ENDTERM PAPER CONCISE DMBD 2024

PYQ PAPER SOLUTION (Formatted with Tables & Enhanced Readability - No Leading Spaces)

Question 1 (Compulsory)

a) Describe the components of database systems. [PYQ Q1a from problem, implies general importance] (3 Marks)

- **Definition:** A database system is an integrated set of components for defining, creating, managing, and using databases.
- **Main Components:**

Component	Description
1. Hardware	Physical devices supporting the database system (e.g., servers, storage disks, network devices, memory).
2. Software	Programs governing database operation and management.
	Database Management System (DBMS): Core software for database interaction.
	Query Processor: Parses, optimizes, executes queries (includes DDL interpreter, DML compiler, query evaluation engine).
	Storage Manager: Manages physical data storage (buffer, file, transaction management - concurrency/recovery).
	Application Programs & Utilities: Software to access/manipulate data, generate reports, backups (e.g., report generators, import/export tools).
3. Data	Collection of facts stored in the database.
	Operational Data: Day-to-day operational information.
	Metadata (Data about Data): Describes structure, constraints of operational data (stored in data dictionary/repository).
4. Users	Individuals interacting with the database.
	Database Administrators (DBAs): Manage overall system (security, backup, performance).
	Database Designers: Define database structure (schema).
	Application Programmers: Develop applications interfacing with the database.
	End Users: Access database for tasks (querying, reporting, updating).
5. Procedures	Instructions and rules for database design and use (e.g., login, backup/recovery methods, standards).

---

b) Distinguish between primary key, candidate key and super key. [PYQ Q1b from problem, and in notes] (3 Marks)

Feature	Superkey	Candidate Key	Primary Key
Definition	Any attribute or set of attributes that uniquely identifies a tuple (row) within a relation.	A minimal superkey. It is a superkey from which no attribute can be removed without losing its uniqueness property.	The candidate key chosen by the database designer to uniquely identify tuples in a relation.
Minimality	Not necessarily minimal. It may contain redundant attributes.	Must be minimal.	Must be minimal (as it's a chosen candidate key).
Uniqueness	Guarantees uniqueness for each tuple.	Guarantees uniqueness for each tuple.	Guarantees uniqueness for each tuple.
Number	A relation can have many superkeys.	A relation can have one or more candidate keys.	A relation has only one primary key.
Null Values	Generally, attributes part of a superkey (if chosen as CK/PK) should not be null.	Attributes forming a candidate key should ideally not accept null values.	Cannot contain null values (Entity Integrity Constraint).
Example	If {SID, CID} is CK, then {SID, CID, SName} is a superkey.	In Employee, {EmpID} and {SSN} could be candidate keys.	From CKs {EmpID}, {SSN}, DBA might choose {EmpID} as PK.

---

c) Explain the advanced data manipulation using SQL. [PYQ - from problem and in notes] (3 Marks)

- **Definition:** Advanced DML in SQL extends basic operations for complex data retrieval, analysis, and modification.
- **Key Features:**
- **1. Complex Queries:**
- **Joins:** Combining data from multiple tables (`INNER`, `LEFT`, `RIGHT`, `FULL`, `CROSS JOIN`).
- **Subqueries (Nested Queries):** Queries within other SQL statements (`WHERE`, `SELECT`, `FROM`, `HAVING`), including correlated subqueries.
- **Aggregate Functions with Grouping:** `COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()` with `GROUP BY` (calculations on groups) and `HAVING` (filtering groups).
- **2. Set Operations:** Combining `SELECT` results using `UNION [ALL]`, `INTERSECT`, `EXCEPT`/`MINUS`.
- **3. Window Functions:** Calculations across related rows without collapsing them (e.g., ranking, moving averages).

- **4. Common Table Expressions (CTEs):** `WITH` clause for temporary, named result sets, improving readability/modularity.
- **5. Bulk Operations:** Efficient large-volume data operations (e.g., `INSERT INTO ... SELECT ...`).

