CS Exit Ethiopia

Revision Outlines

Computer Science Exit Examination

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

Fundamental of Database Systems

Revision Outlines

Database approach, benefits and costs

- Database versus DBMS
  - Coherent related application facts
  - DBMS defines/stores/retrieves/updates
  - DBMS protects/shares
  - Content versus management software
  - Database system applications/people/infrastructure
- File approach versus DBMS
  - Application-specific formats/access procedures
  - Repeated facts/update disagreement
  - Data isolation
  - Shared integrity difficulties
  - Concurrent update difficulties
  - Centralized metadata
  - Multiple views/authorization/transactions
  - Backup/recovery
  - Controlled redundancy ≠ zero duplication
  - Complexity/administration/storage/licensing costs
  - Tiny fixed application tradeoffs
  - Lightweight embedded DBMSs
  - Useful concurrent access
  - Uncontrolled concurrency → anomalies
- Database personnel
  - DBA access authorization
  - Backup/recovery/availability
  - Monitoring/tuning/routine maintenance
  - Coordinating database use
  - Designers → schema
  - Developers → user-facing programs

Data models and product names

- Data model choices
  - Structure/relationships/constraints/operations
  - Conceptual ER → entities/relationships
  - Relational → tables
  - Hierarchical → parent-child tree
  - Network → pointer-based owner/member links
  - Network permits multiple parents
  - Objects → identity/state/behavior
  - Physical → layouts/access paths
  - Representations ≠ query stages
- Products versus vendor/language
  - Relational PostgreSQL/MySQL/Oracle Database
  - Relational Microsoft SQL Server/IBM Db2
  - MongoDB → document DBMS
  - Java → programming language
  - IBM → vendor
  - IBM answer needs product name

Relations, schemas, instances and metadata

- Relation terminology
  - Relation → set of tuples
  - Tuple/row/record → attribute values
  - Attribute/column/field → property
  - Degree → columns
  - Cardinality → rows
  - Classical distinct tuples
  - Atomic cells/attribute domains
  - Meaningless row order
  - Named attributes → meaningless column order
  - SQL duplicates unless constrained
- Schema versus instance
  - Schema → names/structures/constraints

- Instance/state/snapshot → current data
- Student insertion → state change
- Column addition → schema change
- Metadata → data describing data
- Catalog/data dictionary
- Definitions/types/constraints
- Privileges/optimizer statistics
- Schema diagram → tables/key references
- Database diagram terminology overlap

Three-schema architecture and data independence | Exam focus

- Three-schema abstraction
  - External → user-group views
  - Conceptual → whole logical database
  - Entity types/relationships/types/constraints
  - Internal → files/placement/access paths
  - Abstraction levels ≠ deployment tiers
- Data independence
  - Physical → internal changes
  - Index addition preserves conceptual schema
  - Logical → conceptual changes
  - Preserved external views/application interfaces
  - Table split with compatible view
  - Logical independence harder
  - Applications depend on logical meaning
  - Abstraction hides irrelevant detail
  - Program-data insulation separates definitions/code

Requirements through implementation | Exam focus

- Requirements through implementation
  - Business facts/transactions/reports
  - Volumes/privacy/quality targets
  - Conceptual design → ER/EER

- Logical design → relations/keys/constraints
- Logical normalization check
- Physical → organization/indexes/storage
- Implementation → create/load objects
- Testing → requirements

• Design methodology qualifications
- Logical design tied to model
- Potential DBMS independence
- Later vendor-specific refinement
- Universal dependence claims need methodology
- Conceptual enterprise view → downward
- Normalization dependencies → upward
- Useful approaches ≠ mandatory directions

Entities, attributes and ER notation | Exam focus

• ER concepts and Chen notation
- Entity → distinguishable real/conceptual thing
- Student/course instances
- Entity type → common properties
- Entity set → current collection
- Relationship → entity association
- Rectangle → entity type
- Diamond → relationship
- Ellipse → attribute
- Double ellipse → multivalued
- Dashed ellipse → derived
- Inspect diagram legend

• Independent attribute dimensions
- Simple versus composite
- Address → City/Street
- Single-valued versus multivalued
- Student → several phone numbers
- Composite single-valued address

