type represents the association between entity types. For example, _Enrolled in‘ is a relationship type that exists between entity Student and Course. In ER diagram, relationship is represented by a diamond and connecting the entities with lines.

Question sources

Database Fundamentals practice questions · Question 63

Question 139 | Q01388C

A relation is in 2NF if:

A. All the values of non-key attributes are dependent fully on the candidate key.

B. Any non-key attribute that are dependent on only part of the candidate key should be moved to another relation where the partial key is the actual full key.

C. It must be already in the 1NF.

D. All of the above.

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer D: All of the above.

Second normal form requires 1NF and full dependence of each non-prime attribute on candidate keys; partial dependencies must be moved to another relation. A states the criterion, B the repair, and C the prerequisite; D collects all three. A alone is not a complete definition without C.

Why each choice is right or wrong

A - Incorrect: Full dependency is a required 2NF property. (PDF pp. 17.)

B - Incorrect: Moving a partially dependent attribute to the relation where its determinant is the whole key is the stated repair. (PDF pp. 17.)

C - Incorrect: 1NF is expressly required. (PDF pp. 17.)

D - Correct: All three agree with the module's definition and repair rule. (PDF pp. 17.)

Primary note proof

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key. For Enrollment(SID,CID,SName,CTitle,Grade), assume key (SID,CID), SID→SName and CID→CTitle. Split into Student(SID,SName), Course(CID,CTitle) and Enrollment(SID,CID,Grade). Grade still depends on the whole pair. A relation whose every candidate key is single-attribute has no nonprime partial-key dependency. Higher forms inherit 2NF, so no partial dependencies establishes the minimum guaranteed stage only when 1NF is also assumed.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 10-15

> Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Question sources

Database Fundamentals practice questions · Question 63

Question 140 | SM0017

Which one of the following is NOT a single level ordered index?

A. Secondary index

B. Primary index

C. Clustering index

D. Multilevel index

Hint

Compare index organization with its access benefits and maintenance costs.

Answer D: Multilevel index

In this single-level ordered-index classification, the listed types are primary, clustering and secondary indexes. A multilevel index is the exception to that listed single-level classification.

Why each choice is right or wrong

A - Incorrect: Secondary is listed as single level. (PDF pp. 21.)

B - Incorrect: Primary is listed as single level. (PDF pp. 21.)

C - Incorrect: Clustering is listed as single level. (PDF pp. 21.)

D - Correct: Multilevel is outside the specified single-level group. (PDF pp. 21.)

Primary note proof

Study-note section FDB25

> In ordered-file indexing terminology, a primary ordered index is on the file's ordering key and can have one entry per page, hence sparse. A clustering index is on a nonunique ordering field, often with an entry per distinct value. Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists. A dense index represents each search value; sparse indexes rely on ordered data between entries. Multilevel indexes index lower index levels. B/B+ trees support ordered and range search; hashing targets equality; bitmap indexes suit some analytic workloads. No one type is universally best.

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 19-24

> Single Level Ordered Indexes Here it is assumed that a file already exists with some primary organization like unordered, ordered, or hashed organizations. For a file with a given record structure consisting of several fields, an index access structure is usually defined on a single field of a file, called an indexing field (or indexing attribute). Types of Single Level Ordered Indexes are Primary, Clustering and Secondary.

Question sources

MOE Computer Science Exit Exam · Question 5-1

National Computer Science Exit Exam · Question 32

National Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 42

Question 141 | SM0023

Among the following, which one best describes a Database Management System (DBMS)?

A. It is responsible for creating and maintaining a database

B. It can help users to manage computer's activity

C. It can assist process images

D. It can assist you to process document

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer A: It is responsible for creating and maintaining a database

A DBMS is software facilitating definition, construction, manipulation and sharing of databases. This directly matches creating and maintaining a database.

Why each choice is right or wrong

A - Correct: Matches the module's DBMS definition. (PDF pp. 5.)

B - Incorrect: Managing general computer activity describes a broader different role. (PDF pp. 5.)

C - Incorrect: Image processing is not the database-management role. (PDF pp. 5.)

D - Incorrect: Document processing is not the database-management definition. (PDF pp. 5.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-16

> Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server.

Question sources

MOE Computer Science Exit Exam · Question 6-3

National Computer Science Exit Exam · Question 82

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 43

Question 142 | SM0034

Given the schemas: BOOK (Book_id, Title, Publisher_name), BOOK_AUTHOR (Book_id, Author_name), PUBLISHER (Name, Address, Phone). Which one of the following SQL is correct to delete the
publisher named 'XWZ' from the PUBLISHER table?

A. DELETE FROM BOOK_AUTHOR WHERE Address = 'XWZ'

B. DELETE FROM PUBLISHER WHERE Name = 'XWZ'

C. DELETE FROM PUBLISHER WHERE Address = 'XWZ'

D. DELETE FROM BOOK WHERE Name = 'XWZ'

Hint

Track which rows each data-changing command affects.

Answer B: DELETE FROM PUBLISHER WHERE Name = 'XWZ'

The module illustrates DELETE FROM HumanResources.JobCandidate WHERE JobCandidateID = 13 (PDF p.77). Substitute the requested relation PUBLISHER and its identifying Name predicate: DELETE FROM PUBLISHER WHERE Name = 'XWZ'. The stem's schemas distinguish Name from Address and omit both from BOOK_AUTHOR. This identifies the requested statement; successful execution may additionally depend on existing referencing rows and the deletion policy (PDF pp.48-49).

Why each choice is right or wrong

A - Incorrect: BOOK_AUTHOR is the wrong target and has no Address column in the supplied schema (compare the target/predicate form on PDF p.77).

B - Correct: PUBLISHER is the target and Name is the requested name filter, matching the module's DELETE FROM ... WHERE example (PDF p.77).

C - Incorrect: Filtering Address does not select by publisher name, despite a valid target table (PDF p.77's predicate pattern).

D - Incorrect: BOOK is the wrong target and has no Name column in the supplied schema (PDF p.77's target/predicate pattern).

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB29

DELETE FROM Publisher WHERE Name = 'XWZ'; deletes matching publisher rows, subject to reference policies.

Study-note section FDB29

Both clear relation entries and delete rows describe that effect.

Related reading (context, not primary note proof)

Study-note section FDB16

Study-note section FDB15

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 77 | lines 14-18

Example 1: Commit Transaction BEGIN TRANSACTION; DELETE FROM HumanResources.JobCandidate WHERE JobCandidateID = 13; COMMIT;

Fundamental of Database Systems module | PDF page 48 | lines 13-17

iv. Referential Integrity ● A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key ● Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 49 | lines 4-11

Deletion rules ● A row should not be deleted from the referenced table (parent) if there are matching rows in the referencing table (child) ● Three ways to handle this ◦ Restrict -- disallow the delete ◦ Nullify -- reset the foreign keys in the child to some NULL or dummy value ◦ Cascade -- Delete all rows in the child where there is a foreign key matching the key in the parent row being deleted

Question sources

MOE Computer Science Exit Exam · Question 10-3

National Computer Science Exit Exam · Question 9

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 44

Question 143 | SM0048

What does program data independence in the database approach entail?

A. The application may be affected when data changes

B. Data and applications are defined separately

C. Programs can create redundant data independently

D. Both program and data are defined together

Hint

Trace which schema levels a proposed change affects.

Answer B: Data and applications are defined separately

Program/data insulation separates application definitions from data definitions, allowing schema changes without corresponding higher-level changes under the relevant independence conditions.

Why each choice is right or wrong

A - Incorrect: Says changes necessarily affect applications, contrary to insulation. (PDF pp. 5, 6, 8.)

B - Correct: Separate definitions express program/data insulation. (PDF pp. 5, 6, 8.)

C - Incorrect: Creating redundant data independently is not the defined independence. (PDF pp. 5, 6, 8.)

D - Incorrect: Defining both together contradicts insulation. (PDF pp. 5, 6, 8.)

Primary note proof

Study-note section FDB04

> Physical data independence permits changing internal storage without changing the conceptual schema, for example adding an index. Logical data independence permits conceptual changes while preserving affected external views and application interfaces, for example splitting a table while maintaining a compatible view. It is harder because applications depend on logical meaning. Data abstraction hides irrelevant details; program-data insulation separates data definitions from application code.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 22-25

> 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.

Fundamental of Database Systems module | PDF page 6 | lines 2-4

Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Fundamental of Database Systems module | PDF page 8 | lines 8-15

2.4 Data Independence The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. The two types of data independence: 1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. 2. Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas do not need be changed as well.

Question sources

MOE Computer Science Exit Exam · Question 14-3

National Computer Science Exit Exam · Question 50

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 45

Question 144 | SM0071

Which statements are NOT correct descriptions of a fan trap or its general remedy in an ER diagram? Select all that apply. Treat B as a warning pattern, not a sufficient condition.

A. It occurs when there is no pathway between related entities

B. Highly likely when more than two one-to-many relationships fan out from an entity

C. When there is ambiguous pathway between some entity occurrences in related entity types

D. It can be resolved by readjusting the model so that no one-to-many relationships fan out from an entity

Hint

Trace the actual relationship paths and check whether the requested association follows.

Answer A, D: A. It occurs when there is no pathway between related entities; D. It can be resolved by readjusting the model so that no one-to-many relationships fan out from an entity

A fan trap concerns an ambiguous association inferred through branches from a common entity. A missing or broken path is instead a chasm-type issue. Also, legitimate one-to-many branches need not all be eliminated: the model should record the actual missing association. Multiple branches can warn of a fan trap, and ambiguity is its defining issue, so B and C are not false.

Why each choice is right or wrong

A - Correct: This is false about fan traps: a missing or broken path describes a chasm-type problem.

B - Incorrect: Several 1:N branches from a common entity can create the ambiguous association characteristic of fan traps.

C - Incorrect: Ambiguity about associations between entity occurrences is the defining fan-trap problem.

D - Correct: This is false as a general remedy: valid 1:N fans may remain after explicitly modeling the required association.

Primary note proof

Study-note section FDB14

> A fan trap occurs when two 1:N relationships fan from a common entity and a path suggests an association it cannot determine. Department→Employee and Department→Project do not reveal which employee works on which project; joining through Department pairs every employee with every project in that department. Record WorksOn explicitly when that is the fact needed.

Study-note section FDB14

> A missing path describes a chasm, not the defining ambiguity of a fan.

Study-note section FDB14

> Multiple 1:N relationships are legitimate; eliminating all fans is not a universal repair.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 27-31

> 3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Fundamental of Database Systems module | PDF page 11 | lines 24-31

> Cardinality The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types: One to one (1-1) - When each entity in each entity set can take part only once in the relationship, the cardinality is one to one. One to many(1-M)- When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.

Question sources

MOE Computer Science Exit Exam · Question 20-3

National Computer Science Exit Exam · Question 24

National Computer Science Exit Exam

MOE Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 46

Question 145 | SM0086

In which situation does using a DBMS become costly and not recommended?

A. In embedded systems having too small storage space

B. In situation where more than one user should access the database

C. To deny access to unauthorized users

D. To control redundancy

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer A: Severely storage-constrained embedded systems can make a DBMS too costly.

FDB01 says storage, administration and complexity may outweigh DBMS benefits in a tiny fixed application, while also recognizing lightweight embedded DBMSs. A is therefore the listed situation where resource constraints can make a DBMS inappropriate. The other scenarios are central reasons for using a DBMS: sharing, authorization and redundancy control. Embedded use alone does not forbid a DBMS.

Why each choice is right or wrong

A - Correct: Severely limited storage can make DBMS overhead unjustified, though lightweight embedded DBMSs may still fit.

B - Incorrect: Controlled multiuser access is a benefit supplied by database management rather than a reason to avoid it.

C - Incorrect: Authorization helps deny unauthorized access and is a reason to use DBMS protection.

D - Incorrect: Controlling redundancy is a benefit of shared database management rather than a contraindication.

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

Complexity, administration, storage and licensing can outweigh benefits for a tiny fixed application; lightweight embedded DBMSs also exist.

Study-note section FDB01

It controls redundancy rather than guaranteeing zero duplication.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-25

Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server. 1.2 Database System versus File System The traditional file processing system is file-directory structure supported by a conventional operating system. A file system organization of data lucks a number of major features of a database system, such as Data redundancy and inconsistency, Difficulty in accessing data, Data isolation, Integrity problems, Atomicity problems, Concurrent access anomalies. 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.

Fundamental of Database Systems module | PDF page 19 | lines 4-10

5.1 Introduction To translate the logical description of data into technical specifications for storing and retrieving data, a physical database design is needed. The goal is to create a design for storing and retrieving data that will provide adequate performance and insure database integrity, security, and availability. Some decisions need to be made on the physical database design. They are Attribute data types, data/file storages, File organizations and Indexes.

Question sources

MOE Computer Science Exit Exam · Question 24-3

National Computer Science Exit Exam · Question 72

National Computer Science Exit Exam

Computer Science Exit Exam practice collection · Question 47

Question 146 | SM0208

Indexes in database tables speed up the _____ operations.

A. Read operations

B. Delete operations

C. Update operations

D. Write operations

Hint

Compare index organization with its access benefits and maintenance costs.

Answer A: Read operations

Indexes are auxiliary access structures explicitly used to speed record retrieval for search conditions. This is a read-performance purpose; it does not claim that maintaining indexes makes every change operation faster.

Why each choice is right or wrong

A - Correct: Read retrieval is the stated purpose. (PDF pp. 21.)

B - Incorrect: Deletion may require maintenance; it is not the stated general benefit. (PDF pp. 21.)

C - Incorrect: Updating data is not the stated general purpose. (PDF pp. 21.)

D - Incorrect: Writing is not the stated retrieval purpose. (PDF pp. 21.)

Primary note proof

Study-note section FDB25

> An index stores search keys with pointers or row identifiers. It often speeds selective retrieval but consumes space and must be maintained on writes. A full scan can win for a tiny table or a query returning most rows. Indexes do not themselves establish referential integrity.

Study-note section FDB25

> Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists.

Study-note section FDB25

> A dense index represents each search value; sparse indexes rely on ordered data between entries.

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 5-9

> 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.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 3-1

Computer Science Exit Exam practice collection · Question 56

Question 147 | SM0226

An e-commerce database already has a suitable normalized schema. Customer-order queries repeatedly select a small fraction of rows by a search key. Which technique directly adds a maintained search path for those lookups?

A. Normalization

B. Denormalization

C. Partitioning

D. Indexing

Hint

Compare design techniques by the specific problem each solves.

Answer D: D. Indexing

Indexing stores search keys and pointers that support selective retrieval without scanning every row. That directly meets the requested search-path improvement. Normalization addresses dependency anomalies, denormalization stores redundant or precomputed facts, and partitioning divides a dataset. Those techniques can be useful elsewhere, but the specified task is to add an access path to the existing normalized schema.

Why each choice is right or wrong

A - Incorrect: Normalization organizes dependencies; it does not add the requested search path.

B - Incorrect: Denormalization repeats or precomputes facts rather than directly adding an index path.

C - Incorrect: Partitioning divides the dataset rather than adding the requested maintained search-key structure.

D - Correct: Indexing directly supplies the selective lookup search path.

Primary note proof

Study-note section FDB38

> Normalization reduces dependency-driven anomalies. Indexing supplies search paths. Denormalization deliberately repeats or precomputes facts to reduce selected read costs, increasing maintenance work. Partitioning divides rows or columns; replication creates copies to improve locality or availability while complicating updates. These techniques can coexist. An e-commerce or gaming label alone does not select one.

Study-note section FDB25

> An index stores search keys with pointers or row identifiers.

Study-note section FDB25

> It often speeds selective retrieval but consumes space and must be maintained on writes.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-27

> 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship.

Fundamental of Database Systems module | PDF page 21 | lines 5-9

> 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.

Fundamental of Database Systems module | PDF page 117 | lines 18-29

> 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

Question sources

Computer Science Exit Exam · Sene 2016 · Question 9-1

Computer Science Exit Exam practice collection · Question 68

Question 148 | SM0251

Which SQL command is used to modify existing data in a table?

A. ALTER

B. UPDATE

C. MODIFY

D. CHANGE

Hint

Track which rows each data-changing command affects.

Answer B: UPDATE

UPDATE changes existing row values using a target table, SET assignments and an optional WHERE predicate. ALTER instead changes an object's definition. The bare words MODIFY and CHANGE do not form the UPDATE statement required for changing stored row values in the stated SQL forms. They should not be treated as synonyms for the command merely because they describe the task in everyday language. B names the actual row-update command.

Why each choice is right or wrong

A - Incorrect: Incorrect: ALTER changes schema definitions rather than stored row values.

B - Correct: Correct: UPDATE changes existing rows using SET and, when needed, WHERE.

C - Incorrect: Incorrect: MODIFY is not the UPDATE command required for the stated row-value operation.

D - Incorrect: Incorrect: CHANGE is not the UPDATE command required for the stated row-value operation.

Primary note proof

Study-note section FDB29

> UPDATE Book SET Price = 600 WHERE ID = 'B001'; changes matching rows. UPDATE without WHERE targets every row.

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 7-11

> 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 54 | lines 16-18

> • <privileges> is a list of SELECT <columns>, INSERT <columns>, DELETE, and UPDATE <columns>, or simply ALL

Fundamental of Database Systems module | PDF page 55 | lines 2-5

> UPDATE(Salary) ON Employee TO Finance The user ‗Finance' can view the entire Employee table, and can change Salary values, but cannot change other values or pass on their privilege

Question sources

Computer Science Exit Exam · Sene 2016 · Question 16-4

Computer Science Exit Exam practice collection · Question 54

Question 149 | SM0252

Which is the first normal form in the usual normalization sequence that requires atomic values and no repeating groups?

A. Second Normal Form (2NF)

B. Third Normal Form (3NF)

C. Boyce-Codd Normal Form (BCNF)

D. First Normal Form (1NF)

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer D: First Normal Form (1NF)

First normal form is the first stage requiring atomic values and no repeating groups in cells. It does not by itself remove partial or transitive functional dependencies. Second normal form adds the full-dependency condition, third normal form adds the determinant/prime-attribute rule, and BCNF requires every nontrivial determinant to be a superkey. These stronger forms inherit atomicity, but the question asks for the first stage defining it.