---

d) Explain the Data Control Language (DCL) commands. [PYQ - implied in notes under DBMS components/security] (3 Marks)

- **Definition:** DCL commands in SQL manage permissions and control access to database objects.
- **Primary DCL Commands:**
- **1. `GRANT`:**
- **Purpose:** Gives specific privileges (e.g., `SELECT`, `INSERT`, `UPDATE`) on objects (tables, views) to users/roles.
- **Syntax (Simplified):** `GRANT privilege_list ON object_name TO user_list [WITH GRANT OPTION];`
- **`WITH GRANT OPTION`:** Allows grantee to further grant received privileges.
- **2. `REVOKE`:**
- **Purpose:** Removes previously granted privileges from users/roles.
- **Syntax (Simplified):** `REVOKE [GRANT OPTION FOR] privilege_list ON object_name FROM user_list [CASCADE | RESTRICT];`
- **`GRANT OPTION FOR`:** Revokes only the grant ability.
- **`CASCADE`:** Propagates revocation to users granted by this user.
- **`RESTRICT`:** Fails if privilege was passed on (often default).

---

e) Find the minimum number of tables required to represent the given ER diagram in the relational model. [PYQ - related to ER to Relational Mapping rules] (3 Marks)

- **Analysis:**
- **Entity M (Strong):** 1 table: `Table_M (M1_PK, M2, M3)`
- **Entity P (Strong):** 1 table: `Table_P (P1_PK, P2)`
- **Entity N (Weak, identifying R2 with P):** 1 table: `Table_N (P1_FK_PKpart, N1_PKpart, N2)`
- **Relationship R1 (M to P, 1:N):** PK of M (`M1`) becomes FK in `Table_P`. No new table. `Table_P` becomes `(P1_PK, P2, M1_FK)`.
- **Relationship R2 (P to N, M:1, Identifying):** Handled by `Table_N`'s structure. No additional table.
- **Resulting Tables:**

1. `Table_M (M1_PK, M2, M3)`
2. `Table_P (P1_PK, P2, M1_FK)` (references `Table_M`)
3. `Table_N (P1_FK_PKpart, N1_PKpart, N2)` (references `Table_P`)

- **Conclusion:** Minimum number of tables required is **3**.

UNIT-I

Question 2:

a) Explain the database systems architecture with a suitable diagram. [PYQ Q2a from problem, related to ANSI/SPARC] (8 Marks)

- **ANSI/SPARC Three-Schema Architecture:** Standard architecture for data independence.

Level	Description	Content	Purpose
1. External Level	Highest level, user/application specific views (Subschemas).	Multiple external schemas, each showing a relevant portion of the DB.	Customized views, security, simplified interaction.
2. Conceptual Level	Unified, comprehensive logical view of the entire database (Global View / Schema).	All entities, attributes, relationships, constraints; technology-independent.	Stable DB description, basis for views, hides physical details.
3. Internal Level	Lowest level, physical storage details (Physical View / Schema).	Data structures, file organizations (B+-trees, hashing), access paths, storage allocation; technology-dependent.	Efficient data storage and retrieval.

- **Data Independence:**
- **Logical Data Independence:** Modify conceptual schema without changing external schemas/apps (e.g., add attribute).
- **Physical Data Independence:** Modify internal schema without changing conceptual/external schemas (e.g., add index).

---

b) Describe the database design life cycle. [PYQ Q2b from problem, and in notes] (7 Marks)

- **Definition:** DDLC is a systematic, phased approach for designing, implementing, and maintaining a database.
- **Phases:**

Phase	Objective	Key Activities	Output
1. Requirements Specification & Planning	Understand & document complete user data needs.	User interviews, review documents, identify business rules, define scope.	Detailed requirements specification.
2. Conceptual Data Modeling	Create high-level, technology-independent data model.	Use ERD/EERD, define entities, attributes, relationships, constraints. Presentation/Design-Specific layers.	Conceptual schema (e.g., ER diagram). Validation.

Phase	Objective	Key Activities	Output
3. Logical Data Modeling	Transform conceptual model to a chosen data model (typically relational).	Map ER/EER to tables, columns, PKs, FKs. Apply normalization.	Logical schema (e.g., normalized relational tables).
4. Physical Data Modeling	Design DBMS-specific internal storage and access methods.	Define data types, create indexes, plan file organization, clustering.	Physical schema (DDL statements).
5. Implementation & Testing	Create & test the database and its functionality.	Execute DDL, populate data, develop/test applications.	Operational database, tested applications.
6. Deployment, Operation & Maintenance	Put DB into production and ensure its continued smooth operation.	Monitor, tune, manage security, backups/recovery, apply updates.	Functioning and maintained database system.