- Age derived from BirthDate
- Derived can also be single-valued
- Names alone cannot establish multiplicity

Superkeys, candidate keys and primary keys | Exam focus

- Key hierarchy
  - Superkey → tuple uniqueness
  - Candidate key → minimal superkey
  - Remove component → uniqueness lost
  - Minimal ≠ globally fewest columns
  - Primary → selected candidate
  - Alternate → remaining candidate
  - Composite → multiple attributes
  - Composite need not be primary
- Key assumptions and traps
  - Unique ID → {ID} superkey
  - {ID,City} nonminimal superkey
  - Several candidates/foreign keys
  - At most one primary-key constraint
  - Multicolumn primary key allowed
  - String keys allowed
  - Prefer stable compact identifiers
  - Email/SSN/EmployeeID uniqueness needs rules
  - Single-column options ≠ composite example
- Worked example Eligible versus preferred primary keys
  - CustID/Email/ContactNo candidate keys
  - All eligible
  - CustID preference needs stability/size policy
  - Uniqueness alone cannot choose

Weak entities and identifying relationships | Exam focus

- Weak entity identification
  - No independent complete key

- Owner normally strong entity
- Owner/strong entity-set terminology overlap
- Partial key = discriminator
- Distinguishes same-owner weak entities
- Owner key + partial key

• Dependent weak-entity mapping
- Names unique within employee
- Key (EmployeeID,DependentName)
- EmployeeID foreign key
- Total identifying participation
- Chen double rectangle
- Double identifying diamond
- Partial-key dashed underline
- Foreign key alone ≠ weak entity

Cardinality, participation and direction | Exam focus

• Relationship cardinality and degree
- Maximum → 1:1/1:N/M:N
- Minimum zero → optional/partial
- Minimum ≥1 → mandatory/total
- Maximum one ≠ compulsory participation
- Degree → entity roles
- Unary recursive/binary/ternary

• Direction and notation assumptions
- Employee→Department N:1
- Department→Employee 1:N
- Building→Location possibly N:1
- Names alone insufficient
- Advisor arrows to both entities
- Common convention → maximum-one both ends
- Resulting 1:1
- Unknown convention → unsafe arrow inference

Map entities, composite and multivalued attributes | Exam focus

- Strong-entity mapping
  - One relation per entity type
  - Preserve simple attributes
  - Select candidate key
  - Expand composite components
  - Flat atomic relational design
  - Calculate derived values
  - Avoid stale stored copies
- Multivalued-attribute mapping
  - Separate owner-key/value relation
  - StudentPhone(StudentID,Phone)
  - Primary key (StudentID,Phone)
  - StudentID foreign key
  - Separate phone rows
  - No comma-separated lists
  - No Phone1/Phone2 repetition
  - Explicit omitted mapping step
  - Entity-based table names valid

Map 1:1 and 1:N relationships | Exam focus

- 1:N relationship mapping
  - One-side key → many-side FK
  - Relationship attributes on many side
  - Department/Employee DeptID reference
  - HireDate relationship attribute
  - Total participation → NOT NULL
  - Partial → nullable unassigned reference
- 1:1 relationship mapping
  - Foreign key plus UNIQUE
  - Prefer totally participating side
  - Reduce NULLs
  - NOT NULL → mandatory reference

- Merge only compatible lifecycles/participation
- A→B many-to-one reference in A
- Total or partial participation allowed
- Total-only mapping claim overstated

Map M:N, recursive and n-ary relationships | Exam focus

- M:N association table
  - Enrollment(StudentID,CourseID,Grade)
  - References Student and Course
  - One enrollment/pair → composite key
  - Repeated attempts → Term/AttemptNo key
  - Ordinarily three tables
- Recursive and ternary mapping
  - UserLink(UserID,OtherUserID)
  - Two User foreign keys
  - Directed versus undirected links
  - Self-link policy
  - Employee nullable SupervisorID
  - Recursive 1:N supervision
  - Supply(SupplierID,PartID,ProjectID,Quantity)
  - Preserve participating references/triples
  - Binary replacements can lose triples
  - Association key from actual cardinalities
  - All references not automatically necessary