Why each choice is right or wrong

A - Incorrect: Incorrect: 2NF inherits atomicity and additionally excludes nonprime partial dependencies; it is a later stage.

B - Incorrect: Incorrect: 3NF inherits the earlier atomicity condition and adds a stronger dependency test.

C - Incorrect: Incorrect: BCNF adds the superkey-determinant rule and is not the first stage requiring atomicity.

D - Correct: Correct: 1NF is the first stage defining atomic values and no repeating groups.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 17-1

Question 150 | SM0254

What is the purpose of normalization in database design?

A. To increase data redundancy

B. To decrease data integrity

C. To eliminate data duplication

D. To simplify data retrieval

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer C: To eliminate data duplication

Normalization systematically decomposes relations to eliminate or reduce duplication and insertion/update/deletion anomalies. Eliminate is understood as the design goal, not a claim that every possible repeated value or necessary key reference disappears.

Why each choice is right or wrong

A - Incorrect: Increasing redundancy reverses the goal. (PDF pp. 16.)

B - Incorrect: Decreasing integrity reverses the anomaly-prevention goal. (PDF pp. 16.)

C - Correct: Eliminating duplicated data is directly stated. (PDF pp. 16.)

D - Incorrect: Simplifying retrieval is not the stated normalization purpose and does not follow automatically from decomposition. (PDF pp. 16.)

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Study-note section FDB19

> Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Related reading (context, not primary note proof)

Study-note section FDB38

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-29

> 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship. Normal Forms are maintenance rules that are used to eliminate or reduce the redundancy in database relations. These normal forms are 1NF (First Normal Form), 2NF (Second Normal Form)

Variant 1 | Alternative version 1

What is the purpose of normalization in database design?

A. To reduce redundancy and improve data integrity

B. To speed up query performance

C. To create complex relationships between tables

D. To encrypt sensitive data

Answer A: To reduce redundancy and improve data integrity

Normalization systematically decomposes relations to eliminate or reduce duplication and insertion/update/deletion anomalies. Eliminate is understood as the design goal, not a claim that every possible repeated value or necessary key reference disappears.

Why each choice is right or wrong

A - Correct: Reducing redundancy and preventing anomalies is the normalization goal (PDF p.16).

B - Incorrect: Query performance is not the stated normalization purpose; decomposition can require additional joins.

C - Incorrect: Relationships support decomposition but creating complexity is not its goal.

D - Incorrect: Encryption is a security mechanism, not normalization.

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Related reading (context, not primary note proof)

Study-note section FDB38

Question sources

Computer Science Exit Exam · Sene 2016 · Question 17-3

Computer Science MCQ practice collection · Question 131

Computer Science MCQ practice collection · Question 152

Computer Science Exit Exam practice collection · Question 53

Question 151 | SM0262

Which of the following IS NOT a type of database management system (DBMS)?

A. PostgreSQL

B. MongoDB

C. Java

D. MySQL

Hint

Compare the data models by how they organize records and relationships.

Answer C: Java is a programming language, not a DBMS.

The question uses type loosely to list named systems. FDB02 identifies PostgreSQL and MySQL as relational DBMS products and MongoDB as a document DBMS. Java is explicitly a programming language. Its ability to connect to databases does not make it a database management system, so C alone fits the negative question.

Why each choice is right or wrong

A - Incorrect: PostgreSQL is a relational DBMS product, so it does not satisfy the negative question.

B - Incorrect: MongoDB is a document DBMS, even though it is not a relational system.

C - Correct: Java is a programming language rather than a database management system.

D - Incorrect: MySQL is a relational DBMS product and therefore is not the requested exception.

Primary note proof

Study-note section FDB02

> PostgreSQL, MySQL, Oracle Database, Microsoft SQL Server and IBM Db2 are relational DBMS products. MongoDB is a document DBMS. Java is a programming language. IBM alone names a vendor, not a database product. Distinguish a vendor such as IBM from a specific database product.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-16

> Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 20-2

Computer Science Exit Exam practice collection · Question 57

Question 152 | SM0273

The E-R data model is based on a perception of the real world that consists of a set of basic objects called _____.

A. Objects

B. Classes

C. Entities

D. Attributes

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer C: Entities

The ER model uses entities, attributes and relationships; the actual physical/conceptual objects are entities. Objects is a generic word, while entities is the model-specific category requested.

Why each choice is right or wrong

A - Incorrect: Objects is the generic description, not the requested ER term. (PDF pp. 9, 10.)

B - Incorrect: Classes are not the basic-object term in this ER definition. (PDF pp. 9, 10.)

C - Correct: Entities is the named category for those real-world objects. (PDF pp. 9, 10.)

D - Incorrect: Attributes are properties of entities rather than the objects themselves. (PDF pp. 9, 10.)

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 3-6

> 3.1 Database Modeling High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships.

Fundamental of Database Systems module | PDF page 10 | lines 3-8

> An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). In ER diagrams, an entity is placed in a rectangle. Each entity has

attributes, the particular properties that describe it. Entities that do not have key attributes of their own are called weak entity. The regular entity types that have a key attribute are called strong entity types. Weak entities are identified by being related to specific strong entities

Question sources

Computer Science Exit Exam · Sene 2016 · Question 24-1

Computer Science Exit Exam practice collection · Question 52

Question 153 | SM0275

What is the primary key in a relational database?

A. A unique identifier for each row in a table

B. A foreign key from another table

C. A key used for encryption

D. An index on a non-unique column

Hint

Test uniqueness and minimality separately when classifying keys.

Answer A: A unique identifier for each row in a table

A primary key is the selected candidate key used to identify tuples uniquely; its values must be unique and non-null. The key may consist of more than one attribute.

Why each choice is right or wrong

A - Correct: Uniquely identifies each row, consistent with the key definition. (PDF pp. 11, 14.)

B - Incorrect: A foreign key serves a reference role and need not be this table's primary key. (PDF pp. 11, 14.)

C - Incorrect: Encryption keys have a different security role. (PDF pp. 11, 14.)

D - Incorrect: A non-unique index does not meet key uniqueness. (PDF pp. 11, 14.)

Primary note proof

Study-note section FDB07

A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Study-note section FDB07

> If ID is unique, both {ID} and {ID,City} are superkeys, but only {ID} is minimal.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Fundamental of Database Systems module | PDF page 14 | lines 17-20

> Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 24-3

Computer Science Exit Exam practice collection · Question 51

Question 154 | SM0280

What is the role of a database administrator (DBA)?

A. Designing user interfaces for database applications

B. Developing front-end applications

C. Writing complex SQL queries

D. Managing and maintaining databases

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer D: Managing and maintaining databases

The DBA administers the database and DBMS, authorizes access, coordinates/monitors use and acquires resources. Those responsibilities fit management and maintenance.

Why each choice is right or wrong

A - Incorrect: User-interface design is not the defining DBA responsibility described. (PDF pp. 6.)

B - Incorrect: Front-end development is not the defining responsibility. (PDF pp. 6.)

C - Incorrect: Writing a difficult query may occur, but is not the administration role defined. (PDF pp. 6.)

D - Correct: Management and maintenance match the listed administration duties. (PDF pp. 6.)

Primary note proof

Study-note section FDB01

> The DBA manages access authorization, backup/recovery, availability, monitoring, tuning and routine maintenance, coordinating database use; designers define the schema and application developers build user-facing programs.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 10-16

> 1.4 Actors on the scene The people whose jobs involve the day-to-day use of a large database are called actors on the scene. In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 26-2

Computer Science Exit Exam practice collection · Question 55

Question 155 | SM0304

Which of the following IS NOT a type of database model?

A. Linear

B. Object-Oriented

C. Hierarchical

D. Relational

Hint

Compare the data models by how they organize records and relationships.

Answer A: Linear

The model taxonomy names object-oriented, hierarchical/network, and relational concepts. Linear is not among the stated database-model categories; this is a taxonomy question within that stated list, not a universal claim about every possible model name.

Why each choice is right or wrong

A - Correct: Linear is outside the supplied taxonomy. (PDF pp. 7, 13.)

B - Incorrect: Object-Oriented is explicitly a conceptual-model example. (PDF pp. 7, 13.)

C - Incorrect: Hierarchical is named in the model discussion. (PDF pp. 7, 13.)

D - Incorrect: Relational is explicitly defined as a database model. (PDF pp. 7, 13.)

Primary note proof

Study-note section FDB02

A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one 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 7 | lines 8-19

Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model. 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users. 3. representational (or implementation) data model provides concepts that may be easily understood by end users but that are not too far from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. The widely used relational data models are the network and hierarchical.

Fundamental of Database Systems module | PDF page 13 | lines 18-20

3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple.

Question sources

Computer Science Exit Exam · Sene 2016 · Question 34-1

Computer Science Exit Exam practice collection · Question 58

Question 156 | SS133

A relation has atomic values with no repeating groups, and no nonprime attribute partially depends on any candidate key. What is the strongest normal form guaranteed by these facts alone?

A. 4NF

B. 2NF

C. 1NF

D. 3NF

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer B: B. 2NF

Atomic values without repeating groups establish 1NF. Adding the absence of nonprime partial dependencies establishes 2NF. Higher normal forms have additional requirements involving nonkey determinants or multivalued dependencies, and none of those requirements follows from the stated facts. Thus 2NF is the strongest guaranteed level, even though the relation could also happen to satisfy higher forms.

Why each choice is right or wrong

A - Incorrect: No restriction on nontrivial multivalued dependencies has been established.

B - Correct: 1NF plus the absence of nonprime partial dependencies guarantees 2NF.

C - Incorrect: 1NF holds, but 2NF is the stronger guarantee.

D - Incorrect: 3NF requires additional determinant/prime-attribute conditions.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells. It does not mean merely declaring a primary key. A mapped ER design that correctly splits composite and multivalued structures gives ordinary 1NF tables; ER drawing alone guarantees no higher form.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB20

> Higher forms inherit 2NF, so no partial dependencies establishes the minimum guaranteed stage only when 1NF is also assumed.

Related reading (context, not primary note proof)

Study-note section FDB21

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science model exam collection · Question 40

Question 157 | SS142

Which one of the following is not among the characterizing features of database approach?

A. Sharing of data

B. @-tTM Application-data dependency

C. j Oc. Self-describing

D. Data abstraction

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer B: @-tTM Application-data dependency

Database systems support sharing, self-description and abstraction/insulation. Application-data dependency is the opposite of insulation and independence.

Why each choice is right or wrong

A - Incorrect: Sharing is expressly a characteristic. (PDF pp. 5, 6.)

B - Correct: Application-data dependency contradicts the stated insulation characteristic. (PDF pp. 5, 6.)

C - Incorrect: Self-description is expressly a characteristic. (PDF pp. 5, 6.)

D - Incorrect: Abstraction is expressly a characteristic. (PDF pp. 5, 6.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB04

> Data abstraction hides irrelevant details; program-data insulation separates data definitions from application code.

Study-note section FDB04

> Logical data independence permits conceptual changes while preserving affected external views and application interfaces, for example splitting a table while maintaining a compatible view.

Related reading (context, not primary note proof)

Study-note section FDB03

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 22-29

> 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.
> Self-describing nature - A fundamental characteristic of the database approach is that the database system contains not only the database itself (data) but also a complete definition or description of the database structure and constraints. This information is called meta-data, and it describes the structure of the database.

Fundamental of Database Systems module | PDF page 6 | lines 2-9

> Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data. Support of multiple views of the data - A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Sharing of data and multiuser transaction processing - A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database.

Question sources

Computer Science model exam collection · Question 49

Question 158 | SS146

"Database is a logically coherent collection of data with some inherent meaning" describes that

A. the entity should be physical object

B. number of attributes should be limited

C. there should be one primary key attribute

D. all attributes of the entity should be related

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer D: all attributes of the entity should be related

A database is a collection of related data with implicit meaning. D is the closest reading of coherence in the question: attributes should describe related subject matter, not arbitrary unrelated data. It does not mean every attribute functionally determines every other.

Why each choice is right or wrong

A - Incorrect: Entities may have conceptual existence too. (PDF pp. 5, 10, 11.)

B - Incorrect: No bound on the number of attributes is implied. (PDF pp. 5, 10, 11.)

C - Incorrect: A primary key may have multiple attributes; coherence does not require a single attribute. (PDF pp. 5, 10, 11.)

D - Correct: Related attributes are the offered interpretation consistent with meaningful related data. (PDF pp. 5, 10, 11.)

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB07

> A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB06

> An entity type describes common properties; its current collection of entities is an entity set.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 5-9

> Data is known facts that can be recorded and that have implicit meaning. Database is a collection of related data. For example, consider the names, telephone numbers, and addresses of the people you know. Nowadays, this data is typically stored in mobile phones, which have their own simple database software. This collection of related data with an implicit meaning is a database. A database can be defined as a self-describing collection of integrated records.

Fundamental of Database Systems module | PDF page 10 | lines 3-5

> An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course). In ER diagrams, an entity is placed in a rectangle.

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Computer Science model exam collection · Question 53

Question 159 | SS157

For HOTEL(Hotel_id INTEGER PRIMARY KEY, Hotel_Name STRING, Sub_city STRING), which statement correctly inserts one row? Assume Hotel_id = 1 is unused and no other constraints prevent the insert.

A. INSERT INTO HOTEL(Hotel_id,Hotel_Name,Sub_city) Values('Hilton', Yeka')

B. INSERT INTO HOTEL(Hotel_id,Hotel_Name,Sub_city) Values (1,'Hilton','Yeka')

C. INSERT INTO HOTEL(1,'Hilton','Yeka')

D. INSERT Values INTO HOTEL(Hotel_id,Hotel_Name,Sub_city) Values(1,'Hilton','Yeka')

Hint

Track which rows each data-changing command affects.

Answer B: INSERT INTO HOTEL(Hotel_id,Hotel_Name,Sub_city) Values (1,'Hilton','Yeka')

An INSERT must match each named column with a compatible value and follow INSERT INTO ... VALUES syntax. B supplies the integer identifier and the two text values in the named order; the stated unused-key assumption removes a possible primary-key conflict. A supplies too few values and has damaged string quoting. C puts values where column names belong and omits VALUES, while D places an extra Values token before INTO. These are incorrect statements, not unresolved alternatives.

Why each choice is right or wrong

A - Incorrect: Incorrect: three named columns require three compatible values; this text supplies only two and also has malformed quoting.

B - Correct: Correct: it supplies an unused integer key and two strings in the named column order with the required INSERT syntax.

C - Incorrect: Incorrect: literal values occupy the column-list position and the VALUES clause is absent.

D - Incorrect: Incorrect: INSERT Values INTO has an extra token before INTO and breaks the required INSERT structure.

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Related reading (context, not primary note proof)

Study-note section FDB07

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 77 | lines 19-24

> Example 2: Rollback Transaction CREATE TABLE ValueTable (id int); BEGIN TRANSACTION; INSERT INTO ValueTable VALUES(1); INSERT INTO ValueTable VALUES(2); ROLLBACK;

Fundamental of Database Systems module | PDF page 14 | lines 14-20

> Domain Constraints - Every domain (cell in the table or attribute value) must contain atomic values (smallest indivisible units). We perform data type check here, which means we need to assign a data type to a column and limit the values that it can contain. Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

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

5.2 Choosing Attribute Data Types CHAR - fixed-length character VARCHAR() - variable-length characters DATE - actual date BLOB - binary large object (good for graphics images, audio, sound clips, etc.) NUMERIC - positive/negative numbers Exact • SMALLINT → 16 bit • INT → 32 bit

Question sources

Computer Science model exam collection · Question 64

Question 160 | SS172

DORM(DormID, FloorNumber) has primary key DormID. STUDENT(IdNo, Name, Department, DormID) has primary key IdNo. Which table-constraint fragment correctly declares STUDENT.DormID as a reference to DORM? Capacity six is enforced separately; the surrounding CREATE TABLE syntax is supplied.

A. FOREIGN KEY (DormID) REFERENCED BY DORM(DormID)

B. DORM table is a referencing relation

C. FOREIGN KEY (DormID) REFERENCES DORM(DormID)

D. STUDENT table is a referenced relation

Hint

Trace keys and cardinalities through the integrated schema.

Answer C: C. FOREIGN KEY (DormID) REFERENCES DORM(DormID)

The student row stores the dorm identifier, making STUDENT the referencing child and DORM the referenced parent. The required clause names the child column and uses REFERENCES to identify Dorm's key. C has this syntax after the naming repair. The foreign key validates a dorm reference; it does not by itself enforce the six-student capacity rule, which the question handles separately.

Why each choice is right or wrong

A - Incorrect: REFERENCED BY is not the required child-to-parent clause.

B - Incorrect: DORM is the referenced parent, not the referencing child.

C - Correct: REFERENCES DORM(DormID) correctly targets the parent key.

D - Incorrect: STUDENT is the referencing child.

Primary note proof

Study-note section FDB39

Use Department(DeptID,Name), Student(SID,Name,DeptID,DormID), Course(CID,Title), Enrollment(SID,CID,Grade) and Dorm(DormID,FloorNumber). Declare all entity IDs primary keys; Student.DeptID and Student.DormID reference their parents; Enrollment has composite key (SID,CID) and both references. Make DormID nullable until assignment. Separate StudentPhone(SID,Phone) for multiple contacts. Normalize repeated department/course facts into their owning tables. A simple foreign key does not enforce the six-student capacity; use a

concurrency-safe trigger, allocation transaction or explicit slot design with six unique slots per dorm.

Study-note section FDB39

STUDENT is the referencing child; DORM is referenced.

Study-note section FDB39

Consistent names and an added Student.DormID column are required before execution.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 18-21

4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as foreign key in S that represents the other entity type participating in R. Include any attributes of the 1:N relation type as attributes of S.