UNIT-II

Question 4:

a) Explain the ER model and EER Model to map with logical schema. [PYQ Q4a from problem, and in notes] (8 Marks)

- **Mapping ER Model to Relational Schema:**
- **Strong Entity:** New table; attributes become columns; choose PK.
- **Weak Entity:** New table; PK = PK of owner (as FK) + partial key.
- **1:1 Relationship:**
- **Option 1 (Foreign Key):** PK of one entity as unique FK in the other.
- **Option 2 (Merged Relation):** If both total participation, merge tables.
- **Option 3 (Relationship Relation):** New table with PKs from both entities.
- **1:N Relationship:** PK of '1'-side as FK in 'N'-side table. Relationship attributes on 'N'-side.
- **M:N Relationship:** New junction table; PK = composite of PKs of both entities (as FKs). Relationship attributes here.
- **Multi-valued Attribute:** New table; PK = PK of original entity (as FK) + multi-valued attribute.
- **Mapping EER Model to Relational Schema:**
- **Specialization/Generalization (SC/sc):**
- **Option 1 (SC + sc's tables):** Table for SC, table for each sc (PK of SC is PK & FK in sc).
- **Option 2 (SC table only):** One table for SC, includes all sc attributes (use NULLs, type attribute).
- **Option 3 (sc's tables only):** Table for each sc, includes inherited SC attributes (SC total & disjoint).

- **Specialization Hierarchy/Lattice:** Apply chosen SC/sc mapping recursively. Shared subclass (lattice) may have multiple FKs if Option 1.
- **Categorization (Union Type):** Table for category (surrogate PK). Tables for superclasses. Category PK is FK in each superclass table.

Question 5:

a) Define Normalization. Explain the types of Normalization. [PYQ Q5a from problem, and in notes] (8 Marks)

- **Normalization Definition:** Process of organizing database data to reduce redundancy and improve integrity by decomposing tables.
- **Goals:** Minimize redundancy, eliminate anomalies (insertion, update, deletion), ensure logical dependencies.
- **Normal Forms (Types):**

Normal Form	Rule
1NF	All attribute values are atomic (no repeating groups/multi-valued attributes in a cell).
2NF	1NF + No partial dependencies (non-key attributes fully depend on the *entire* PK).
3NF	2NF + No transitive dependencies (non-key attributes don't depend on other non-key attributes).
BCNF	Stricter 3NF. Every determinant (X in X -> Y) must be a superkey.
4NF	BCNF + No non-trivial multi-valued dependencies (MVDs: X ->> Y) unless X is a superkey.
5NF (PJ/NF)	4NF + No join dependencies (JDs) not implied by candidate keys (ensures lossless decomposition).

---

b) Explain the Mapping of higher degree relationships. [PYQ Q5b from problem, and in notes] (7 Marks)

- **Definition:** Higher-degree relationships (ternary, n-ary) involve >=3 entity types.
- **Mapping Approach (Associative Entity / Junction Table):**
- **1. Create New Relation (`S`):** For the n-ary relationship `R`.
- **2. Include Foreign Keys:** Add PK of each participating entity `Ei` as an FK in `S`.
- **3. Determine PK of `S`:** Typically, composite of all included FKs: `{PK(E1), ..., PK(En)}`.
- **4. Map Relationship Attributes:** Attributes of `R` become columns in `S`.
- **Conceptual Model First (Alternative):** Decompose n-ary into a gerund (associative) entity in ERD first, then map this entity and its binary relationships.

- **Cardinality Constraints:** Complex for n-ary; often need application logic/triggers.
- **Decomposition Consideration:** Evaluate if n-ary can be meaningfully decomposed into several binary relationships.