EER inheritance and category constraints | Exam focus

- EER inheritance constraints
  - Superclass/subclass
  - Inherited attributes/relationships
  - Specialization → Undergraduate/Graduate
  - Generalization → common Student properties
  - Disjoint → no overlap
  - Overlapping → shared members

- Total → every superclass member
- Partial → membership not compulsory

• Inheritance and category mapping
- Person(PersonID,Name)
- Student(PersonID,Program)
- Subclass PersonID both PK/FK
- Preserved inherited identity
- Category/union → Person or Company
- Intersection → both superclasses
- Object inheritance also supported
- ER answer requires EER scope
- Structural inheritance ≠ executable methods

Fan traps and chasm traps

• Fan trap
- Common entity fans into two 1:N
- Department→Employee and Department→Project
- Path cannot determine WorksOn
- Department join → every employee/project pair
- Record WorksOn explicitly

• Chasm trap
- Optional relationship breaks path
- Unstaffed branch property disappears
- Add required direct relationship
- Or revise model
- Multiple 1:N relationships legitimate
- Eliminating all fans ≠ universal repair
- Missing path → chasm
- Fan → association ambiguity

Integrity and constraint taxonomy | Exam focus

• Integrity categories
- Domain → type/allowed values

- Key → uniqueness
- Entity → every PK component non-NULL
- Referential → non-NULL FK matches key
- Eligible unique/candidate referenced key
- Self-references allowed
- Parent need not have child

• Constraint mechanisms and interpretation

- Inherent → data model
- Schema-based → DDL
- Semantic/application → triggers/application logic
- DDL expressiveness limits
- Integrity → validity
- Authorization → allowed access
- Uniqueness/domain validation both integrity
- NULL Department PK → entity violation
- Manager=107 missing employee
- Declared FK → referential violation
- One INSERT → multiple violations

Write and enforce SQL constraints | Exam focus

• SQL constraint declaration

- Enrollment StudentID/CourseID INTEGER NOT NULL
- REFERENCES Student(StudentID)/Course(CourseID)
- Grade INTEGER CHECK
- Grade BETWEEN 0 AND 100
- PRIMARY KEY (StudentID,CourseID)
- Column name then type
- Commas separate declarations
- CHECK clause ≠ standalone GRANT-like statement
- POSITIVE not general SQL keyword
- CHECK (Amount > 0)

• Constraint NULL and reference behavior

- PRIMARY KEY → unique/non-NULL

- UNIQUE need not imply NOT NULL
- UNIQUE NULL treatment DBMS-dependent
- CHECK unknown can pass
- NOT NULL forbids missing values
- NO ACTION/RESTRICT
- CASCADE
- SET NULL
- Referenced keys can change
- Children/action determine legality

Functional dependencies and attribute closure | Exam focus

- Functional dependencies
  - X→Y: agreeing X implies agreeing Y
  - Every legal tuple pair
  - Business constraint ≠ sample correlation
  - Full → all determinant attributes
  - Partial → proper determinant subset
  - Prime → some candidate key
  - Nonprime → no candidate key
- Closure and Armstrong axioms
  - Start X^+ with X
  - Applicable FD → add RHS
  - Repeat until fixed point
  - X^+ covers relation ↔ superkey
  - Remove/recompute → minimality
  - Reflexivity: $Y \subseteq X \Rightarrow X \rightarrow Y$
  - Augmentation: $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$
  - Transitivity: $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$

Find all keys before deciding normal form | Exam focus

- Find every candidate key
  - R(A,B,C,D)
  - C→D,C→B,B→C

- A absent from RHS
- Every key needs A
- (AB)+=ABCD via B→C→D
- (AC)+=ABCD
- AB/AC minimal
- BC cannot derive A
- Prime A/B/C
- Nonprime D

• Highest normal form with partial dependencies
- C→D and B→D
- Proper subsets of AB/AC
- 2NF fails
- Atomic values → highest 1NF
- Read functional dependencies accurately
- C3D,C>PB,BOC alone insufficient
- Request actual FD set
- Never calculate guessed symbols

• Worked example Transitive dependency key test
- R(A,B,C), F={A→B,B→C}
- A+=ABC → key A
- Nonkey B → 3NF violation
- Candidate-key B changes conclusion
- Chain alone insufficient