Fundamental of Database Systems module | PDF page 48 | lines 13-17

iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 49 | lines 23-28

Referential Integrity • Ensures that dependent relationships in the data are maintained. In Oracle, for example: • CREATE TABLE table-name (attr1 attr-type PRIMARY KEY, attr2 attr-type NOT NULL, ..., attrM attr-type REFERENCES owner.tablename(attrname) ON DELETE CASCADE, ...

Question sources

Computer Science model exam collection · Question 79

Question 161 | SS189

Which one of the following is an appropriate sequence of database design processes?

A. Logical database design, Enterprise data modeling, Physical database design, Database implementation

B. Enterprise data modeling, Logical database design, Database implementation, Physical database design

C. Logical database design, Enterprise data modeling, Physical database design, Database implementation

D. Enterprise data modeling, Logical database design, Physical database design, Database implementation

Hint

Identify the design phase from its inputs and intended deliverable.

Answer D: Enterprise modeling → logical design → physical design → implementation.

FDB05 starts with business requirements and conceptual ER modeling, then maps to relations and constraints in logical design, chooses storage and indexes in physical design, and creates/loads actual objects in implementation. D preserves that order. The other choices either begin logical design before enterprise modeling or place implementation before the storage decisions it must implement.

Why each choice is right or wrong

A - Incorrect: Logical design should use the conceptual enterprise model rather than precede its creation.

B - Incorrect: Physical storage decisions precede implementation, not follow it in the stated design sequence.

C - Incorrect: This duplicates A and again places logical design before enterprise modeling.

D - Correct: It follows conceptual enterprise modeling, relational mapping, storage design and actual implementation.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 7-26

> 3.2 Phases of database design The phases of database development life cycle (DDLC) in the Database Management System (DBMS) are explained below. Requirement Analysis The most important step in implementing a database system is to find out what is needed i.e what type of a database is required for the business organization, daily volume of data, how much data needs to be stored in the master files etc. In order to collect all this information, a database analyst spends a lot of time within the business organization talking to people, end users and getting acquainted with the day-to-day process. Conceptual Database Design In this phase the database designers will make a decision on the database model that perfectly suits the organization‘s requirement. The database designers will study the documents prepared by the analysis in the requirement analysis stage and then start development of a system model that fulfils the needs. Logical database design (Data model mapping) The conceptual design is translated into internal model which includes mapping of all objects i.e design of tables, indexes, views, transaction, access privileges etc. Physical Design Physical design or Database implementation needs the

formation of special storage related constructs. These constructs consist of storage groups, table spaces, data files, tables etc.

Question sources

Computer Science model exam collection · Question 96

Question 162 | AM0132

Which of the following is a primary key in a relational database?

A. A unique identifier for a row in a table

B. A foreign key from another table

C. A non-unique attribute in a table

D. A combination of attributes in a table

Hint

Test uniqueness and minimality separately when classifying keys.

Answer A: A unique identifier for a row in a table

A primary key is selected from candidate keys to uniquely identify a row and must be unique/non-null. Multiple attributes alone do not make a set a primary key unless those requirements and selection hold.

Why each choice is right or wrong

A - Correct: Gives the essential identifying property. (PDF pp. 11, 14.)

B - Incorrect: A foreign key is a reference and is not necessarily the selected primary identifier. (PDF pp. 11, 14.)

C - Incorrect: Non-unique attributes cannot uniquely identify rows. (PDF pp. 11, 14.)

D - Incorrect: A combination can be a composite primary key, but an arbitrary combination is not sufficient. (PDF pp. 11, 14.)

Primary note proof

Study-note section FDB07

A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Study-note section FDB16

PRIMARY KEY implies uniqueness and non-NULL.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Fundamental of Database Systems module | PDF page 14 | lines 17-20

Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Question sources

Computer Science MCQ practice collection · Question 132

Question 163 | AM0133

What is the purpose of foreign keys in a relational database?

A. To establish relationships between tables

B. To store encrypted data

C. To enforce data types in columns

D. To improve query performance

Hint

Match the validity requirement to the appropriate constraint definition.

Answer A: To establish relationships between tables

Foreign keys link child records to matching parent keys and preserve consistency between related tables.

Why each choice is right or wrong

A - Correct: Establishes the described reference relationship. (PDF pp. 48, 14.)

B - Incorrect: Encryption is a security mechanism, not the FK purpose. (PDF pp. 48, 14.)

C - Incorrect: Data types are controlled by domain constraints. (PDF pp. 48, 14.)

D - Incorrect: Performance improvement is not the stated FK purpose. (PDF pp. 48, 14.)

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB07

> A primary key is one selected candidate key; remaining candidates are alternate keys.

Study-note section FDB07

> Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys.

Related reading (context, not primary note proof)

Study-note section FDB38

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 14 | lines 21-26

> Referential Integrity Constraints - The Referential integrity constraints is specified between two relations or tables and used to maintain the consistency among the tuples in two relations. Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Question sources

Computer Science MCQ practice collection · Question 133

Question 164 | AM0134

In the usual normalization sequence, which is the first normal form requiring every nonprime attribute to depend fully on every candidate key, rather than on a proper subset?

A. First normal form (1NF)

B. Second normal form (2NF)

C. Third normal form (3NF)

D. Boyce-Codd normal form (BCNF)

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer B: Second normal form (2NF)

Second normal form is the first stage requiring atomic relations to have no nonprime attribute partially dependent on any candidate key. First normal form addresses atomic values, so it does not impose that dependency rule. Third normal form and BCNF are stronger stages that inherit the absence of partial dependencies, but they are not the first stage introducing it. BCNF specifically requires every nontrivial functional-dependency determinant to be a superkey.

Why each choice is right or wrong

A - Incorrect: Incorrect: 1NF requires atomic values but does not exclude partial dependencies.

B - Correct: Correct: 2NF is the first stage excluding nonprime partial dependencies on candidate keys.

C - Incorrect: Incorrect: 3NF inherits 2NF and imposes its additional determinant/prime-attribute condition; it is not the first stage.

D - Incorrect: Incorrect: BCNF requires every nontrivial determinant to be a superkey; it is stronger than the first stage imposing the partial-dependency rule.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 10-15

> Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science MCQ practice collection · Question 134

Question 165 | AM0135

What defines a composite key in database design?

A. A key made up of two or more attributes

B. A foreign key that references multiple tables

C. A unique identifier for a column in a table

D. A key used for data encryption

Hint

Test uniqueness and minimality separately when classifying keys.

Answer A: A. A key made up of two or more attributes

Composite describes the number of attributes in a key: it contains two or more attributes. It does not mean the key has necessarily been selected as the primary key, nor does it refer to the number of tables reached by a reference. A column name alone is not a multicolumn identification key. Encryption uses a different meaning of key.

Why each choice is right or wrong

A - Correct: Two or more key attributes are the defining feature.

B - Incorrect: The number of referenced tables does not define composition.

C - Incorrect: A column identifier does not describe a multicolumn key.

D - Incorrect: An encryption key is a different concept.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> A composite-key example must include multiple columns; a single-column key is not composite.

Study-note section FDB07

A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 15 | lines 11-17

Partial Dependency - If an attribute which is not a member of the key is dependent on some part of the key (if we have composite/candidate key) then that attribute is partially functionally dependent on the key. Let {A, B} is the key set and C is no key attribute, then if {A, B} → C and [B → C or A → C] holds. We say C is partially functionally dependent on {A, B}. Partial dependency occurs when a non-prime attribute is functionally dependent on part of a candidate/composite key.

Fundamental of Database Systems module | PDF page 11 | lines 7-11

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Computer Science MCQ practice collection · Question 135

Question 166 | AM0136

What is the purpose of an index in a database?

A. To store large binary objects

B. To enforce referential integrity

C. To improve data retrieval performance

D. To encrypt sensitive data

Hint

Compare index organization with its access benefits and maintenance costs.

Answer C: To improve data retrieval performance

Indexes are auxiliary structures used to speed retrieval for search conditions, rather than stored binary values or security mechanisms.

Why each choice is right or wrong

A - Incorrect: BLOB is the named large-binary-object type. (PDF pp. 21, 19.)

B - Incorrect: Foreign keys handle referential links; indexes are defined for retrieval. (PDF pp. 21, 19.)

C - Correct: Exactly the stated retrieval-performance purpose. (PDF pp. 21, 19.)

D - Incorrect: Encryption is a separate security mechanism. (PDF pp. 21, 19.)

Primary note proof

Study-note section FDB25

> An index stores search keys with pointers or row identifiers. It often speeds selective retrieval but consumes space and must be maintained on writes. A full scan can win for a tiny table or a query returning most rows. Indexes do not themselves establish referential integrity.

Study-note section FDB23

> DATE represents a date and BLOB binary content such as images.

Study-note section FDB15

> Domain integrity restricts type and allowed values.

Related reading (context, not primary note proof)

Study-note section FDB38

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 5-9

> 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.

Fundamental of Database Systems module | PDF page 19 | lines 11-15

> 5.2 Choosing Attribute Data Types CHAR - fixed-length character VARCHAR() - variable-length characters DATE - actual date BLOB - binary large object (good for graphics images, audio, sound clips, etc.)

Question sources

Computer Science MCQ practice collection · Question 136

Question 167 | AM0137

What is a data dictionary in database design?

A. A database table that stores user information

B. A collection of tables in a database

C. A file system used to store database files

D. A repository of metadata about the database structure

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer D: A repository of database metadata.

A data dictionary describes the database rather than serving as ordinary application data storage. FDB03 says it records definitions, types, constraints and often privileges and optimizer statistics. User information may appear in an application table, but that is not the dictionary definition. Nor is a dictionary simply all tables or the physical file system.

Why each choice is right or wrong

A - Incorrect: Ordinary user records are application data, not the defining purpose of a metadata repository.

B - Incorrect: A collection of tables is not specifically the repository describing those tables and constraints.

C - Incorrect: The physical file system stores files; a dictionary records descriptive database metadata.

D - Correct: The dictionary is the repository describing database structure, data types and constraints.

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 26-29

> Self-describing nature - A fundamental characteristic of the database approach is that the database system contains not only the database itself (data) but also a complete definition or description of the database structure and constraints. This information is called meta-data, and it describes the structure of the database.

Question sources

Computer Science MCQ practice collection · Question 137

Question 168 | AM0138

Which of the following is an example of a database constraint?

A. UNIQUE

B. SELECT

C. INSERT

D. DELETE

Hint

Read each SQL constraint's scope and enforcement rule.

Answer A: UNIQUE

Uniqueness constrains distinct tuple identifiers, whereas SELECT, INSERT and DELETE are language operations in the DML list. UNIQUE is the offered constraint name matching that property.

Why each choice is right or wrong

A - Correct: Uniqueness is explicitly an integrity condition. (PDF pp. 14, 26.)

B - Incorrect: SELECT is a retrieval command. (PDF pp. 14, 26.)

C - Incorrect: INSERT is a manipulation command. (PDF pp. 14, 26.)

D - Incorrect: DELETE is a manipulation command. (PDF pp. 14, 26.)

Primary note proof

Study-note section FDB16

> Example: CREATE TABLE Enrollment (StudentID INTEGER NOT NULL REFERENCES Student(StudentID), CourseID INTEGER NOT NULL REFERENCES Course(CourseID), Grade INTEGER CHECK (Grade BETWEEN 0 AND 100), PRIMARY KEY (StudentID,CourseID)); Each column declaration gives its name then type, with commas between declarations. CHECK is a constraint clause, not a standalone GRANT-like statement. POSITIVE is not a general SQL constraint keyword; CHECK (Amount > 0) expresses positivity.

Study-note section FDB26

> DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE.

Study-note section FDB26

> Some texts separate SELECT as DQL.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 14 | lines 17-20

> Key Constraints- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique. An attribute that can uniquely identify a tuple in a relation is called

> the key or primary key of the table. The value of the attribute for different tuples in the relation has to be unique. Primary keys must be unique and cannot be null.

Fundamental of Database Systems module | PDF page 26 | lines 10-11

> 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Question sources

Computer Science MCQ practice collection · Question 138

Question 169 | AM0139

What is the purpose of referential integrity in a database?

A. To prevent unauthorized access to data

B. To ensure data consistency and integrity across related tables

C. To improve query performance

D. To encrypt sensitive data

Hint

Match the validity requirement to the appropriate constraint definition.

Answer B: To ensure data consistency and integrity across related tables

Referential integrity maintains consistency between related tuples by requiring non-null child references to resolve to existing parent keys. It controls valid relationships rather than authorization, performance, or encoding.

Why each choice is right or wrong

A - Incorrect: Unauthorized access is a security issue. (PDF pp. 14, 48.)

B - Correct: Matches the module's consistency-between-relations definition. (PDF pp. 14, 48.)

C - Incorrect: Query performance is not the referential validity rule. (PDF pp. 14, 48.)

D - Incorrect: Encryption is a separate security mechanism. (PDF pp. 14, 48.)

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB38

> Both encryption and access control can protect patient data.

Study-note section FDB38

> Authentication verifies identity; role-based authorization limits allowed records/actions; encryption protects data at rest or in transit; auditing records activity.

Related reading (context, not primary note proof)

Study-note section FDB25

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 14 | lines 21-26

> Referential Integrity Constraints - The Referential integrity constraints is specified between two relations or tables and used to maintain the consistency among the tuples in two relations. Referential Integrity constraints in DBMS are based on the concept of Foreign Keys. A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity • A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Variant 1 | Alternative version 1

What is the purpose of referential integrity in a database?

A. To prevent unauthorized access to data

B. To ensure data consistency and integrity

C. To improve query performance

D. To encrypt sensitive data

Answer B: To ensure data consistency and integrity

Referential integrity maintains consistency between related tuples by requiring non-null child references to resolve to existing parent keys. It controls valid relationships rather than authorization, performance, or encoding.

Why each choice is right or wrong

A - Incorrect: Authorization controls access; referential integrity checks reference validity (PDF pp.47-48).

B - Correct: Consistent valid references between related records match the integrity definition (PDF pp.14,48).

C - Incorrect: Retrieval speed is not the foreign-key consistency condition.

D - Incorrect: Encoding sensitive data is a separate security mechanism.

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB38

> Scalability, availability, fast retrieval, redundancy control and security are all reasonable design considerations.

Study-note section FDB38

> Authentication verifies identity; role-based authorization limits allowed records/actions; encryption protects data at rest or in transit; auditing records activity.

Related reading (context, not primary note proof)

Study-note section FDB25

Question sources

Computer Science MCQ practice collection · Question 139

Computer Science MCQ practice collection · Question 154

Question 170 | AM0140

What is denormalization in database design?

A. The process of breaking down a complex table into smaller tables

B. The process of adding redundancy to improve data retrieval performance

C. The process of eliminating duplicate values from a table

D. The process of converting data into a different format for storage

Hint

Compare design techniques by the specific problem each solves.

Answer B: Deliberately adding redundancy to improve selected retrieval workloads.

FDB38 defines denormalization as deliberately repeating or precomputing facts to reduce selected read costs, with extra maintenance work. It is a workload-driven trade-off rather than elimination of all duplication. Breaking tables apart typically describes normalization; changing storage format is a different concern. Thus B captures both the mechanism and its intended retrieval benefit.

Why each choice is right or wrong

A - Incorrect: Splitting tables according to dependencies is normalization, rather than the redundancy introduced by denormalization.

B - Correct: Denormalization deliberately repeats or precomputes information to improve selected retrieval workloads.

C - Incorrect: Eliminating duplication describes a different direction from deliberately introducing useful redundant facts.

D - Incorrect: Changing storage format does not define the dependency and redundancy trade-off of denormalization.

Primary note proof

Study-note section FDB38

> Normalization reduces dependency-driven anomalies. Indexing supplies search paths. Denormalization deliberately repeats or precomputes facts to reduce selected read costs, increasing maintenance work. Partitioning divides rows or columns; replication creates copies to improve locality or availability while complicating updates. These techniques can coexist. An e-commerce or gaming label alone does not select one.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-27

> 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship.

Question sources

Computer Science MCQ practice collection · Question 140

Question 171 | AM0141

What is the purpose of a view in a database?

A. To define the structure of a table

B. To store complex queries for reuse

C. To encrypt sensitive data

D. To enforce data types in columns

Hint

Distinguish a stored query definition from stored data and trace subquery scope.

Answer B: Store a query definition for reuse as a virtual table.

FDB33 says CREATE VIEW stores a query definition and presents a virtual table. B describes reusing that query, although a view need not be complex. Ordinary views do not inherently store result rows. Table definitions and column types belong to schema DDL, and views do not encrypt sensitive content merely by hiding columns or rows.

Why each choice is right or wrong

A - Incorrect: A view defines a virtual query result rather than the base table's storage structure.

B - Correct: A view saves a query definition that can be reused as a virtual table.

C - Incorrect: Restricting exposed data through a view does not encrypt that data.

D - Incorrect: Column data types are enforced by schema definitions and constraints rather than the view's primary purpose.

Primary note proof

Study-note section FDB33

> A subquery is a SELECT nested inside another statement. It may appear in WHERE, HAVING or FROM, not only the first two. IN tests membership; EXISTS tests whether at least one qualifying row exists. Correlated subqueries refer to the outer row. A scalar subquery must produce an appropriate single value. Beware NOT IN when its subquery can return NULL; NOT EXISTS often expresses the intended absence more safely.

Study-note section FDB33

> CREATE VIEW CSStudent AS SELECT StudentID,Name FROM Student WHERE Major = 'CS'; stores a query definition and presents a virtual table.

Study-note section FDB33

> An ordinary view does not necessarily store its result; a materialized view does.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 5-6

> Support of multiple views of the data - A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

Fundamental of Database Systems module | PDF page 56 | lines 7-12

> Views ● A subset of the database presented to some set of users ◦ SQL: CREATE VIEW viewname AS SELECT field1, field2, field3,..., FROM table1, table2 WHERE <where clause>; ◦ Note: —queries‖ in Access function as views

Fundamental of Database Systems module | PDF page 57 | lines 14-21

> ● Views, along with privileges, allow for customised access ● Views provide _derived‘ tables ◦ A view is the result of a SELECT statement which is treated like a table ◦ You can SELECT from (and

sometimes UPDATE etc) views just like tables Creating Views CREATE VIEW <name> AS <select stmt> ● <name> is the name of the new view

Question sources

Computer Science MCQ practice collection · Question 141

Question 172 | AM0142

Which database model represents data as a collection of objects?