UNIT-III

Question 6:

a) Describe the database creation using SQL with help of suitable examples. [PYQ Q6a from problem, and in notes] (8 Marks)

- **SQL DDL for Database Creation:** Primarily uses `CREATE TABLE`.
- **`CREATE TABLE` Syntax (Simplified):** `CREATE TABLE table_name (col1_def, col2_def, ..., [table_constraints]);`
- **Key Components:**

Component	Description	Examples
`table_name`	Name of the table.	`Employees`, `Products`
Column Definition	`column_name data_type [column_constraints]`	`EmpID INT PRIMARY KEY`, `ProductName VARCHAR(100) NOT NULL`
`data_type`	Specifies data type.	`INTEGER`, `VARCHAR(n)`, `DATE`, `DECIMAL(p,s)`
Column Constraints	Apply to individual columns.	`NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `CHECK (Salary > 0)`, `DEFAULT 'Active'`, `REFERENCES Departments(DeptID)`
Table Constraints	Apply to one or more columns.	`PRIMARY KEY (OrderID, ItemID)`, `UNIQUE (Email)`, `FOREIGN KEY (DeptID) REFERENCES Departments(DeptID) ON DELETE SET NULL`, `CHECK (StartDate < EndDate)`
Referential Actions	Part of `FOREIGN KEY`.	`ON DELETE CASCADE`, `ON UPDATE RESTRICT`

Question 7:

a) Describe the cursor and type of cursor. What is the need of cursor in database programming? [PYQ Q7a from problem, and in notes] (8 Marks)

- **Cursor:** A database object for row-by-row processing of a query's result set within an application.
- **Need (Impedance Mismatch):** Bridges set-oriented SQL and row-oriented host languages.
- **Operations:** `DECLARE` (defines), `OPEN` (executes query), `FETCH` (retrieves row), `UPDATE WHERE CURRENT OF`, `DELETE WHERE CURRENT OF`, `CLOSE` (releases).
- **Types of Cursors:**

Cursor Type	Description	Key Syntax/Behavior
Read-Only	Allows only fetching data.	Default or `FOR READ ONLY`.
Updatable	Allows fetching, modifying, deleting rows.	`FOR UPDATE [OF columns]`. Query must be simple.
Forward-Only	Default. Rows fetched sequentially (first to last).	-
Scrollable	Allows flexible movement (`NEXT`, `PRIOR`, `FIRST`, `LAST`, `ABSOLUTE n`, `RELATIVE n`).	`SCROLL CURSOR`.
Insensitive	Operates on a snapshot of data at open time; changes by others not visible.	`INSENSITIVE CURSOR`.
Sensitive	Attempts to reflect underlying data changes made after cursor open.	Behavior varies by DBMS.
Keyset-Driven	Keys fixed at open; non-key changes visible. Deleted rows are "holes"; new qualifying rows usually not seen.	A type of sensitive cursor.

---

b) What is database trigger? Explain the types of Trigger. [PYQ Q7b from problem, and in notes] (7 Marks)

- **Database Trigger:** Procedural code automatically executed by DBMS on DML events (`INSERT`, `UPDATE`, `DELETE`) or DDL events on a specified table/view.
- **ECA Model:** Event (DML/DDL op), Condition (optional Boolean), Action (SQL block/procedure).
- **Types of Triggers:**

Trigger Type	Description	Key Characteristics
Row-Level	Fires once for each affected row.	`FOR EACH ROW`. Can access `:OLD` and `:NEW` row values.
Statement-Level	Default. Fires once per DML statement.	-
Timing (BEFORE/AFTER)		
`BEFORE`	Action executes *before* the DML event.	For validation, modification of new data.
`AFTER`	Action executes *after* the DML event.	For auditing, complex integrity, propagating changes.
`INSTEAD OF`	Used with views (esp. non-updatable). Fires *instead* of DML on view; trigger code acts on base tables.	Enables DML operations on complex views.
DDL Triggers	Fire on DDL events (`CREATE TABLE`, etc.).	DBMS-specific. For auditing schema changes,

Trigger Type	Description	Key Characteristics
		enforcing conventions.
Logon/Logoff	System-Level. Fire on user session connect/disconnect.	DBMS-specific. For auditing sessions.
Database Event	Server-Level. Fire on DB events (startup, shutdown, errors).	DBMS-specific. For admin tasks.
Compound (Oracle)	Defines actions for multiple timing points (BEFORE STMT, BEFORE ROW, AFTER ROW, AFTER STMT) for one DML event.	Simplifies logic and variable sharing.