Insertion, deletion and update anomalies

• Normalization anomalies
- Repeated Enrollment StudentName
- Update → every copy must change
- Missed copies → disagreement
- Insertion → student needs enrollment
- Deletion → last enrollment loses student
- Revoke ≠ normalization anomaly

• Normalization benefits and limits

- Separate facts by determinants
- Reduce repetition/anomalies
- Declared FDs suffice
- Bad current data unnecessary
- Generated row ID ≠ dependency repair
- Repeated facts persist
- Potential integrity/storage improvement
- Extra keys/joins
- Storage savings not guaranteed
- Faster retrieval not guaranteed

First and second normal forms | Exam focus

• First normal form
- Atomic chosen-domain values
- No repeating cell groups
- Primary-key declaration insufficient
- Correct composite/multivalued mapping → 1NF
- ER drawing ≠ higher-form guarantee

• Second normal form
- 1NF prerequisite
- No nonprime partial candidate-key dependency
- Enrollment key (SID,CID)
- SID→SName
- CID→CTitle
- Student(SID,SName)
- Course(CID,CTitle)
- Enrollment(SID,CID,Grade)
- Grade → whole pair
- Every candidate single-attribute → no partials
- No partials alone needs 1NF assumption
- Higher forms inherit 2NF

Third normal form, BCNF and higher forms | Exam focus

- 3NF versus BCNF
  - Every nontrivial X→A
  - 3NF: X superkey or A prime
  - BCNF: X superkey
  - Whole-key mnemonic limited
  - Overlapping keys need full tests
  - Remove nonprime/nonkey-determinant dependencies
- Sale normalization diagnosis
  - Key (Date,Customer,Product)
  - Product→Vendor
  - Vendor→VendorCity
  - Product proper key subset
  - Vendor/VendorCity violate 2NF
  - Atomic → highest 1NF
  - Transitivity does not imply highest 2NF
  - 4NF → nontrivial MVDs
  - Nonsuperkey MVD determinants
  - 5NF → join dependencies
  - MVD ≠ list inside cell
- Worked example Transitive Employee decomposition
  - EID→DeptID
  - DeptID→DeptName
  - Employee(EID,DeptID)
  - Department(DeptID,DeptName)
  - Department name stored once

Decomposition: lossless and dependency preserving | Exam focus

- Lossless decomposition
  - Join reconstructs exact legal tuples
  - Lossy → spurious combinations
  - Binary FD losslessness test
  - Common attributes determine R1 or R2

- Shared column alone insufficient

• Preservation and decomposition algorithms
  - Dependency preservation → local constraint enforcement
  - No join required
  - Separate from losslessness
  - 3NF synthesis → minimal FD cover
  - Relation per determinant/dependents
  - Remove contained relations
  - Add candidate-key relation if absent
  - BCNF → offending X→Y
  - Separate XY and remainder with X
  - BCNF may sacrifice dependency preservation

Choose data types and storage levels | Exam focus

• Data types by meaning
  - CHAR(n) fixed length
  - VARCHAR(n) variable bounded length
  - DATE → date
  - BLOB → binary/images
  - INTEGER → whole numbers
  - DECIMAL(p,s) → exact decimal
  - Money needs specified scale
  - FLOAT/REAL/DOUBLE approximate
  - Names/widths/precision DBMS-dependent
  - Reject universal numeric-width claims

• Storage hierarchy and type selection
  - Primary → RAM/processor cache
  - Usually volatile
  - Secondary → persistent online SSD/HDD
  - Pages copied to CPU buffers
  - Tertiary → tape/removable archives
  - Types from meaning/range/accuracy
  - Integers not universally faster than strings

Records, pages and primary file organization

- Pages and blocking calculations
  - Page/block → I/O unit
  - Several records/page
  - Fixed unspanned record size r
  - Usable page space B
  - Blocking factor = floor(B/r)
  - Pages = ceil(n/blocking_factor)
  - Headers/slots reduce usable space
- File organization tradeoffs
  - Heap → unordered/simple inserts
  - Heap search → possible full scan
  - Sorted → range scans/binary search
  - Ordering-field search
  - Costly inserts/overflow handling
  - Hashed → hash(key) bucket
  - Equality lookup/collisions/overflow
  - Hash order unsuitable for ordinary ranges
  - Choose from reads/writes/predicates
  - Partitioning → split
  - Replication → copy
  - Migration → move
  - Archiving → older data leaves active service