A. Hierarchical model

B. Relational model

C. Object-oriented model

D. Network model

Hint

Compare the data models by how they organize records and relationships.

Answer C: Object-oriented model.

FDB02 says object models represent objects with identity, state and behavior. That matches a collection of objects directly. Hierarchical models use a parent-child tree, relational models use tables, and classic network models link records through owner/member pointers. Those representations should not be conflated merely because programming languages can wrap their data in objects.

Why each choice is right or wrong

A - Incorrect: The hierarchical model organizes records in a parent-child tree rather than primarily as objects.

B - Incorrect: The relational model represents data in tables with tuples and attributes.

C - Correct: The object model represents objects with identity, state and behavior.

D - Incorrect: The network model represents linked records through owner/member connections rather than object behavior.

Primary note proof

Study-note section FDB02

A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one 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 7 | lines 8-19

> Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model. 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users. 3. representational (or implementation) data model provides concepts that may be easily understood by end users but that are not too far from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. The widely used relational data models are the network and hierarchical.

Fundamental of Database Systems module | PDF page 13 | lines 18-27

> 3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Computer Science MCQ practice collection · Question 142

Question 173 | AM0143

What is the purpose of data normalization in database design?

A. To improve data storage efficiency

B. To simplify the database structure

C. To eliminate data redundancy

D. To optimize query performance

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer C: To eliminate data redundancy

Normalization organizes/decomposes tables to avoid redundancy and anomalies. Storage may benefit, but the explicit defining purpose among these choices is reducing/eliminating repeated data.

Why each choice is right or wrong

A - Incorrect: Storage efficiency can be a consequence, but the cited definition does not guarantee a storage result. (PDF pp. 16.)

B - Incorrect: Decomposition does not necessarily simplify the whole schema. (PDF pp. 16.)

C - Correct: Eliminating redundancy is explicitly stated. (PDF pp. 16.)

D - Incorrect: Query speed is not the stated normalization goal. (PDF pp. 16.)

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-29

> 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship. Normal Forms are maintenance rules that are used to eliminate or reduce the redundancy in database relations. These normal forms are 1NF (First Normal Form), 2NF (Second Normal Form)

Question sources

Computer Science MCQ practice collection · Question 143

Question 174 | AM0143_2

In the customary single-key teaching case, which is the first normalization stage summarized as dependence on the key, the whole key, and nothing but the key?

A. First normal form (1NF)

B. Second normal form (2NF)

C. Third normal form (3NF)

D. Boyce-Codd normal form (BCNF)

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer C: Third normal form (3NF)

In the stated single-key teaching case, 2NF excludes partial dependencies but can still allow a nonprime fact to depend on the key through a nonkey determinant. The next defining stage is 3NF, summarized by the whole-key-and-nothing-but-the-key mnemonic. BCNF strengthens the determinant rule, and 4NF addresses multivalued dependencies. Stronger forms inherit earlier conditions, so the explicit first-stage scope makes 3NF the unique selection rather than treating inherited properties as rival answers.

Why each choice is right or wrong

A - Incorrect: Incorrect: 1NF concerns atomic values and does not eliminate the specified transitive dependencies.

B - Incorrect: Incorrect: 2NF excludes partial dependencies but can retain a nonprime transitive dependency through a nonkey determinant.

C - Correct: Correct: 3NF is the first stage imposing the specified nonprime transitive-dependency restriction in this teaching case.

D - Incorrect: Incorrect: BCNF strengthens 3NF by requiring every nontrivial determinant to be a superkey; it is not the first stage described.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 10-15

> Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science MCQ practice collection · Question 143

Question 175 | AM0144

Which database model represents data as a collection of records linked through pointers?

A. Hierarchical model

B. Relational model

C. Object-oriented model

D. Network model

Hint

Compare the data models by how they organize records and relationships.

Answer D: Network model.

FDB02 specifically describes the classic network model as records connected by pointer-based owner/member links, allowing multiple parents. That is the representation in the stem. Hierarchical data is restricted to a parent-child tree; relational data uses tables; object-oriented data emphasizes identity, state and behavior. Pointer-linked records therefore identify the network model in this comparison.

Why each choice is right or wrong

A - Incorrect: A hierarchical model restricts record organization to a parent-child tree rather than general network links.

B - Incorrect: A relational model uses tables and logical keys rather than the classic pointer-based network representation.

C - Incorrect: Object modeling is defined by identity, state and behavior, not this classic linked-record formulation.

D - Correct: Classic network databases connect records through pointer-based owner/member links.

Primary note proof

Study-note section FDB02

A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one 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 7 | lines 8-19

Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model. 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users. 3. representational (or implementation) data model provides concepts that may be easily understood by end users but that are not too far from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. The widely used relational data models are the network and hierarchical.

Fundamental of Database Systems module | PDF page 13 | lines 18-27

3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Computer Science MCQ practice collection · Question 144

Question 176 | AM0145

What is the purpose of database indexing?

A. To encrypt sensitive data

B. To enforce data integrity constraints

C. To improve data retrieval performance

D. To ensure data consistency

Hint

Compare index organization with its access benefits and maintenance costs.

Answer C: To improve data retrieval performance

Indexes provide additional access paths and explicitly speed retrieval in response to search conditions; the purpose is performance of reads.

Why each choice is right or wrong

A - Incorrect: Encoding data is not the defined indexing purpose. (PDF pp. 21.)

B - Incorrect: The cited purpose is access speed, not integrity enforcement. (PDF pp. 21.)

C - Correct: Directly matches the retrieval definition. (PDF pp. 21.)

D - Incorrect: Consistency is a constraint/transaction concern rather than the indexing purpose defined. (PDF pp. 21.)

Primary note proof

Study-note section FDB25

> An index stores search keys with pointers or row identifiers. It often speeds selective retrieval but consumes space and must be maintained on writes. A full scan can win for a tiny table or a query returning most rows. Indexes do not themselves establish referential integrity.

Study-note section FDB25

> Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists.

Study-note section FDB38

> Indexing supplies search paths.

Related reading (context, not primary note proof)

Study-note section FDB15

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 5-9

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.

Question sources

Computer Science MCQ practice collection · Question 145

Question 177 | AM0146

Which database model represents data as a collection of tables with rows and columns?

A. Hierarchical model

B. Relational model

C. Object-oriented model

D. Network model

Hint

Compare the data models by how they organize records and relationships.

Answer B: Relational model

The relational model represents data using relations, conventionally shown as tables of tuples and attributes. The other choices have different defining structures: a hierarchical model uses a parent-child tree, a classic network model uses pointer-based owner/member links, and an object model uses objects with identity, state and behavior. Products may combine features, but the question asks for the model defined by rows and columns, which is relational.

Why each choice is right or wrong

A - Incorrect: Incorrect: the hierarchical model organizes records in a parent-child tree.

B - Correct: Correct: the relational model uses relations represented as tables of rows and columns.

C - Incorrect: Incorrect: the object model represents objects with identity, state and behavior.

D - Incorrect: Incorrect: the classic network model uses pointer-based owner/member links, allowing multiple parents.

Primary note proof

Study-note section FDB02

A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one query.

Related reading (context, not primary note proof)

Study-note section FDB03

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 18-27

> 3.6 The Relational Database Model The relational model represents the database as a collection of relations. A table is represented with columns and rows. Each table is an entity in the ER diagram. Each row is known as a tuple. Each table of the column has a name or attribute of the entity. 3.7 Relational Model Concepts in DBMS Attribute: Each column in a Table. Attributes are the properties which define a relation Tables - In the Relational model, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes. Tuple - It is a single row of a table, which contains a single record.

Question sources

Computer Science MCQ practice collection · Question 146

Question 178 | AM0147

What is the purpose of a database schema?

A. To enforce data types in columns

B. To store database files

C. To define the structure and organization of a database

D. To ensure data integrity

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer C: To define the structure and organization of a database

A schema is the database description specified during design. It expresses structure, types, relationships and constraints; enforcing a particular constraint is narrower than the description's overall purpose.

Why each choice is right or wrong

A - Incorrect: Types are included, but enforcement alone is narrower than the schema description. (PDF pp. 7, 7.)

B - Incorrect: Files hold data physically; the schema describes the database. (PDF pp. 7, 7.)

C - Correct: Structure and organization match the schema definition. (PDF pp. 7, 7.)

D - Incorrect: Integrity conditions can be specified in a schema, but integrity alone is not its overall descriptive purpose. (PDF pp. 7, 7.)

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

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

> A data model a collection of concepts that can be used to describe the structure of a database and provides the necessary means to achieve data abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.

Fundamental of Database Systems module | PDF page 7 | lines 20-24

> 2.2 Schemas and Instances The description of a database is called the database schema, which is specified during database design and is not expected to change frequently. The data in the database at a particular moment of time is called a database state or snapshot. It is also called the current set of occurrences or instances in the database.

Question sources

Computer Science MCQ practice collection · Question 147

Question 179 | AM0148

What is the role of a database administrator (DBA) in database management?

A. To design and implement the database schema

B. To write complex SQL queries

C. To manage and maintain the database system

D. To develop database applications

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer C: To manage and maintain the database system

The DBA administers database and DBMS resources, authorizes access and coordinates/monitors use. Management and maintenance summarize those system responsibilities.

Why each choice is right or wrong

A - Incorrect: Schema design may be a duty in practice but is not the defining administrator role stated. (PDF pp. 6.)

B - Incorrect: Complex query writing is not the defining administrative responsibility. (PDF pp. 6.)

C - Correct: Matches resource administration and monitoring. (PDF pp. 6.)

D - Incorrect: Developing applications is a different role from the stated administration duties. (PDF pp. 6.)

Primary note proof

Study-note section FDB01

> The DBA manages access authorization, backup/recovery, availability, monitoring, tuning and routine maintenance, coordinating database use; designers define the schema and application developers build user-facing programs.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 10-16

> 1.4 Actors on the scene The people whose jobs involve the day-to-day use of a large database are called actors on the scene. In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software. Administering these resources is the responsibility of the database administrator (DBA). The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

Question sources

Computer Science MCQ practice collection · Question 148

Computer Science MCQ practice collection · Question 169

Question 180 | AM0149

Which database model represents data as a hierarchical structure with parent-child relationships?

A. Hierarchical model

B. Relational model

C. Object-oriented model

D. Network model

Hint

Compare the data models by how they organize records and relationships.

Answer A: Hierarchical model.

The defining phrase is a hierarchical parent-child structure. FDB02 assigns that tree organization to the hierarchical model. The relational model uses tables; the object model represents identity, state and behavior; the network model permits more general links including multiple parents. A parent-child tree therefore selects A directly.

Why each choice is right or wrong

A - Correct: The hierarchical model organizes records into a parent-child tree.

B - Incorrect: The relational model represents data as tables rather than a prescribed parent-child tree.

C - Incorrect: The object model represents object identity, state and behavior rather than requiring a tree.

D - Incorrect: The network model permits more general connections and multiple parents rather than the restricted tree structure.

Primary note proof

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one 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 7 | lines 8-19

> Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model. 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users. 3. representational (or implementation) data model provides concepts that may be easily understood by end users but that are not too far from the way data is organized in computer storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. The widely used relational data models are the network and hierarchical.

Fundamental of Database Systems module | PDF page 13 | lines 5-8

> 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.

Question sources

Computer Science MCQ practice collection · Question 149

Question 181 | AM0150

What is the first step in the database design process?

A. Defining relationships between tables

B. Determining the database schema

C. Analyzing the information requirements

D. Creating database queries

Hint

Identify the design phase from its inputs and intended deliverable.

Answer C: Analyzing the information requirements

The initial design step is to understand the organization's information needs, required transactions, reports and constraints. Those requirements guide conceptual modeling, which is then mapped into a logical schema. Defining table relationships and determining the schema therefore depend on the earlier analysis. Queries express required operations against a model rather than replacing the initial work of identifying what information and behavior are needed. C is the first step in that sequence.

Why each choice is right or wrong

A - Incorrect: Incorrect: table relationships are designed to satisfy requirements already identified.

B - Incorrect: Incorrect: schema choices follow the analysis of required information and operations.

C - Correct: Correct: analyze information requirements before choosing the model and schema.

D - Incorrect: Incorrect: query construction depends on understanding the required information and intended database structure.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 7-20

> 3.2 Phases of database design The phases of database development life cycle (DDLC) in the Database Management System (DBMS) are explained below. Requirement Analysis The most important step in implementing a database system is to find out what is needed i.e what type of a database is required for the business organization, daily volume of data, how much data needs to be stored in the master files etc. In order to collect all this information, a database analyst spends a lot of time within the business organization talking to people, end users and getting acquainted with the day-to-day process. Conceptual Database Design In this phase the database designers will make a decision on the database model that perfectly suits the organization‘s requirement. The database designers will study the documents prepared by the analysis in the requirement analysis stage and then start development of a system model that fulfils the needs.

Question sources

Computer Science MCQ practice collection · Question 150

Question 182 | AM0151

Read pairs left→right. Student↔Course and Product↔Supplier permit many occurrences in both directions. Each employee belongs to one department with many employees. Each order belongs to one customer, who may place many orders. Which pair is one-to-many?

A. Student and Course

B. Employee and Department

C. Customer and Order

D. Product and Supplier

Hint

Read the relationship in both directions and check participation constraints.

Answer C: C. Customer and Order

Customer→Order is one-to-many because one customer can place several orders and each order belongs to one customer. Employee→Department runs in the opposite direction: many employees share one department, making it many-to-one. The student/course and product/supplier pairs explicitly permit many partners in both directions and are many-to-many. Cardinality follows these rules and the direction of the pair.

Why each choice is right or wrong

A - Incorrect: Student→Course is M:N under the stated enrollment rule.

B - Incorrect: Employee→Department is N:1.

C - Correct: Customer→Order is 1:N under the ownership rule.

D - Incorrect: Product→Supplier is M:N under the stated supply rule.

Primary note proof

Study-note section FDB09

> Cardinality specifies maximum associated occurrences: 1:1, 1:N or M:N. Minimum cardinality specifies optional/partial participation (zero permitted) or mandatory/total participation (at least one). A maximum of one does not mean participation is compulsory. Relationship degree counts participating entity roles: unary recursive, binary or ternary.

Study-note section FDB09

> If each employee belongs to one department and each department has many employees, Employee→Department is N:1; Department→Employee is 1:N.

Study-note section FDB09

> Read business rules in both directions.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 24-31

> Cardinality The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types: One to one (1-1) - When each entity in each entity set can take part only once in the relationship, the cardinality is one to one. One to many(1-M)- When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.

Fundamental of Database Systems module | PDF page 12 | lines 2-3

> Many to many (M-M) - When entities in all entity sets can take part more than once in the relationship cardinality is many to many.

Question sources

Computer Science MCQ practice collection · Question 151

Question 183 | AM0153

Which of the following is a benefit of using database indexes?

A. Increased data redundancy

B. Decreased data integrity

C. Improved data retrieval performance

D. Simplified data modeling

Hint

Compare index organization with its access benefits and maintenance costs.

Answer C: Improved data retrieval performance

The auxiliary index provides alternative access paths to speed retrieval under search conditions. This is the directly supported benefit.

Why each choice is right or wrong

A - Incorrect: Redundancy is not the benefit identified by the index definition. (PDF pp. 21.)

B - Incorrect: Reduced integrity is not a stated benefit. (PDF pp. 21.)

C - Correct: Retrieval improvement is explicitly stated. (PDF pp. 21.)

D - Incorrect: Simplification of the logical model is not the index's stated function. (PDF pp. 21.)

Primary note proof

Study-note section FDB25

> An index stores search keys with pointers or row identifiers. It often speeds selective retrieval but consumes space and must be maintained on writes. A full scan can win for a tiny table or a query returning most rows. Indexes do not themselves establish referential integrity.

Study-note section FDB25

> Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists.

Study-note section FDB25

> A dense index represents each search value; sparse indexes rely on ordered data between entries.

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 5-9

> 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.

Question sources

Computer Science MCQ practice collection · Question 153

Question 184 | AM0155

Employee has required, unique SSN and Employee ID values. Names and salaries may repeat. Which listed individual attributes are candidate keys? Select all that apply.

A. Social Security Number (SSN)

B. Employee Name

C. Employee ID

D. Employee Salary

Hint

Test uniqueness and minimality separately when classifying keys.

Answer A, C: A. Social Security Number (SSN); C. Employee ID

A candidate key is a minimal attribute set guaranteed to identify a tuple. Each required unique single-attribute identifier is already minimal, so SSN and Employee ID both qualify. Names and salaries may repeat and therefore cannot guarantee identification. The question asks for candidate keys rather than the one candidate later selected as the primary key, allowing both identifiers.

Why each choice is right or wrong

A - Correct: Required unique SSN identifies each employee and is minimal.

B - Incorrect: Repeated names are permitted, so Name is not a candidate key.

C - Correct: Required unique Employee ID independently identifies each employee.

D - Incorrect: Repeated salaries are permitted, so Salary is not a candidate key.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> If ID is unique, both {ID} and {ID,City} are superkeys, but only {ID} is minimal.

Study-note section FDB07

> Prefer stable, compact identifiers when practical, but Email, SSN or EmployeeID is a candidate only if business rules guarantee uniqueness.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Computer Science MCQ practice collection · Question 155

Question 185 | AM0156

What is the purpose of a foreign key in database design?

A. To enforce data types in columns

B. To define primary key constraints

C. To establish relationships between tables

D. To optimize query performance

Hint

Match the validity requirement to the appropriate constraint definition.