UNIT-IV

Question 8:

a) Explain the Indexing and its methods with the help of suitable examples. [PYQ Q8a from problem, and in notes] (8 Marks)

- **Indexing:** DB performance optimization technique creating separate data structure (index key + pointers) for faster data retrieval by avoiding full table scans.
- **Methods/Types of Indexing:**

Index Type	Description	Example
Primary	Index on ordering key; data file physically ordered by this key. At most one.	Index on `EmployeeID` if table is sorted by `EmployeeID`.
Clustered	Physical order of data rows matches index key order. At most one per table.	`CREATE CLUSTERED INDEX IX_Orders_Date ON Orders(OrderDate);`
Secondary	Non-Clustered. Index order differs from physical row order. Multiple per table.	`CREATE INDEX IX_Emp_LName ON Employees(LastName);`
B+-Tree	Most common. Balanced tree; leaf nodes linked for efficient range/equality searches.	Default for most `CREATE INDEX` statements.
Hash	Uses hash function for direct address. Fast for exact equality; not for range.	Index on `CustomerEmail` for `WHERE Email = '...'`.
Unique	Enforces uniqueness on indexed column(s). PKs are always unique.	`CREATE UNIQUE INDEX UQ_Emp_Email ON Employees(Email);`
Composite	Multi-column. Index on >=2 columns. Column order matters.	`CREATE INDEX IX_Orders_CustDate ON Orders(CustomerID, OrderDate);`
Covering	Non-clustered. Contains all columns for a query, allowing index-only scan.	Index on `(CustID, OrderDate, Amount)` for `SELECT OrderDate, Amount WHERE CustID=...`

b) Discuss about the database security used in the database modelling. [PYQ Q8b from problem, and in notes] (7 Marks)

- **Integrating Security in Database Modeling:** Security considerations throughout all design phases.
- **Phases & Considerations:**

Modeling Phase	Security Considerations
1. Conceptual	Identify Sensitive Data. Define Access Privileges Conceptually (User Roles & permissions). Consider Views for Abstraction. Clarify Data Ownership.
2. Logical	Design Views for column-level & row-level security. Plan Granular SQL Privileges (`SELECT`, `INSERT`). Design for Separation of Duties.
3. Physical (Informed by Modeling)	Implement using `GRANT`/`REVOKE` (DAC), Role-Based Access Control (RBAC), Stored Procedures, Triggers for auditing, Encryption.

Question 9:

a) Clustering [PYQ Q9a from problem, and in notes]

- **Definition:** Physically storing related data records close together on disk (same/adjacent blocks) to minimize Disk I/O for related accesses.
- **How it Works:** Organizing rows by values of a clustering key.
- **Types:**
- **Intra-Table Clustering (Clustered Index):** Physical row order in a table matches its clustered index order. (One per table).
- **Inter-Table Clustering (Co-clustering):** Physically interleaving rows from >=2 related tables based on join key (PK-FK).
- **Benefits:** Reduced Disk I/O, faster joins (co-clustering), efficient range queries (clustered index).
- **Drawbacks:** DML overhead, only one clustered index/table.

---

b) De-normalization [PYQ from problem, and in notes]

- **Definition:** Intentionally adding controlled redundancy to a normalized schema to improve read performance (query speed) for specific critical queries by reducing joins or pre-calculating values.
- **Rationale:** To optimize performance when normalization leads to too many joins for critical queries.
- **Techniques:**
- **Pre-joining Tables:** Adding attributes from "one-side" to "many-side" table.
- **Storing Derived/Calculated Values:** E.g., storing `OrderTotal` instead of summing items.
- **Combining Tables:** Merging tables with 1:1 or tight 1:N relationships.
- **Trade-offs:** Faster reads vs. increased storage, inconsistency risk, complex DML.

c) Database Tuning [PYQ Q9c from problem, and in notes]

- **Definition:** Systematic process of optimizing database system performance. Iterative: Monitor -> Identify Bottlenecks -> Diagnose -> Implement Changes -> Measure.
- **Areas of Tuning:**

Tuning Area	Key Activities
Conceptual Schema	Normalization/Denormalization choices, partitioning (vertical/horizontal).
Queries & Views	Rewriting SQL, optimizing view definitions, influencing query optimizer (hints, statistics).
Physical Design	Index selection/management, clustering, file organization.
Application Code	Optimizing DB interaction (batching, connection pooling).
DBMS Parameters	Adjusting configuration (memory, I/O, concurrency).
Hardware/OS	Ensuring sufficient resources, OS optimization.

- **Workload Analysis:** Critical first step: understand query/update types, frequencies, goals, data access patterns.