Ordered, dense, sparse, and multilevel indexes

- Index purpose and costs
  - Search keys plus pointers/row identifiers
  - Selective retrieval benefit
  - Space overhead
  - Write maintenance
  - Tiny table → scan can win
  - Most rows → scan can win
  - Index ≠ referential-integrity enforcement

- Index varieties and assumptions
  - Primary ordered → file ordering key
  - One entry/page → sparse
  - Clustering → nonunique ordering field
  - Often one entry/distinct value
  - Secondary → alternative access paths
  - Every searchable value represented
  - Record pointers/pointer lists
  - Dense → each search value
  - Sparse → ordered intervening data
  - Multilevel → indexes on indexes
  - B/B+ trees → ordered/range search
  - Hashing → equality
  - Bitmap → some analytic workloads
  - No universally best index

SQL categories and exact scope

- SQL command categories
  - DDL → structure
  - CREATE/ALTER/DROP
  - Textbook RENAME/TRUNCATE often DDL
  - DML → retrieve/change data
  - SELECT/INSERT/UPDATE/DELETE
  - SELECT sometimes DQL
  - DCL → GRANT/REVOKE privileges
  - TCL → COMMIT/ROLLBACK/SAVEPOINT
  - Classify by effect, not length
- Deletion and dialect scope
  - DELETE → rows removed/table retained
  - DROP TABLE → object removed
  - ALTER → definition changed
  - TRUNCATE → table emptied
  - Implementation-specific mechanisms

- PostgreSQL TRUNCATE rollback allowed
- No-rollback claim not universal
- SQL Server DROP DATABASE Sales, suppliers
- Comma list vendor-specific
- SQLite shell .databases → names/files
- .database undocumented command

SELECT and its result columns

• SELECT output semantics
- SELECT Name,Address FROM Customer
- Mailing-label fields only
- SELECT * → all columns
- DISTINCT → duplicate result removal
- AS → alias
- ALL → duplicate permission
- ALL ≠ all columns
- Default SQL preserves duplicates
- Column reads require appropriate FROM
- Literal SELECT 1 can omit FROM

• Written versus conceptual query order
- SELECT/FROM/WHERE/GROUP BY/HAVING/ORDER BY
- FROM/JOIN → WHERE
- GROUP BY → HAVING
- SELECT expressions → ORDER BY
- SELECT normally preserves state
- WHERE alone ≠ executable query
- Clause/statement distinction

Predicates, strings and NULL

• SQL predicate syntax
- Strings → single quotes
- WHERE ID = 'B001'
- Standard double quotes → identifiers

- Vendor string modes differ
- E.Department resolves ambiguity
- Comparisons =/<>/</>
- AND/OR/NOT
- Parentheses clarify precedence
- LIKE 'Ab%' → prefix
- BETWEEN includes endpoints

• NULL and timestamp boundaries

- NULL → missing/unknown
- Salary = NULL → unknown
- IS NULL/IS NOT NULL
- WHERE retains true only
- BirthDate >= DATE '2000-01-01'
- BirthDate < DATE '2001-01-01'
- Half-open timestamp year interval
- No December 31 time omissions
- Expression indexes support functions
- Functions not universally unindexable

INSERT, UPDATE and DELETE

• INSERT column/value correspondence

- INSERT INTO Hotel(HotelID,HotelName,SubCity)
- VALUES (1,'Hilton','Yeka')
- Counts/types must match
- Keys/constraints still apply
- No column list → declared order

• UPDATE and DELETE scope

- UPDATE Book SET Price = 600
- WHERE ID = 'B001'
- Matching rows only
- No UPDATE WHERE → every row
- DELETE FROM Publisher WHERE Name='XWZ'
- Reference policies still apply

- DELETE FROM r → all rows
- Table definition remains
- Clear entries/delete rows synonymous
- Several UPDATEs can be valid
- Appropriateness needs requested change/dialect