Answer C: To establish relationships between tables

A foreign key links a child occurrence to the matching parent candidate/primary key. Its defining purpose is the intertable relationship.

Why each choice is right or wrong

A - Incorrect: Domain constraints specify legal types. (PDF pp. 48.)

B - Incorrect: The parent key is an identifying constraint, while the child FK expresses the reference. (PDF pp. 48.)

C - Correct: Matches the stated linking purpose. (PDF pp. 48.)

D - Incorrect: Performance is not the defining reference function. (PDF pp. 48.)

Primary note proof

Study-note section FDB15

Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB07

A primary key is one selected candidate key; remaining candidates are alternate keys.

Study-note section FDB07

A composite key contains multiple attributes and need not be the chosen primary key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity ● A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key ● Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Computer Science MCQ practice collection · Question 156

Question 186 | AM0157

Which of the following database design principles promotes data independence?

A. Entity-Relationship (ER) modeling

B. Normalization

C. Database indexing

D. Data abstraction

Hint

Trace which schema levels a proposed change affects.

Answer D: Data abstraction

Abstraction hides irrelevant details and helps achieve data independence. Three-schema separation then permits changes at one level without forcing higher-level schema changes.

Why each choice is right or wrong

A - Incorrect: ER modeling describes conceptual structure but is not the explicit independence principle here. (PDF pp. 6, 8.)

B - Incorrect: Normalization addresses redundancy/anomalies. (PDF pp. 6, 8.)

C - Incorrect: Indexes address retrieval paths and performance. (PDF pp. 6, 8.)

D - Correct: Abstraction is explicitly connected to independence. (PDF pp. 6, 8.)

Primary note proof

Study-note section FDB04

> Physical data independence permits changing internal storage without changing the conceptual schema, for example adding an index. Logical data independence permits conceptual changes while preserving affected external views and application interfaces, for example splitting a table while maintaining a compatible view. It is harder because applications depend on logical meaning. Data abstraction hides irrelevant details; program-data insulation separates data definitions from application code.

Study-note section FDB05

> Conceptual design turns them into an ER/EER model.

Study-note section FDB05

> Logical design maps the conceptual model to relations, keys and constraints and checks normalization.

Related reading (context, not primary note proof)

Study-note section FDB19

Study-note section FDB25

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 6 | lines 2-4

> Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Fundamental of Database Systems module | PDF page 8 | lines 8-15

> 2.4 Data Independence The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. The two types of data independence: 1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. 2. Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas do not need be changed as well.

Question sources

Computer Science MCQ practice collection · Question 157

Question 187 | AM0158

Which type of database relationship is represented by a join table?

A. One-to-one relationship

B. One-to-many relationship

C. Many-to-many relationship

D. Hierarchical relationship

Hint

Identify relationship degree and cardinality before choosing a relational mapping.

Answer C: Many-to-many relationship.

A join table stores pairs of references to the two participating entity tables, optionally with relationship attributes. FDB12 uses Enrollment(StudentID,CourseID,Grade) for the many-to-many Student-Course relationship. One-to-one and one-to-many relationships normally use a foreign key in an entity table instead. A hierarchy is a structural pattern, not the relationship cardinality represented by this usual association mapping.

Why each choice is right or wrong

A - Incorrect: A 1:1 relationship normally uses a UNIQUE foreign key rather than requiring a join table.

B - Incorrect: A 1:N relationship normally stores the one-side key in the many-side entity table.

C - Correct: A join or association table stores the pairs making up a many-to-many relationship.

D - Incorrect: Hierarchical organization is not the standard cardinality category represented by this association-table mapping.

Primary note proof

Study-note section FDB12

> A binary M:N relationship needs an association table in addition to the two entity tables. Enrollment(StudentID,CourseID,Grade) references Student and Course; (StudentID,CourseID) is a key if one enrollment per pair is allowed. Repeated attempts require Term or AttemptNo in the key. This ordinarily produces three tables.

Study-note section FDB11

> For 1:N, place the one-side key as a foreign key in the many-side table; store relationship attributes there.

Study-note section FDB11

> For 1:1, one side may hold the other key with a UNIQUE constraint.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 12 | lines 2-3

> Many to many (M-M) - When entities in all entity sets can take part more than once in the relationship cardinality is many to many.

Fundamental of Database Systems module | PDF page 12 | lines 8-24

> 3.4 Mapping ER-models to relational tables 1. For each strong entity type E, create a relation R that includes all the attributes of E. 2. Choose one of the key attributes of E as the primary key for R. For each weak entity type W with owner entity type E (which is a strong entity), create a relation R & include all attributes of _W‘ as attributes of R. Also, include the primary key attribute of the relation that correspond to the owner entity type as foreign key attributes of R . 3. For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. Foreign Key approach: Choose one of the relations-say S-and

include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. 4. For each binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type. Include the primary key of the relation T as foreign key in S that represents the other entity type participating in R. Include any attributes of the 1:N relation type as attributes of S. 5. For each n-ary relationship type R, where n>2, create a new relationship S to represent R. Include the primary keys of the relations that represent the participating entity types as foreign key attributes in S. Also include any attributes of the n-ary relationship type as attributes of S.

Question sources

Computer Science MCQ practice collection · Question 158

Question 188 | AM0159

Employee ID uniquely identifies an employee and their name, salary and department. Names may repeat; the same salary may occur in different departments, and a department may contain different salaries. Which listed functional dependency is guaranteed?

A. Employee ID -> Employee Name

B. Employee Name -> Employee ID

C. Employee Salary -> Employee Department

D. Employee Department -> Employee Salary

Hint

Compute closures using the declared dependencies and test minimality.

Answer A: A. Employee ID -> Employee Name

A functional dependency is guaranteed by business rules for every legal state. Unique Employee ID determines all employee attributes, including Name, so A follows. Repeated names can correspond to different IDs, equal salaries can occur in different departments, and one department may contain different salaries. Those permissions prevent the other proposed directions from being guaranteed dependencies.

Why each choice is right or wrong

A - Correct: The unique employee identifier determines the employee's name.

B - Incorrect: Repeated names can belong to different IDs.

C - Incorrect: The same salary may occur in different departments.

D - Incorrect: One department may contain employees with different salaries.

Primary note proof

Study-note section FDB17

X→Y means every legal pair of tuples agreeing on X also agrees on Y. It is a business constraint, not a correlation observed in one sample. Full dependency requires all determinant attributes;

partial dependency holds through a proper subset. A prime attribute belongs to some candidate key; a nonprime attribute belongs to none.

Study-note section FDB17

Remove attributes and recompute to test minimality.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 15 | lines 5-9

A functional dependency is a constraint between two attributes or sets of attributes of entities in a database. A functional dependency is denoted by X → Y, between two sets of attributes X and Y that are subsets of relation R. The constraint is that, the values of Y component of a tuple in r depend on, or are determined by the values of X component. Alternatively, the values of X component of a tuple uniquely (or functionally) determine the values of Y component.

Fundamental of Database Systems module | PDF page 11 | lines 7-11

Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Question sources

Computer Science MCQ practice collection · Question 159

Question 189 | AM0160

What is the purpose of a data dictionary in database design?

A. To store user credentials

B. To manage database backups

C. To define the structure and attributes of data elements

D. To enforce referential integrity constraints

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer C: To describe the structure and attributes of data elements.

The dictionary contains metadata explaining table definitions, attribute types and constraints. FDB03 therefore supports C as its purpose. It may record privileges, but storing user credentials is not its defining function. Backup management is an administrative operation, and enforcing a reference requires DBMS constraint machinery rather than merely the repository that describes the constraint.

Why each choice is right or wrong

A - Incorrect: Credential storage is not the defining purpose of the database structure metadata repository.

B - Incorrect: Backup management is an administrative activity rather than the dictionary's structural-description purpose.

C - Correct: The dictionary records definitions, attributes, data types and constraints describing database elements.

D - Incorrect: It describes constraints; the DBMS enforcement machinery actually checks referential integrity during operations.

Primary note proof

Study-note section FDB03

> A schema describes names, structures and constraints; an instance, state or snapshot is the data at a particular time. Inserting a student changes the state, while adding a column changes the schema. Metadata is data describing data. The catalog or data dictionary records table definitions, types, constraints and often access privileges and optimizer statistics. A schema diagram draws tables and key references; labels such as database diagram can overlap.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 26-29

> Self-describing nature - A fundamental characteristic of the database approach is that the database system contains not only the database itself (data) but also a complete definition or description of the database structure and constraints. This information is called meta-data, and it describes the structure of the database.

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

> A data model a collection of concepts that can be used to describe the structure of a database and provides the necessary means to achieve data abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.

Question sources

Computer Science MCQ practice collection · Question 160

Computer Science MCQ practice collection · Question 184

Question 190 | AM0161

Compared with scanning all pages of a large heap file, which advantage can an RDBMS with a suitable index provide for a query selecting a small fraction of rows? Assume index lookup costs less than the scan.

A. Easy integration with external systems

B. Lower storage requirements

C. No need for data normalization

D. Faster data retrieval

Hint

Compare design techniques by the specific problem each solves.

Answer D: D. Faster data retrieval

The suitable index provides a search path for the selective query, reducing the work compared with scanning the large heap under the stated cost assumption. This supports faster retrieval for this workload. It does not remove normalization concerns, guarantee smaller storage or establish external-system compatibility. Index space and maintenance costs remain separate trade-offs from the query's retrieval advantage.

Why each choice is right or wrong

A - Incorrect: External integration is outside the stated lookup comparison.

B - Incorrect: Indexes consume additional storage rather than guaranteeing a reduction.

C - Incorrect: An index does not remove the need for dependency-aware design.

D - Correct: The stated cheaper indexed lookup supports faster retrieval for this query.

Primary note proof

Study-note section FDB38

> Normalization reduces dependency-driven anomalies. Indexing supplies search paths. Denormalization deliberately repeats or precomputes facts to reduce selected read costs, increasing maintenance work. Partitioning divides rows or columns; replication creates copies to improve locality or availability while complicating updates. These techniques can coexist. An e-commerce or gaming label alone does not select one.

Study-note section FDB25

> It often speeds selective retrieval but consumes space and must be maintained on writes.

Study-note section FDB25

> A full scan can win for a tiny table or a query returning most rows.

Related reading (context, not primary note proof)

Study-note section FDB01

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 14-21

> Database Management System (DBMS) is then a tool for creating and managing databases. DBMS is a software that facilities the processes of defining, constructing, manipulating, and sharing databases. Example: MySQL server, MS ACESS, Microsoft SQL server. 1.2 Database System versus File System The traditional file processing system is file-directory structure supported by a conventional operating system. A file system organization of data lucks a number of major features of a database system, such as Data redundancy and inconsistency, Difficulty in accessing data, Data isolation, Integrity problems, Atomicity problems, Concurrent access anomalies.

Fundamental of Database Systems module | PDF page 21 | lines 5-9

> 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.

Question sources

Computer Science MCQ practice collection · Question 161

Question 191 | AM0163

OrderLine has one row per line number within an order. OrderID and LineNo together uniquely identify a row; neither alone does. Which listed attribute set is a composite key?

A. Employee ID

B. Product SKU

C. Customer Name

D. (OrderID, LineNo)

Hint

Test uniqueness and minimality separately when classifying keys.

Answer D: D. (OrderID, LineNo)

A composite key contains more than one attribute. OrderID identifies an order and LineNo identifies a line within it; the stated rules make their pair unique while neither component alone is sufficient. That establishes a minimal composite key. The other options each name only one attribute and therefore do not illustrate a composite key, regardless of any separate uniqueness policy.

Why each choice is right or wrong

A - Incorrect: Employee ID is a single attribute.

B - Incorrect: Product SKU is a single attribute.

C - Incorrect: Customer Name is a single listed attribute.

D - Correct: The pair is unique and requires both stated attributes.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> If ID is unique, both {ID} and {ID,City} are superkeys, but only {ID} is minimal.

Study-note section FDB07

> A composite-key example must include multiple columns; a single-column key is not composite.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-11

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Fundamental of Database Systems module | PDF page 15 | lines 11-17

> Partial Dependency - If an attribute which is not a member of the key is dependent on some part of the key (if we have composite/candidate key) then that attribute is partially functionally dependent on the key. Let {A, B} is the key set and C is no key attribute, then if {A, B} → C and [B → C or A → C] holds. We say C is partially functionally dependent on {A, B}. Partial dependency occurs when a non-prime attribute is functionally dependent on part of a candidate/composite key.

Question sources

Computer Science MCQ practice collection · Question 163

Question 192 | AM0164

What is the purpose of data validation in database design?

A. To enforce referential integrity constraints

B. To improve query performance

C. To eliminate data redundancy

D. To ensure data integrity and quality

Hint

Match the validity requirement to the appropriate constraint definition.

Answer D: To ensure data integrity and quality

Validation checks that input values meet legal domains/criteria, protecting integrity and consistency. D is the broad purpose; referential validation is one narrower kind rather than the whole definition.

Why each choice is right or wrong

A - Incorrect: Referential checks are one specific constraint, not the overall purpose of validation. (PDF pp. 47, 48, 49.)

B - Incorrect: Performance is not the legal-input goal. (PDF pp. 47, 48, 49.)

C - Incorrect: Redundancy reduction belongs to normalization. (PDF pp. 47, 48, 49.)

D - Correct: Integrity and input quality match domain validation and CHECK criteria. (PDF pp. 47, 48, 49.)

Primary note proof

Study-note section FDB15

> Domain integrity restricts type and allowed values. Key integrity enforces uniqueness. Entity integrity requires every primary-key component to be non-NULL. Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed. A parent need not have a child.

Study-note section FDB19

> Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Study-note section FDB25

> Indexes do not themselves establish referential integrity.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 47 | lines 4-19

> ● Security vs Integrity ◦ Database security makes sure that the user is authorised to access information ◦ Database integrity makes sure that (authorised) users use that information correctly ● Integrity constraints ◦ Domain constraints apply to data types ◦ Attribute constraints apply to columns ◦ Relation constraints apply to rows in a single table ◦ Database constraints apply between tables ● Intrarecord integrity (enforcing constraints on contents of fields, etc.) ● Referential Integrity (enforcing the validity of references between records in the database) ● Concurrency control (ensuring the validity of database updates in a shared multiuser

environment) ● The constraints we wish to impose in order to protect the database from becoming inconsistent.

Fundamental of Database Systems module | PDF page 48 | lines 4-8

ii. Attribute domain constraints ● Every attribute has a domain, that is a set of values that are legal for it to use ● For example: ◦ The domain of sex in the employee relation is —M‖ or —F‖ ● Domain ranges can be used to validate input to the database

Fundamental of Database Systems module | PDF page 49 | lines 19-22

● CREATE TABLE table-name (attr2 attr-type NOT NULL, forbids NULL values attrN attr-type CHECK (attrN = UPPER(attrN) verifies that the data meets certain criteria attrO attr-type DEFAULT default_value); Supplies default values

Question sources

Computer Science MCQ practice collection · Question 164

Question 193 | AM0165

Which normal forms have defining restrictions on functional dependencies? Select all that apply. Treat 1NF as the atomic-value/no-repeating-groups requirement.

A. First Normal Form (1NF)

B. Second Normal Form (2NF)

C. Third Normal Form (3NF)

D. Boyce-Codd Normal Form (BCNF)

Hint

Test determinants and prime attributes against the stated normal-form rules.

Answer B, C, D: B. Second Normal Form (2NF); C. Third Normal Form (3NF); D. Boyce-Codd Normal Form (BCNF)

Second normal form excludes nonprime partial functional dependencies on candidate keys. Third normal form checks each nontrivial dependency using the determinant's superkey status or the dependent's prime status. BCNF strengthens that test by requiring a superkey determinant. These three therefore use functional-dependency restrictions. First normal form instead establishes the atomic-value foundation and absence of repeating groups.

Why each choice is right or wrong

A - Incorrect: 1NF addresses atomic values and repeating groups rather than the stated dependency-based tests.

B - Correct: 2NF is defined through absence of nonprime partial functional dependencies on candidate keys.

C - Correct: 3NF tests nontrivial FDs using determinant superkeys and prime dependent attributes.

D - Correct: BCNF requires every nontrivial FD determinant to be a superkey.

Primary note proof

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Study-note section FDB20

> A relation whose every candidate key is single-attribute has no nonprime partial-key dependency.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 2-15

> First normal form (1NF) A relation (table) R is in 1NF if and only if (iff) all underlying domains of attributes contain only atomic (simple or indivisible) values. i.e. The value of any attribute in a tuple (row) must be a single value from the domain of their attribute. It is defined to disallow multivalued attributes, composite attributes, and their combinations. In other words, 1NF disallows relations within relations or relations as attribute values within tuples. It must be remove repeating groups (values) in a single field. The primary key with repeating group attributes are moved into a new table. When a relation contains no repeating groups (values), it is in first normal form (1NF). Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

> Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of ‗R' are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science MCQ practice collection · Question 165

Question 194 | AM0166

Which are direct representational purposes of conceptual/logical data modeling, rather than DBMS enforcement or physical query tuning? Select all that apply.

A. To enforce data constraints

B. To optimize query performance

C. To define the structure and organization of a database

D. To represent real-world entities and their relationships

Hint

Identify the design phase from its inputs and intended deliverable.

Answer C, D: C. To define the structure and organization of a database; D. To represent real-world entities and their relationships

Conceptual modeling represents domain entities and their relationships, while logical modeling defines database structures, keys and constraints. Both are direct representational purposes, so C and D qualify. Modeling can describe constraints, but the DBMS and application implementation enforce them during operations. Query performance can inform physical design, yet it is not the direct representational task specified in this question.

Why each choice is right or wrong

A - Incorrect: A model describes constraints, while implementation and DBMS machinery enforce them on actual operations.

B - Incorrect: Query performance can inform physical design, but it is not the general representational purpose asked here.

C - Correct: Modeling specifies the database's structure and organization.

D - Correct: Conceptual modeling represents real-world entities and their relationships.

Primary note proof

Study-note section FDB05

> First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations.

Study-note section FDB02

> Conceptual ER models describe entities and relationships.