CREATE and ALTER tables

• CREATE TABLE and primary keys
- Employee EmpID INTEGER PRIMARY KEY
- Name VARCHAR(80)
- Name/type stay together
- Commas between column declarations
- ALTER TABLE Employee ADD PRIMARY KEY (EmpID)
- Existing values unique/non-NULL
- Second PK requires replacing existing constraint

• ALTER and malformed syntax
- ADD COLUMN Email VARCHAR(120)
- PostgreSQL-style syntax
- DROP COLUMN Email
- ADD CONSTRAINT uq_email UNIQUE (Email)
- Named constraint
- DALTER malformed token
- Column-name/type comma malformed
- Repair ≠ printed statement executes

Join behavior and preservation

• Join matching semantics
- Inner → matching ON pairs
- Equijoin → equality
- Theta join → other comparison
- Natural → every same-named column
- Ordinarily inner equality join
- Natural merges duplicate join columns

- USING (DeptID) explicit shared columns
- CROSS JOIN → every pair
- No matching condition

• Outer joins and join arity
- LEFT preserves unmatched left
- RIGHT preserves unmatched right
- FULL preserves both
- Missing-side columns NULL-padded
- Inner/natural-inner discard unmatched rows
- Self-join → aliases/base-table occurrences
- Operator combines two inputs
- Inputs may be earlier joins
- Query may join many tables
- Base-table count versus occurrence count
- Binary arity distinction

Aggregation, GROUP BY and HAVING

• Aggregation and group filtering
- COUNT(*) → rows
- COUNT(x) → non-NULL x
- SUM/AVG → non-NULL numeric values
- AVG, not AVERAGE
- GROUP BY → distinct-value groups
- WHERE filters individual rows
- HAVING filters aggregate groups

• Grouped query validity
- Department/AVG(Salary) output
- WHERE Department <> 'Sales'
- GROUP BY Department
- HAVING AVG(Salary) > 35000
- Ordinary aggregate predicate invalid in WHERE
- EmpID/FirstName/LastName not grouped
- Unless DBMS establishes functional dependence

- Grouping column need not be selected
- Damaged query defect ≠ omitted grouping output

Subqueries and views

• Subquery forms and NULL traps
- Nested SELECT
- WHERE/HAVING/FROM locations
- IN → membership
- EXISTS → qualifying row existence
- Correlated → outer-row reference
- Scalar → appropriate single value
- NOT IN with possible NULL
- NOT EXISTS often safer absence test

• Views and rewrite tradeoffs
- CREATE VIEW CSStudent AS SELECT
- StudentID/Name WHERE Major='CS'
- Stored query definition/virtual table
- Repeated-query simplification
- Derived values
- Configured access restriction
- Ordinary view need not store result
- Materialized view stores result
- Subquery→join can change duplicates
- Rewrite not guaranteed faster

Set operations and SQL conversion

• SQL set operations
- UNION → compatible results/deduplication
- UNION ALL → duplicates retained
- UNION DISTINCT dialect-dependent
- PostgreSQL accepts explicit DISTINCT
- Not separate algebra rule
- INTERSECT → common rows

- EXCEPT → first-only rows
- Same column count
- Compatible corresponding types

• Conversion and UNION repair
- CAST('42' AS INTEGER)
- Value must permit conversion
- CONVERT vendor-specific conversion
- Both options need dialect
- UNION xyz FROM table2 malformed
- Missing second SELECT
- SELECT expressions versus table names
- xyz ≠ table name

Relational algebra: calculate results

• Relational algebra operators
- Selection σ → tuple filtering
- Projection π → attributes
- Set projection removes duplicates
- Rename ρ → relation/attribute names
- Union/difference/intersection → compatible schemas
- R−S not commutative
- Union/intersection commutative
- Inner joins commutative with adjusted predicates
- Up to attribute order
- LEFT OUTER not generally commutative
- Vague join needs variant

• Product counts and algebra basis
- $|R \times S| = |R| \times |S|$
- Product degree = degree(R)+degree(S)
- Rename conflicting attributes
- 200×100=20,000
- 5×7=35
- Join = selection over product