Module passages - related reading

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

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

> A data model a collection of concepts that can be used to describe the structure of a database and provides the necessary means to achieve data abstraction. By structure of a database we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of basic operations for specifying retrievals and updates on the database.

Fundamental of Database Systems module | PDF page 9 | lines 3-6

> 3.1 Database Modeling High-level conceptual data models provide concepts for presenting data in ways that are close to the way people perceive data. A typical example is the entity relationship model, which uses main concepts like entities, attributes and relationships.

Fundamental of Database Systems module | PDF page 9 | lines 27-31

> 3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Question sources

Computer Science MCQ practice collection · Question 166

Question 195 | AM0167

Which option states a measurable nonfunctional quality requirement for a database system?

A. The database service must achieve at least 99.9% uptime.

B. User authentication

C. Entity-Relationship (ER) diagram

D. Data normalization

Hint

Compare design techniques by the specific problem each solves.

Answer A: A. The database service must achieve at least 99.9% uptime.

A nonfunctional requirement states a quality target such as availability or response time. The uptime threshold specifies a measurable availability requirement and fits that category. User authentication names system behavior, while an ER diagram is a modeling artifact and normalization is a design technique. These may support requirements, but they do not themselves state the quantified quality requested here.

Why each choice is right or wrong

A - Correct: The 99.9% uptime threshold is a measurable availability target.

B - Incorrect: Authentication names behavior rather than a quantified quality target.

C - Incorrect: An ER diagram is a design artifact.

D - Incorrect: Normalization is a design technique.

Primary note proof

Study-note section FDB38

Privacy requires a threat model. Authentication verifies identity; role-based authorization limits allowed records/actions; encryption protects data at rest or in transit; auditing records activity. Both encryption and access control can protect patient data. Scalability, availability, fast retrieval, redundancy control and security are all reasonable design considerations. Functional requirements describe behavior; nonfunctional requirements quantify qualities such as 99.9% uptime or response time. Authentication and encryption mechanisms alone do not uniquely establish that distinction.

Study-note section FDB19

Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Study-note section FDB05

Conceptual design turns them into an ER/EER model.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 19 | lines 4-10

5.1 Introduction To translate the logical description of data into technical specifications for storing and retrieving data, a physical database design is needed. The goal is to create a design for storing and retrieving data that will provide adequate performance and insure database integrity, security, and availability. Some decisions need to be made on the physical database design. They are Attribute data types, data/file storages, File organizations and Indexes.

Fundamental of Database Systems module | PDF page 52 | lines 22-27

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 • Encryption to encode sensitive data

Fundamental of Database Systems module | PDF page 53 | lines 2-13

• Authentication schemes to positively identify a person attempting to gain access to the database • 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 • OS/Network security • Security policies and protocols • Encryption and passwords • DBMS security

Question sources

Computer Science MCQ practice collection · Question 167

Question 196 | AM0168

What is the purpose of data manipulation in database management?

A. To define the database schema

B. To create database backups

C. To insert, update, and delete data

D. To manage concurrent access to data

Hint

Classify each SQL command by the kind of change it performs.

Answer C: To insert, update, and delete data

The DML subsection includes INSERT, UPDATE and DELETE. SQL examples show insertion/deletion and an UPDATE privilege permits changing values, grounding the data-manipulation purpose. DML also includes retrieval, so C describes its modification part rather than all of DML.

Why each choice is right or wrong

A - Incorrect: Schema definition is DDL. (PDF pp. 26, 55, 77.)

B - Incorrect: Backup is not the DML purpose identified. (PDF pp. 26, 55, 77.)

C - Correct: Insertion, updating and deletion are explicitly DML operations. (PDF pp. 26, 55, 77.)

D - Incorrect: Concurrency control is a separate transaction-management responsibility. (PDF pp. 26, 55, 77.)

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 26 | lines 4-11

> SQL represents Structured Query Language. SQL is a comprehensive database language: It has statements for data definitions, and updates. Hence, it is both a DDL (Data Definition language) and a DML (Data Manipulation Language). 7.1.1 DDL (data definition language) Basic DDL commands we usually use are: CREATE, DROP, ALTER, TRUNCATE and RENAME 7.1.2 DML (data manipulation language) Basic DDL commands are: SELECT, PROJECT, JOIN, INSERT, DELETE, and UPDATE

Fundamental of Database Systems module | PDF page 55 | lines 2-5

> UPDATE(Salary) ON Employee TO Finance The user _Finance‘ can view the entire Employee table, and can change Salary values, but cannot change other values or pass on their privilege

Fundamental of Database Systems module | PDF page 77 | lines 14-24

> Example 1: Commit Transaction BEGIN TRANSACTION; DELETE FROM HumanResources.JobCandidate WHERE JobCandidateID = 13; COMMIT; Example 2: Rollback Transaction CREATE TABLE ValueTable (id int); BEGIN TRANSACTION; INSERT INTO ValueTable VALUES(1); INSERT INTO ValueTable VALUES(2); ROLLBACK;

Question sources

Computer Science MCQ practice collection · Question 168

Question 197 | AM0170

During conceptual database design, which listed goal directly concerns representing the business domain before choosing storage layouts and access paths?

A. Efficient storage of data

B. Secure data transmission

C. Optimal query performance

D. Effective representation of real-world entities

Hint

Identify the design phase from its inputs and intended deliverable.

Answer D: D. Effective representation of real-world entities

Conceptual design turns business facts into an ER/EER representation of entities and relationships. Effective representation of the domain is therefore the direct goal in this phase. Storage efficiency and query access paths are later physical design concerns. Secure transmission can be an important requirement, but it is not the conceptual representational task singled out by the revised question.

Why each choice is right or wrong

A - Incorrect: Storage layout efficiency belongs to physical decisions.

B - Incorrect: Transmission security is a requirement or mechanism, not the specified conceptual representation goal.

C - Incorrect: Query performance is addressed through later physical design and tuning.

D - Correct: Representing real-world entities directly matches conceptual design.

Primary note proof

Study-note section FDB05

First collect business facts, required transactions, reports, volumes, privacy rules and quality targets. Conceptual design turns them into an ER/EER model. Logical design maps the conceptual model to relations, keys and constraints and checks normalization. Physical design chooses file organization, indexes and storage. Implementation creates and loads the actual objects; testing checks requirements.

Study-note section FDB38

Indexing supplies search paths.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 16-26

Conceptual Database Design In this phase the database designers will make a decision on the database model that perfectly suits the organization‘s requirement. The database designers will study the documents prepared by the analysis in the requirement analysis stage and then start development of a system model that fulfils the needs. Logical database design (Data model mapping) The conceptual design is translated into internal model which includes mapping of all objects i.e design of tables, indexes, views, transaction, access privileges etc. Physical Design Physical design or Database implementation needs the formation of special storage related constructs. These constructs consist of storage groups, table spaces, data files, tables etc.

Fundamental of Database Systems module | PDF page 19 | lines 4-10

5.1 Introduction To translate the logical description of data into technical specifications for storing and retrieving data, a physical database design is needed. The goal is to create a design for storing and retrieving data that will provide adequate performance and insure database integrity, security, and availability. Some decisions need to be made on the physical database design. They are Attribute data types, data/file storages, File organizations and Indexes.

Question sources

Computer Science MCQ practice collection · Question 170

Question 198 | AM0171

Which are legitimate considerations when designing a database for a large e-commerce site? Select all that apply. Minimal redundancy means controlling avoidable duplication while permitting deliberate replicas.

A. High availability and scalability

B. Minimal data redundancy

C. Fast data retrieval

D. Secure user authentication

Hint

Compare design techniques by the specific problem each solves.

Answer A, B, C, D: A. High availability and scalability; B. Minimal data redundancy; C. Fast data retrieval; D. Secure user authentication

Large e-commerce databases need several complementary qualities. Availability and scalability support continued service and growth; retrieval speed supports interactive use; security protects accounts and records. Controlling avoidable redundancy reduces consistency and maintenance problems even where intentional replicas are useful. All listed considerations therefore belong in the design discussion, with their relative priority determined by the site's actual requirements.

Why each choice is right or wrong

A - Correct: High availability and scalability support growth and continued service for a large site.

B - Correct: Controlling unnecessary redundancy improves consistency and maintenance even when deliberate copies are retained.

C - Correct: Fast retrieval is a legitimate quality target for browsing, orders and customer-facing operations.

D - Correct: Secure authentication is a legitimate security consideration for customer and administrative access.

Primary note proof

Study-note section FDB38

> Privacy requires a threat model. Authentication verifies identity; role-based authorization limits allowed records/actions; encryption protects data at rest or in transit; auditing records activity. Both encryption and access control can protect patient data. Scalability, availability, fast retrieval, redundancy control and security are all reasonable design considerations. Functional requirements describe behavior; nonfunctional requirements quantify qualities such as 99.9% uptime or response time. Authentication and encryption mechanisms alone do not uniquely establish that distinction.

Study-note section FDB38

> In a CMS, tags/categories describe meaning, while foreign keys preserve valid references between content and tags.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-27

> 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship.

Fundamental of Database Systems module | PDF page 19 | lines 4-10

> 5.1 Introduction To translate the logical description of data into technical specifications for storing and retrieving data, a physical database design is needed. The goal is to create a design for storing and retrieving data that will provide adequate performance and insure database integrity, security, and availability. Some decisions need to be made on the physical database design. They are Attribute data types, data/file storages, File organizations and Indexes.

Fundamental of Database Systems module | PDF page 21 | lines 5-9

> 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.

Fundamental of Database Systems module | PDF page 53 | lines 14-21

> 2.2.1 DBMS Security Support ● DBMS can provide some security ◦ Each user has an account, username and password ◦ These are used to identify a user and control their access to information ● DBMS verifies password and checks a user‘s permissions when they try to ◦ Retrieve data ◦ Modify data ◦ Modify the database structure

Fundamental of Database Systems module | PDF page 116 | lines 19-28

> 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 2-4

> vi. Easier expansion (scalability): • Allows new nodes (computers) to be added anytime without changing the entire configuration.

Question sources

Computer Science MCQ practice collection · Question 171

Question 199 | AM0172

Which listed mechanisms directly protect patient confidentiality, covering both protected stored/transmitted content and restrictions on authorized user actions? Select all that apply.

A. Database normalization

B. Data encryption

C. Role-based access control

D. Database indexing

Hint

Compare design techniques by the specific problem each solves.

Answer B, C: B. Data encryption; C. Role-based access control

Encryption protects patient content at rest or in transit against the disclosure threats it covers. Role-based access control restricts which users can read or change patient records. These are complementary privacy protections and both qualify. Normalization reduces dependency anomalies, while indexing supplies retrieval paths; neither mechanism by itself decides who may see patient information or protects encoded content.

Why each choice is right or wrong

A - Incorrect: Normalization reduces dependency anomalies but does not itself restrict unauthorized disclosure.

B - Correct: Encryption protects patient content at rest or in transit against appropriate disclosure threats.

C - Correct: Role-based access control limits which users may access patient records or perform actions.

D - Incorrect: Indexes provide search paths and do not by themselves enforce patient confidentiality.

Primary note proof

Study-note section FDB38

> Privacy requires a threat model. Authentication verifies identity; role-based authorization limits allowed records/actions; encryption protects data at rest or in transit; auditing records activity. Both encryption and access control can protect patient data. Scalability, availability, fast retrieval, redundancy control and security are all reasonable design considerations. Functional requirements describe behavior; nonfunctional requirements quantify qualities such as 99.9% uptime or response time. Authentication and encryption mechanisms alone do not uniquely establish that distinction.

Study-note section FDB25

> Secondary indexes provide other access paths and must represent all searchable values with record pointers or pointer lists.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency 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 52 | lines 22-27

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 ● Encryption to encode sensitive data

Fundamental of Database Systems module | PDF page 53 | lines 2-18

● Authentication schemes to positively identify a person attempting to gain access to the database ● 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 ● OS/Network security ● Security policies and protocols ● Encryption and passwords ● DBMS security 2.2.1 DBMS Security Support ● DBMS can provide some security ◦ Each user has an account, username and password ◦ These are used to identify a user and control their access to information ● DBMS verifies password and checks a user‘s permissions when they try to

Question sources

Computer Science MCQ practice collection · Question 172

Question 200 | AM0173

Which techniques can improve query performance for some suitable workloads, without implying that they always help? Select all that apply.

A. Indexing

B. Data replication

C. Denormalization

D. Partitioning

Hint

Compare design techniques by the specific problem each solves.

Answer A, B, C, D: A. Indexing; B. Data replication; C. Denormalization; D. Partitioning

Indexes add search paths for selective queries. Replicas can improve read locality, denormalization can reduce selected joins or computations, and partitioning can reduce relevant data access when aligned with queries. Each technique can therefore help some workloads. Their benefits are conditional and can involve extra storage or update costs, so selecting all four does not mean every database should use all of them.

Why each choice is right or wrong

A - Correct: Indexes can reduce reads for selective predicates when maintenance and lookup costs are justified.

B - Correct: Replication can place read copies near users or spread suitable read workloads.

C - Correct: Denormalization can avoid selected joins or repeated computation by storing redundant/precomputed facts.

D - Correct: Partitioning can limit relevant data or improve locality when partitioning matches query predicates.

Primary note proof

Study-note section FDB38

Normalization reduces dependency-driven anomalies. Indexing supplies search paths. Denormalization deliberately repeats or precomputes facts to reduce selected read costs, increasing maintenance work. Partitioning divides rows or columns; replication creates copies to improve locality or availability while complicating updates. These techniques can coexist. An e-commerce or gaming label alone does not select one.

Study-note section FDB25

B/B+ trees support ordered and range search; hashing targets equality; bitmap indexes suit some analytic workloads.

Study-note section FDB24

Choose organization from actual reads, writes 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 21 | lines 5-9

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.

Fundamental of Database Systems module | PDF page 116 | lines 25-28

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 18-29

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

Question sources

Computer Science MCQ practice collection · Question 173

Question 201 | AM0174

Users may follow many other users, and each user may be followed by many users. Which option describes the cardinality of this recursive user relationship, rather than a key constraint or procedural enforcement mechanism?

A. Primary key constraints

B. Foreign key constraints

C. Many-to-many relationships

D. Triggers and stored procedures

Hint

Identify relationship degree and cardinality before choosing a relational mapping.

Answer C: C. Many-to-many relationships

The follow rules permit multiple related users in both directions, which is many-to-many cardinality. A recursive association table can store two user references to represent those links. Primary and foreign keys support identification and referential integrity, but they are constraint mechanisms rather than the requested multiplicity classification. Triggers and stored procedures are also not a cardinality category.

Why each choice is right or wrong

A - Incorrect: Primary keys identify rows rather than name relationship multiplicity.

B - Incorrect: Foreign keys validate references rather than describe the many-to-many cardinality.

C - Correct: Both endpoints permit many partners, establishing M:N.

D - Incorrect: Procedures are implementation mechanisms rather than a cardinality type.

Primary note proof

Study-note section FDB12

> A recursive M:N friendship uses UserLink(UserID,OtherUserID), two foreign keys to User; decide whether links are directed and whether self-links are allowed. A recursive 1:N supervision uses a nullable SupervisorID reference in Employee. For a ternary Supply(SupplierID,PartID,ProjectID,Quantity), preserve all participating references; binary replacements can lose which triple occurred. Choose the association key from actual cardinalities rather than automatically assuming every reference is necessary.

Study-note section FDB07

> A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB15

> Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed.

Related reading (context, not primary note proof)

Study-note section FDB09

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 11 | lines 7-15

> Super Key − A set of attributes (one or more) that collectively identifies an entity in an entity set. Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key. Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set. Relationship A relationship type represents the association between entity types. For example, ‗Enrolled in' is a relationship type that exists between entity Student and Course. In ER diagram, relationship is represented by a diamond and connecting the entities with lines.

Fundamental of Database Systems module | PDF page 12 | lines 2-3

> Many to many (M-M) - When entities in all entity sets can take part more than once in the relationship cardinality is many to many.

Fundamental of Database Systems module | PDF page 48 | lines 13-17

> iv. Referential Integrity • A ―foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key • Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Question sources

Computer Science MCQ practice collection · Question 174

Question 202 | AM0175

In the usual normalization sequence, which is the first normal form excluding transitive dependence of nonprime attributes on a key through a nonkey determinant?

A. First Normal Form (1NF)

B. Second Normal Form (2NF)

C. Third Normal Form (3NF)

D. Boyce-Codd Normal Form (BCNF)

Hint

Check atomicity, candidate keys and partial dependencies in that order.

Answer C: Third Normal Form (3NF)

In the stated single-key teaching case, 2NF excludes partial dependencies but can still allow a nonprime fact to depend on the key through a nonkey determinant. The next defining stage is 3NF, summarized by the whole-key-and-nothing-but-the-key mnemonic. BCNF strengthens the determinant rule, and 4NF addresses multivalued dependencies. Stronger forms inherit earlier conditions, so the explicit first-stage scope makes 3NF the unique selection rather than treating inherited properties as rival answers.

Why each choice is right or wrong

A - Incorrect: Incorrect: 1NF concerns atomic values and does not eliminate the specified transitive dependencies.

B - Incorrect: Incorrect: 2NF excludes partial dependencies but can retain a nonprime transitive dependency through a nonkey determinant.

C - Correct: Correct: 3NF is the first stage imposing the specified nonprime transitive-dependency restriction in this teaching case.

D - Incorrect: Incorrect: BCNF strengthens 3NF by requiring every nontrivial determinant to be a superkey; it is not the first stage described.

Primary note proof

Study-note section FDB20

> 1NF requires atomic values in the chosen relational domains and no repeating groups in cells.

Study-note section FDB20

> 2NF requires 1NF and no nonprime attribute partially dependent on any candidate key.

Study-note section FDB21

> For every nontrivial FD X→A, 3NF permits it when X is a superkey or A is prime. BCNF requires X to be a superkey. The mnemonic whole key and nothing but the key describes the common single-key case; it cannot replace the full tests when keys overlap. Remove dependencies of nonprime facts on nonkey determinants.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 17 | lines 10-15

> Second normal form (2NF) Second normal form (2NF) is based on the concept of full functional dependency. A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on the primary key (i.e. the non-prime attributes are not partially dependent on key attributes). Remove any partially dependent attributes and place them in another relation. A partial dependency is when the attribute(s) is/are dependent on a part of a key.

Fundamental of Database Systems module | PDF page 18 | lines 5-8

Third Normal Form (3NF) A relation schema R is in third normal form (3NF) if it is in 2NF and every non-prime attribute of _R‘ are fully functionally dependent on every key of R and non-transitively dependent on every key of R.

Question sources

Computer Science MCQ practice collection · Question 175

Question 203 | AM0176

What is the purpose of denormalization in database design for real-world scenarios?

A. To eliminate data redundancy

B. To optimize query performance

C. To enforce referential integrity

D. To simplify the database schema

Hint

Compare design techniques by the specific problem each solves.

Answer B: To improve selected query performance.

Denormalization repeats or precomputes facts so that selected queries can perform less work, such as fewer joins. FDB38 also states the cost: more maintenance when underlying facts change. It does not aim to eliminate redundancy or enforce foreign-key references. A simpler query path may result, but a simpler schema is not the essential purpose.

Why each choice is right or wrong

A - Incorrect: Denormalization introduces controlled redundancy instead of eliminating it.

B - Correct: Its goal is to reduce costs of selected read queries using repeated or precomputed information.

C - Incorrect: Foreign-key constraints enforce references; denormalization itself does not establish referential integrity.

D - Incorrect: It may simplify selected reads, but schema simplicity is not the defining or guaranteed goal.

Primary note proof

Study-note section FDB38

Normalization reduces dependency-driven anomalies. Indexing supplies search paths. Denormalization deliberately repeats or precomputes facts to reduce selected read costs, increasing maintenance work. Partitioning divides rows or columns; replication creates copies to improve locality or availability while complicating updates. These techniques can coexist. An e-commerce or gaming label alone does not select one.

Study-note section FDB15

Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed.

Study-note section FDB15

Key integrity enforces uniqueness.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 17-27

4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship.

Question sources

Computer Science MCQ practice collection · Question 176

Question 204 | AM0178

Which listed modeling technique directly includes superclass/subclass inheritance? Interpret ER modeling in A as basic ER without the EER extension, relational modeling as ordinary tables/keys, and XML modeling as document structure.

A. Entity-Relationship (ER) modeling

B. Object-oriented data modeling

C. Relational data modeling

D. XML data modeling

Hint

Review inheritance, specialization and category constraints.

Answer B: B. Object-oriented data modeling

Object-oriented data modeling directly represents superclass/subclass inheritance. Enhanced ER also supports inheritance, but A is explicitly limited to basic ER in this question. Ordinary relational tables can encode subtype data through references without making inheritance a direct modeling construct, and document structure is not the object inheritance mechanism being tested. The scope distinction makes the selected technique definite.

Why each choice is right or wrong

A - Incorrect: Basic ER excludes the EER inheritance extension under the stated scope.

B - Correct: Object models directly support superclass/subclass inheritance.

C - Incorrect: Ordinary relational tables encode facts and references rather than directly declaring this inheritance construct.

D - Incorrect: Document structure is not the direct object inheritance model asked for.

Primary note proof

Study-note section FDB13

> EER extends ER with superclass/subclass structure. A subclass inherits the superclass attributes and relationships. Specialization divides Student into Undergraduate and Graduate; generalization combines common properties into Student. Disjoint subclasses cannot overlap; overlapping subclasses can. Total specialization requires every superclass member in some subclass; partial specialization does not.

Study-note section FDB13

> Object models also support inheritance; distinguish basic ER notation from EER when comparing models.

Study-note section FDB13

> Structural EER inheritance alone does not define executable object methods.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 8-11

> Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model.

Fundamental of Database Systems module | PDF page 12 | lines 25-31

> 3.5 Enhanced Entity Relationship (EER) Model The complexityof the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling. To reduce this complexity of modeling we have to make improvements or enhancements to the existing ER model to make it able to handle the complex application in a better way. Enhanced ER (EER) model is a database model that incorporates the concepts of class/subclass relationships and type inheritance into the ER model. The EER model includes all the modeling

Fundamental of Database Systems module | PDF page 13 | lines 2-8

> concepts of the ER model. In addition, it includes the concepts of subclass and superclass, categories (UNION types), attribute and relationship inheritance. EER is created to design more accurate database schemas. 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.

Question sources

Computer Science MCQ practice collection · Question 178

Question 205 | AM0179

Which CMS features directly support either semantic grouping of content or valid stored references between content and tags? Select all that apply.

A. Referential integrity constraints

B. Full-text search capabilities

C. Content categorization and tagging

D. Distributed database architecture

Hint

Compare design techniques by the specific problem each solves.

Answer A, C: A. Referential integrity constraints; C. Content categorization and tagging

Categories and tags describe semantic associations among content items. Referential integrity constraints keep the stored links between content and tags valid. These two features address meaning and reference validity respectively, so both fit. Full-text search supports finding text, while a distributed architecture concerns data placement and coordination. Neither is the requested direct representation or validation of content-to-tag relationships.

Why each choice is right or wrong

A - Correct: Referential constraints preserve valid links between content and related rows, such as tags.

B - Incorrect: Full-text search finds textual content rather than primarily defining or validating item relationships.

C - Correct: Categories and tags describe semantic groupings and associations among content items.

D - Incorrect: Distribution determines placement and coordination rather than directly representing the stated content relationships.

Primary note proof

Study-note section FDB38

In a CMS, tags/categories describe meaning, while foreign keys preserve valid references between content and tags. A ContentTag(ContentID,TagID) association models multiple tags per item; these functions complement each other.

Study-note section FDB15

Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed.

Study-note section FDB15

Key integrity enforces uniqueness.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 27-31

3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Fundamental of Database Systems module | PDF page 48 | lines 13-17

iv. Referential Integrity ● A —foreign key‖ links each occurrence in a relation representing a child entity to the occurrence of the parent entity containing the matching candidate (usually primary) key ● Referential Integrity means that if the foreign key contains a value, that value must refer to an existing occurrence in the parent entity

Fundamental of Database Systems module | PDF page 116 | lines 25-28

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 179

Question 206 | AM0180

Which database design technique is used to represent a hierarchical structure of data elements?

A. Relational data modeling

B. Network data modeling

C. Hierarchical data modeling

D. Object-relational data modeling

Hint

Compare the data models by how they organize records and relationships.

Answer C: Hierarchical data modeling.

A hierarchical structure organizes elements into parent-child levels. FDB02 assigns this representation to the hierarchical model, so C directly matches the requested design concept. Relational modeling instead uses tables, network modeling permits more general record links, and richer object-relational types do not by themselves require a hierarchical organization.

Why each choice is right or wrong

A - Incorrect: Relational modeling represents facts in tables rather than requiring a hierarchical tree.

B - Incorrect: Network modeling permits linked records with multiple parents rather than a restricted hierarchy.

C - Correct: Hierarchical modeling explicitly represents the parent-child structure asked for in the stem.

D - Incorrect: An object-relational approach adds richer types to relational storage rather than defining this tree organization.

Primary note proof

Study-note section FDB02

> A data model supplies concepts for structure, relationships, constraints and operations. Conceptual ER models describe entities and relationships. Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents. Object models represent objects with identity, state and behavior. A physical model specifies record layouts and access paths. These are different representation choices, not stages of one query.

Study-note section FDB-S02

> Richer relational types do not by themselves require all data to form a parent-child tree; that representation is the defining concern of hierarchical modeling.

Study-note section FDB-S02

> An object-relational DBMS retains relational data management while supporting richer extensions such as user-defined data types, functions and operators.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 8-19

> Types of Data Models 1. High-level or conceptual data models provide concepts that are close to the way many users perceive data. Entity - Relationship (E/R) model and Object-Oriented Model are examples of conceptual/high level data model. 2. Low-level or physical data models provide concepts that describe the details of how data is stored on the computer storage media. Concepts provided by physical data models are generally meant for computer specialists, not for end users. 3. representational (or implementation) data model provides concepts that may be easily understood by end users but that are not too far from the way data is organized in computer

> storage. Representational data models hide many details of data storage on disk but can be implemented on a computer system directly. The widely used relational data models are the network and hierarchical.

Fundamental of Database Systems module | PDF page 13 | lines 5-8

> 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.

Question sources

Computer Science MCQ practice collection · Question 180

Question 207 | AM0181

Which benefit directly follows from reducing dependency-driven insertion, deletion and update anomalies through normalization, without assuming reduced storage or simpler queries?

A. Simplified data retrieval

B. Increased data redundancy

C. Enhanced data integrity

D. Reduced storage requirements

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer C: C. Enhanced data integrity

Normalization separates facts according to their determinants, reducing repeated facts and the insertion, deletion and update anomalies they cause. That directly supports enhanced data integrity. Storage savings and simpler retrieval are not guaranteed because decomposition can introduce additional keys and joins. Increased redundancy is opposite to the intended reduction of avoidable repetition, so it is not the benefit identified by this question.

Why each choice is right or wrong

A - Incorrect: Normalized retrieval may require more joins, so simplification does not follow.

B - Incorrect: Normalization aims to reduce avoidable repetition.

C - Correct: Reducing inconsistent updates and related anomalies improves integrity.

D - Incorrect: Storage savings can be offset by extra keys and relation overhead.

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly

prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Study-note section FDB19

Reduced redundancy can improve integrity and storage use, but extra keys and joins mean neither storage savings nor faster retrieval is guaranteed.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 16 | lines 5-29

4.3 Database Anomalies Database Anomalies are the flaws in databases which occurs because of poor database design and storing everything in a single flat database. They are problems that are caused usually by redundant data. Types of database anomalies are Insertion anomaly, Deletion anomaly, and Update anomaly. Insertion Anomaly - occurs when the entire primary key is not known and the database cannot insert a new record properly. This would violate entity integrity. It can be avoided by using a sequence number for the primary key. Deletion Anomaly - happens when a record is deleted that results in the loss of other related data. It can be avoided by normalizing tables in a database to minimize their dependency. Update Anomaly - occurs when a change to one attribute value causes the database to either contain inconsistent data or causes multiple records to need changing. It may be prevented by making sure tables are in third normal form. 4.4 Normalization Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. It is a technique of organizing the data in the database. It is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables. Normal forms Redundancy in a relation may causes insertion, deletion, and update anomalies. Normalization is database design technique that minimizes redundancy from a relation or a set of relations. Normalization rules divides larger tables into smaller tables and links them using relationship. Normal Forms are maintenance rules that are used to eliminate or reduce the redundancy in database relations. These normal forms are 1NF (First Normal Form), 2NF (Second Normal Form)

Question sources

Computer Science MCQ practice collection · Question 181

Question 208 | AM0182

Which database design concept allows the division of a large database into smaller, manageable units?

A. Data partitioning

B. Database replication

C. Data migration

D. Data archiving

Hint

Distinguish logical records, disk pages and physical file organization.

Answer A: Data partitioning.

FDB24 explicitly contrasts partitioning, replication, migration and archiving. Partitioning divides the dataset into smaller portions, matching the question. Replication makes additional copies, migration moves data, and archiving takes older data out of active service. Those operations can accompany partitioning, but none of them defines division into manageable units.

Why each choice is right or wrong

A - Correct: Partitioning divides a large dataset into smaller portions or manageable units.

B - Incorrect: Replication creates copies rather than dividing the original dataset into distinct units.

C - Incorrect: Migration moves data to another location or system rather than defining a split.

D - Incorrect: Archiving removes older data from active service rather than generally dividing the database.

Primary note proof

Study-note section FDB24

> A page/block is a unit of database I/O and holds several records. For fixed unspanned records of size r in usable page space B, the blocking factor is floor(B/r), and n records occupy ceil(n/blocking factor) pages. Page headers and slots reduce usable space.

Study-note section FDB24

> Partitioning splits a large dataset; replication copies it; migration moves it; archiving removes older data from active service.

Study-note section FDB24

> A heap has no search-key order and supports simple inserts, but searches may scan all pages.

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 15-30

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

Fundamental of Database Systems module | PDF page 118 | lines 2-18

• A relation can be fragmented in two ways: • 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.

Question sources

Computer Science MCQ practice collection · Question 182

Question 209 | AM0183

Which database design technique is used to capture the relationships between entities in a graphical representation?

A. Relational data modeling

B. Entity-Relationship (ER) modeling

C. Object-oriented data modeling

D. Document data modeling

Hint

Review entity, attribute and relationship notation before interpreting the model.

Answer B: Entity-Relationship (ER) modeling

ERD is explicitly a graphical database-schema technique showing modeled entities and their important relationships. This directly establishes ER modeling for the graphical representation asked about.

Why each choice is right or wrong

A - Incorrect: Relational modeling describes tables; ERD is the named graphical technique. (PDF pp. 9.)

B - Correct: ER modeling is the explicit graphical entities/relationships technique. (PDF pp. 9.)

C - Incorrect: Object-oriented is named but not the described ERD technique. (PDF pp. 9.)

D - Incorrect: Document modeling is not the technique defined in the module. (PDF pp. 9.)

Primary note proof

Study-note section FDB06

> An entity is a distinguishable real or conceptual thing, such as one student or one course. An entity type describes common properties; its current collection of entities is an entity set. A relationship associates entities. In Chen notation rectangles represent entity types, diamonds relationships and ellipses attributes; double ellipses denote multivalued attributes and dashed ellipses derived attributes. Always inspect the diagram legend.

Study-note section FDB02

> Conceptual ER models describe entities and relationships.

Study-note section FDB02

> Relational models use tables; hierarchical models organize records in a parent-child tree; classic network models connect records through pointer-based owner/member links, allowing multiple parents.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 9 | lines 27-31

> 3.3 ERD (Entity Relationship Diagram) ERD a graphical technique for representing a database schema. It is a modeling tool used to depict graphically a DB design before it is actually implemented. It shows the various entities being modeled and the important relationships among them. ERD has three basic components Entity, Relationship and Attribute.

Question sources

Computer Science MCQ practice collection · Question 183

Question 210 | CS0049

In contrary to the traditional file-based approach, in the database approach:

A. Programs define and access their own data

B. Store data items in multiple copies

C. Don't care about the definition of programs and data

D. Data and programs are defined independently

Hint

Trace which schema levels a proposed change affects.

Answer D: Data and programs are defined independently

The database approach insulates programs from data definitions and supports data independence. This is the relevant distinction from programs managing their own files.

Why each choice is right or wrong

A - Incorrect: Own program-specific definitions do not express the database approach's insulation. (PDF pp. 5, 6, 8.)

B - Incorrect: Multiple uncontrolled copies are not the defining benefit and can cause inconsistency. (PDF pp. 5, 6, 8.)

C - Incorrect: The approach explicitly maintains metadata and definitions, not indifference to them. (PDF pp. 5, 6, 8.)

D - Correct: Independent definitions match program/data insulation and independence. (PDF pp. 5, 6, 8.)

Primary note proof

Study-note section FDB04

> Physical data independence permits changing internal storage without changing the conceptual schema, for example adding an index. Logical data independence permits conceptual changes while preserving affected external views and application interfaces, for example splitting a table while maintaining a compatible view. It is harder because applications depend on logical meaning. Data abstraction hides irrelevant details; program-data insulation separates data definitions from application code.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 5 | lines 17-25

> 1.2 Database System versus File System The traditional file processing system is file-directory structure supported by a conventional operating system. A file system organization of data lucks a number of major features of a database system, such as Data redundancy and inconsistency, Difficulty in accessing data, Data isolation, Integrity problems, Atomicity problems, Concurrent access anomalies. 1.3 Characteristics of the Database Approach The main characteristics of the database approach are Self-describing nature, Data abstraction and Insulation between programs and data, Support of multiple views of the data, sharing of data and multiuser transaction processing.

Fundamental of Database Systems module | PDF page 6 | lines 2-4

> Data abstraction and Insulation between programs and data- Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Fundamental of Database Systems module | PDF page 8 | lines 8-15

2.4 Data Independence The three-schema architecture can be used to further explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. The two types of data independence: 1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. 2. Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas do not need be changed as well.

Question sources

Computer Science Exit Exam practice collection · Question 49

Question 211 | CS0050

Assume you are a database designer and you are given a task to specify entities, relationships, constraints on relations and data types for the data to be stored. At which level of the three-schema architecture
would this be addressed?

A. View level

B. Internal level

C. External level

D. Conceptual level

Hint

Trace which schema levels a proposed change affects.

Answer D: Conceptual level

The conceptual level describes the whole database and explicitly concentrates on entities, data types, relationships, operations and constraints while hiding physical storage. The stem repeats those conceptual responsibilities.

Why each choice is right or wrong

A - Incorrect: View level shows a user-group subset. (PDF pp. 7, 8.)

B - Incorrect: Internal level describes physical storage and access paths. (PDF pp. 7, 8.)

C - Incorrect: External is another name for user view, not the global description. (PDF pp. 7, 8.)

D - Correct: Conceptual is the level explicitly listing the requested structural components. (PDF pp. 7, 8.)

Primary note proof

Study-note section FDB04

The external level gives each user group a selected view. The conceptual level describes the whole logical database: entity types, relationships, data types and constraints. The

internal/physical level describes files, record placement and access paths. These are abstraction levels, distinct from client/server deployment tiers.

Module passages - related reading

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

Fundamental of Database Systems module | PDF page 7 | lines 25-30

2.3 Three-schema Architecture and Data Independence The goal of the three-schema architecture is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels: 1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Fundamental of Database Systems module | PDF page 8 | lines 2-7

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. 3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Question sources

Computer Science Exit Exam practice collection · Question 50

Question 212 | NS0217

What is the purpose of normalization in database design?