- Intersection derivable from difference
- Basis σ/π/union/difference/product/rename
- Assignment useful, not indispensable
- Textbook primitive lists vary

Relational calculus: state the desired facts

• Algebra versus calculus
- Algebra → operation sequence
- Procedural at logical level
- Calculus → desired-result predicate
- Declarative/nonprocedural
- Tuple variables → tuples
- Domain variables → attribute values
- Safe/range-restricted expressions
- Results bounded by active domain

• Calculus forms and quantifiers
- {s | Student(s) AND s.Major='CS'}
- Tuple result → CS students
- Domain result → names n
- ∃i,m (Student(i,n,m) AND m='CS')
- Existential → some matching tuple
- Universal → every required case
- SQL declarative
- Optimizer chooses procedural execution plan

Transactions and ACID

• Transactions and ACID
- Transaction → logical operation unit
- Atomicity → all effects or none
- Consistency → correct invariant preservation
- Isolation → concurrent interference control
- Serializable → equivalent serial execution
- Durability → committed crash survival

- Durable logging typical
- Immediate page forcing unnecessary

• Transfer and transaction control
- Subtract/add 100 commit together
- Lost half → atomicity violation
- Possible total-balance invariant violation
- BEGIN → start
- COMMIT → complete
- ROLLBACK → undo uncommitted work
- SAVEPOINT → intermediate rollback point
- Other committed transactions remain
- Syntax/error-handling DBMS-dependent
- @@ERROR/IF SQL Server-specific

Choose techniques from requirements | Exam focus

• Match technique to requirement
- Normalization → dependency anomalies
- Indexing → search paths
- Denormalization → repeat/precompute facts
- Lower selected read costs
- Higher maintenance cost
- Partitioning → rows/columns divided
- Replication → copies
- Locality/availability benefits
- Update complexity
- Techniques can coexist
- E-commerce/gaming label insufficient

• Security and quality requirements
- Privacy → threat model
- Authentication → identity
- Role authorization → records/actions
- Encryption → at rest/in transit
- Auditing → activity records

- Patient data → encryption/access control
- Scalability/availability/retrieval speed
- Redundancy control/security
- Functional → behavior
- Nonfunctional → quantified quality
- 99.9% uptime/response time
- Mechanisms alone cannot classify requirement

• CMS meaning versus integrity
- Tags/categories → meaning
- Foreign keys → valid references
- ContentTag(ContentID,TagID)
- Multiple tags/item
- Complementary functions

Registration and dormitory design | Exam focus

• Registration/dormitory business rules
- Student → one department
- Student/course → many-to-many
- Student → at most one current dorm
- Dorm → at most six students
- Department/Student/Course/Dorm entities
- Enrollment relationship with Grade

• Integrated schema and capacity enforcement
- Department(DeptID,Name)
- Student(SID,Name,DeptID,DormID)
- Course(CID,Title)
- Enrollment(SID,CID,Grade)
- Dorm(DormID,FloorNumber)
- Entity IDs → primary keys
- Student DeptID/DormID → parent FKs
- Enrollment PK (SID,CID)
- Enrollment → both parent references
- Nullable DormID until assignment

- StudentPhone(SID,Phone) → multiple contacts
- Department/course facts in owning tables
- FK alone cannot enforce six capacity
- Concurrency-safe trigger/allocation transaction
- Or six unique slots/dorm

• Worked example Dorm reference direction
- FOREIGN KEY (DormID) REFERENCES Dorm(DormID)
- STUDENT → referencing child
- DORM → referenced parent
- Consistent names
- Student.DormID required before execution

Check SQL syntax and question assumptions

• Check supplied answer keys
- Owner/strong entity both possible
- Row/record synonyms
- Difference/set-difference synonyms
- Negative DROP question → multiple answers
- DELETE/UPDATE/invalid REMOVE
- Highlighting does not establish correctness

• Resolve question defects
- Damaged token → corrected form
- Printed malformed SQL remains invalid
- Unsupported cardinality → rules/direction
- Best/efficient/important → workload/goal/baseline
- Diagram conclusions need symbols/description
- Question index → exceptions/minimal clarification
- Unsupported unique answer ≠ useless teaching