A. To reduce redundancy and improve data integrity

B. To speed up query performance

C. To create complex relationships between tables

D. To encrypt sensitive data

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer A: Reduce avoidable redundancy and associated integrity anomalies.

FDB19 defines normalization as separating facts according to their determinants, reducing avoidable repetition and insertion, deletion and update anomalies. That supports both parts of A. It does not promise universally faster queries, deliberately create complexity for its own sake or encrypt content. Performance and security need their own workload and threat-model choices.

Why each choice is right or wrong

A - Correct: Normalization reduces repeated facts and the anomalies that threaten data consistency.

B - Incorrect: More joins may be needed, so faster query execution is not a universal normalization purpose.

C - Incorrect: Relationships follow fact dependencies; creating complexity is not the goal.

D - Incorrect: Normalization restructures facts and does not encrypt sensitive values.

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Question sources

Computer Science Exit Exam practice collection · Question 53

Question 213 | NS0295

A weak entity uses its partial key together with the key of an independently keyed owner. Which terms describe that associated entity set? Select all that apply.

A. Identifying set

B. Strong entity set

C. Neighbor set

D. Owner set

Hint

Check how a dependent entity obtains its complete identity.

Answer B, D: B. Strong entity set; D. Owner set

The weak entity lacks a complete independent identifier. Its owner's key combines with the partial key to identify it. Because the owner is independently keyed in this question, strong entity set and owner set both describe the associated set. Identifying describes the relationship connecting the sets, while neighbor does not state the necessary ownership and identification dependency.

Why each choice is right or wrong

A - Incorrect: Identifying is normally the relationship role rather than the defined name of the owner entity set.

B - Correct: The owner that supplies an independent identifying key is normally a strong entity set.

C - Incorrect: Neighbor does not express the necessary ownership and identification dependency.

D - Correct: Owner set is the associated entity set whose key identifies the weak entity together with its partial key.

Primary note proof

Study-note section FDB08

> A weak entity lacks a complete independent key. Its owner is normally a strong entity, so owner entity set and strong entity set may both describe the associated set. Its partial key, also called discriminator, distinguishes weak entities of the same owner. Identification combines the owner key and partial key.

Study-note section FDB08

> The weak entity participates totally in its identifying relationship.

Study-note section FDB08

> Chen notation uses a double rectangle, double identifying diamond and dashed underline for a partial key.

Question sources

Computer Science Exit Exam · 2018 · Question 12

Question 214 | NS0297

A language that allows the DBA or user to describe and name the entities, attributes, and relationships required for the application is_____

A. Transaction control language (TCL)

B. Data control language (DCL)

C. Data Manipulation Language (DML)

D. Data Definition Language (DDL)

Hint

Classify each SQL command by the kind of change it performs.

Answer D: Data Definition Language.

Describing and naming application entities, attributes and relationships defines schema structure. FDB26 assigns structural changes to DDL through CREATE, ALTER and DROP. TCL manages transaction boundaries, DCL controls privileges and DML retrieves or changes data. These categories have different effects, so DDL uniquely matches the schema-description role in the stem.

Why each choice is right or wrong

A - Incorrect: TCL manages commit, rollback and savepoints rather than defining schema elements.

B - Incorrect: DCL manages privileges through GRANT and REVOKE rather than describing entities and attributes.

C - Incorrect: DML retrieves or changes instance data rather than defining the application schema.

D - Correct: DDL defines and changes schema structure, including tables, attributes and constraints.

Primary note proof

Study-note section FDB26

> DDL changes structure: CREATE, ALTER, DROP, commonly RENAME and TRUNCATE in textbook classifications. DML retrieves or changes data: SELECT, INSERT, UPDATE, DELETE. Some texts separate SELECT as DQL. DCL manages privileges through GRANT and REVOKE. TCL includes COMMIT, ROLLBACK and SAVEPOINT. Classify a statement by its effect, not its length.

Question sources

Computer Science Exit Exam · 2018 · Question 14

Question 215 | NS0302

In relational-table terminology, which terms describe a unit of related field values treated together by an application? Select all that apply.

A. Rows

B. Field

C. Record

D. Database

Hint

Distinguish a relation's structure from its current contents and its descriptive metadata.

Answer A, C: A. Rows; C. Record

A tuple holds a set of related attribute values as one table unit. Row and record are synonymous names for that unit, so both alternatives qualify. A field is only an individual property, and a database is the larger coherent collection of related data. Grammatical number in Rows does not create a different relational concept from Record.

Why each choice is right or wrong

A - Correct: Rows contain related field values and are the relational counterpart of records.

B - Incorrect: A field is an individual property rather than a collection of related fields.

C - Correct: Record is the stated unit that groups related fields for application use.

D - Incorrect: A database contains a larger coherent collection of data rather than just one record.

Primary note proof

Study-note section FDB03

> A relation is a set of tuples, represented as a table. A tuple/row/record contains related attribute values; an attribute/column/field describes a property. Degree counts columns; cardinality counts rows. Classical relations have distinct tuples, atomic cells and domains for attributes; row order has no meaning. Column order has no logical meaning when attributes are identified by name. SQL tables can contain duplicate rows unless constrained.

Question sources

Computer Science Exit Exam · 2018 · Question 20

Question 216 | NS0315

Which template creates a table with one column? Treat table_name, column_name and column_type as placeholders for valid identifiers and a data type.

A. CREATE table_name (column_name, column type);

B. CREATE table_name (column_type, column name);

C. CREATE TABLE table_name (column_name column_type);

D. CREATE TABLE table_name (column_type, column name);

Hint

Distinguish table definition changes from changes to stored rows.

Answer C: C. CREATE TABLE table_name (column_name column_type);

CREATE TABLE introduces a new table definition. Within its parentheses, each column declaration places the column name before its data type, without a comma between those two parts. Commas separate whole declarations. Option C matches this pattern. A and B omit TABLE, while D reverses and separates the intended name/type pair and therefore fails the requested template.

Why each choice is right or wrong

A - Incorrect: TABLE is absent and the column declaration is split incorrectly.

B - Incorrect: TABLE is absent and the name/type order is reversed.

C - Correct: The name/type pair forms one complete column declaration.

D - Incorrect: The type/name order and comma do not form the intended declaration.

Primary note proof

Study-note section FDB30

> CREATE TABLE Employee (EmpID INTEGER PRIMARY KEY, Name VARCHAR(80)); declares two columns. The name and its type stay together; the comma separates whole column declarations. To add a primary key later: ALTER TABLE Employee ADD PRIMARY KEY (EmpID); existing data must be unique and non-NULL. A table cannot gain a second primary key without replacing the existing constraint.

Question sources

Computer Science Exit Exam · 2018 · Question 40

Question 217 | NS0320

From the following alternatives which one is FALSE about database system?

A. It is very difficult to protect a file under the system

B. In database system the user is not required to write the procedures

C. In database system, DBMS provides a good protection mechanism.

D. It contains a wide variety of sophisticated techniques to store and retrieve the data.

Hint

Review the database approach and match each claimed benefit or cost to its mechanism.

Answer A: The claim that protection is very difficult under a database system is false.

FDB01 contrasts unmanaged file difficulties with DBMS authorization, shared integrity, transactions and recovery. A therefore reverses a stated database benefit. B is understood in the data-access sense: declarative requests need not prescribe retrieval procedures, as FDB36 explains; it does not mean applications never contain code. Protection and sophisticated storage/access facilities in C and D are consistent with the note.

Why each choice is right or wrong

A - Correct: This reverses the note's benefit: a DBMS provides coordinated protection rather than inherently making protection very difficult.

B - Incorrect: Declarative data requests need not prescribe retrieval procedures; this does not exclude ordinary application code.

C - Incorrect: Authorization and controlled data access are explicit DBMS protection mechanisms.

D - Incorrect: DBMS storage and access facilities include indexes, buffers, query processing and other sophisticated mechanisms.

Primary note proof

Study-note section FDB01

> A database is a coherent collection of related facts with meaning for an application. A DBMS is software that defines, stores, retrieves, updates, protects and shares that data. The database is the content; the DBMS manages it. A database system also includes applications, people and infrastructure.

Study-note section FDB01

> Traditional file applications define their own formats and access procedures.

Study-note section FDB01

> A DBMS centralizes metadata and supports multiple views, authorization, transactions, backup and recovery.

Related reading (context, not primary note proof)

Study-note section FDB36

Study-note section FDB25

Question sources

Computer Science Exit Exam · 2018 · Question 45

Question 218 | NS0329

Consider a directed line(->) from the relationship set advisor to both entity sets instructor and student. This indicates _____ cardinality

A. Many to one

B. Many to many

C. One to one

D. One to many

Hint

Read the relationship in both directions and check participation constraints.

Answer C: One-to-one under the common ER arrow convention.

FDB09 explicitly discusses arrows from advisor to both Instructor and Student. Under its common convention, each arrow imposes a maximum of one on the corresponding end. Both ends are therefore maximum-one, producing 1:1. This does not imply total participation, and another diagram notation would require a legend, but the complete note supplies the convention for this example.

Why each choice is right or wrong

A - Incorrect: Many-to-one permits many at one end, contrary to maximum-one arrows at both ends.

B - Incorrect: Many-to-many imposes neither of the maximum-one constraints stated by the arrows.

C - Correct: Maximum one at both ends produces a one-to-one relationship.

D - Incorrect: One-to-many permits multiple associations at one end rather than maximum one at both.

Primary note proof

Study-note section FDB09

> Read business rules in both directions. If each employee belongs to one department and each department has many employees, Employee→Department is N:1; Department→Employee is 1:N. Building→Location could also be N:1, so names alone cannot select a unique example. Under the common arrow convention, arrows from advisor to both Instructor and Student impose maximum-one at both ends, giving 1:1. Without a notation convention, arrows are unsafe evidence.

Question sources

Computer Science Exit Exam · 2018 · Question 56

Question 219 | NS0339

What is the basic goal of Normalization?

A. To decrease data integrity

B. To reduce redundancy

C. To maximize transitive dependency

D. To increase dependency

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer B: Reduce avoidable redundancy.

Normalization separates facts by their determinants to reduce repeated storage and the anomalies that repetition can cause. FDB19 therefore supports reducing redundancy as the basic listed goal. Decreasing integrity is the opposite intended effect, and maximizing transitive or unspecified dependencies is not a normalization objective. Proper dependencies are analyzed rather than increased indiscriminately.

Why each choice is right or wrong

A - Incorrect: Normalization aims to improve integrity by reducing anomalies, not decrease it.

B - Correct: Reducing avoidable repeated facts is a basic normalization goal.

C - Incorrect: Problematic nonkey transitive dependencies are removed or separated rather than maximized.

D - Incorrect: Dependencies are analyzed to place facts correctly; increasing them indiscriminately is not the objective.

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Study-note section FDB21

> Remove dependencies of nonprime facts on nonkey determinants.

Question sources

Computer Science Exit Exam · 2018 · Question 70

Question 220 | NS0344

For a relation with ID, CITY and NAME, ID is guaranteed unique and non-NULL while CITY and NAME may repeat. Which listed sets are superkeys? Select all that apply.

A. NAME

B. ID

C. CITY

D. CITY, ID

Hint

Test uniqueness and minimality separately when classifying keys.

Answer B, D: B. ID; D. CITY, ID

A superkey is any set of attributes guaranteed to identify a tuple. ID has that guarantee, and the larger set containing CITY and ID keeps it. Both sets therefore qualify, even though the larger one is not minimal. CITY and NAME may repeat and cannot independently identify all tuples. Minimality would distinguish candidate keys, but it is not required for superkeys.

Why each choice is right or wrong

A - Incorrect: Repeated names are permitted, so NAME is not guaranteed unique.

B - Correct: ID uniquely identifies every tuple.

C - Incorrect: Repeated cities are permitted, so CITY is not guaranteed unique.

D - Correct: Including unique ID preserves identification even with the extra CITY attribute.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB07

> If ID is unique, both {ID} and {ID,City} are superkeys, but only {ID} is minimal.

Study-note section FDB07

> A table can have several candidates and foreign keys but at most one declared primary-key constraint, possibly containing several columns.

Question sources

Computer Science Exit Exam · 2018 · Question 77

Question 221 | NS0357

_____is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

A. Super key

B. Candidate key

C. Social security key

D. Primary key

Hint

Test uniqueness and minimality separately when classifying keys.

Answer D: Primary key.

FDB07 defines a primary key as the selected candidate key, with the remaining candidates called alternate keys. The stem asks specifically for the designer's selection rather than every unique attribute set. Superkey is broader and candidate key denotes eligibility before selection. A social-security identifier could be chosen, but is not the generic name of that role.

Why each choice is right or wrong

A - Incorrect: A superkey is any unique identifier set and may include unnecessary attributes.

B - Incorrect: Candidate keys are eligible minimal identifiers, not specifically the one selected for the primary role.

C - Incorrect: A social-security identifier is a possible attribute choice, not the generic selected-key category.

D - Correct: The selected candidate key is the primary key.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Question sources

Computer Science Exit Exam · 2018 · Question 95

Question 222 | NS0627

What is the difference between a primary key and a foreign key?

Hint

Test uniqueness and minimality separately when classifying keys.

Answer: A primary key is the selected unique non-NULL identifier; a foreign key references an eligible key.

FDB07 defines the primary key as a selected candidate key. FDB15 requires non-NULL foreign-key values to match an eligible referenced unique/candidate key; foreign keys need not be unique and may permit NULL.

Primary note proof

Study-note section FDB07

> A superkey uniquely identifies a tuple. A candidate key is a minimal superkey: removing any component loses uniqueness. Minimal means no unnecessary attribute, not necessarily the fewest columns among all keys. A primary key is one selected candidate key; remaining candidates are alternate keys. A composite key contains multiple attributes and need not be the chosen primary key.

Study-note section FDB15

> Referential integrity requires a non-NULL foreign-key value to match an eligible referenced unique/candidate key; self-references are allowed.

Study-note section FDB15

> Entity integrity requires every primary-key component to be non-NULL.

Related reading (context, not primary note proof)

Study-note section FDB16

Question sources

Database Fundamentals review questions · Question 1

Question 223 | NS0628

Why is database normalization important? What problems does it solve?

Hint

Distinguish dependency-based design problems from the contents of one sample table.

Answer: Normalization reduces repeated facts and insertion, deletion and update anomalies.

Repeating a student name across enrollments risks inconsistent updates, prevents storing some facts without unrelated rows and can lose a fact when its last repeated row is deleted. Normalization separates facts by their determinants to reduce these anomalies.

Primary note proof

Study-note section FDB19

> Suppose Enrollment repeats StudentName for every course. An update anomaly requires changing every copy of the student's name; missed copies disagree. An insertion anomaly prevents recording a new student until some enrollment exists. A deletion anomaly loses the only record of a student's details when their last enrollment is deleted. Revoke anomaly is not a normalization anomaly.

Study-note section FDB19

> Normalization separates facts according to their determinants, reducing avoidable repetition and associated anomalies.

Study-note section FDB19

> Adding a generated row ID may make insertion technically possible, but does not remove repeated facts or fix the dependency design.

Question sources

Database Fundamentals review questions · Question 2

Question 224 | NS0629

What happens if you UPDATE a row but forget to include a WHERE clause?

Hint

Track which rows each data-changing command affects.

Answer: UPDATE without WHERE targets every row, subject to constraints and transaction outcome.

FDB29 states that an UPDATE without WHERE targets every row. The SET expressions are therefore applied to all rows rather than one intended row; constraints or errors can still prevent completion.

Primary note proof

Study-note section FDB29

> INSERT INTO Hotel(HotelID,HotelName,SubCity) VALUES (1,'Hilton','Yeka'); pairs each named column with a compatible value. Counts and types must match; keys and other constraints still apply. Without a column list, values must follow the table's declared column order.

Study-note section FDB29

> UPDATE without WHERE targets every row.

Study-note section FDB29

> UPDATE Book SET Price = 600 WHERE ID = 'B001'; changes matching rows.

Related reading (context, not primary note proof)

Study-note section FDB16

Question sources

Database Fundamentals review questions · Question 3

Question 225 | NS0630

Why would you use a ROLLBACK command in a transaction?

Hint

Match each transaction guarantee to its precise failure or interference scenario.

Answer: ROLLBACK undoes uncommitted transaction work, for example after a failure or unwanted change.

FDB37 contrasts COMMIT, which completes transaction work, with ROLLBACK, which undoes uncommitted work. Rollback restores the prior transactional state after errors or a decision not to keep the changes; it does not erase other users' committed transactions.

Primary note proof

Study-note section FDB37

> For a transfer, subtracting 100 from one balance and adding 100 to another must commit together; losing either half violates atomicity and may violate the total-balance invariant. BEGIN starts work, COMMIT completes it, ROLLBACK undoes uncommitted work and SAVEPOINT marks an intermediate rollback point. A failed transaction may roll back while other users' committed transactions remain. SQL transaction syntax and error-handling constructs differ by DBMS; @@ERROR/IF is SQL Server-specific.

Question sources

Database Fundamentals review questions · Question 4

Question 226 | NS0631

What is the difference between DELETE and DROP?

Hint

Classify each SQL command by the kind of change it performs.

Answer: DELETE removes rows and retains the table; DROP TABLE removes the table object.

FDB26 distinguishes data deletion from structural removal. DELETE may target selected rows with WHERE, or all rows without it, while leaving the table defined. DROP TABLE removes that relation object.

Primary note proof

Study-note section FDB26

> DELETE removes rows and keeps the table; DROP TABLE removes the table object; ALTER changes its definition. TRUNCATE empties a table using implementation-specific mechanisms. PostgreSQL allows transactional rollback of TRUNCATE, so no rollback information is retained is not a universal SQL property. SQL Server permits DROP DATABASE Sales, suppliers; that comma-list syntax is vendor-specific. SQLite's .databases lists attached names/files in its command-line shell; .database is not the documented command.

Study-note section FDB29

> UPDATE without WHERE targets every row.

Study-note section FDB29

> DELETE FROM r; removes every row but leaves r defined.

Question sources

Database Fundamentals review questions · Question 5