

Database Management (4331603) - Winter 2024 Solution

Milav Dabgar

December 7, 2024

Question 1(a) [3 marks]

Explain three-level database architecture.

Solution

Table:

Table 1. Database Architecture Levels

Level	Description	Purpose
External Level	User views and application programs	Data abstraction for users
Conceptual Level	Complete logical structure	Organization-wide data view
Internal Level	Physical storage details	Storage and access methods

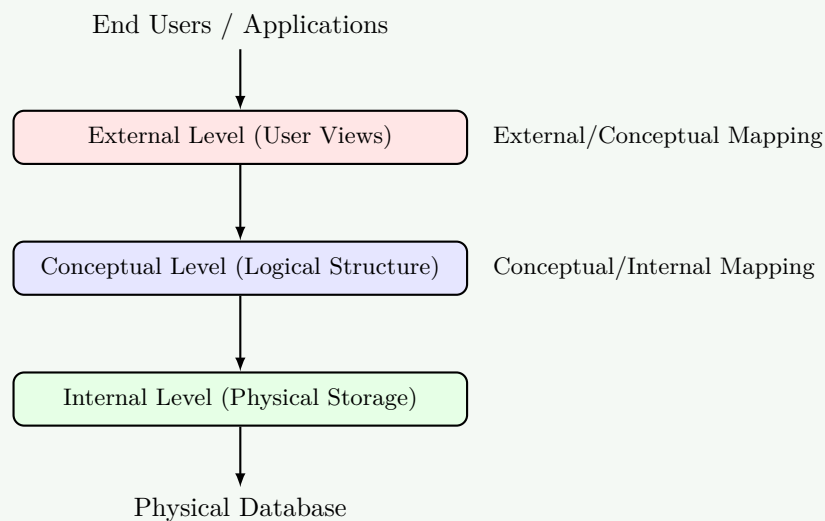


Figure 1. Three-Level Architecture

- **External Level:** Individual user views and specific application requirements
- **Conceptual Level:** Complete database schema without storage details
- **Internal Level:** Physical storage structures and access paths

Mnemonic

“ECI - Every Computer Interface”

Question 1(b) [4 marks]

Explain Total Participation and Partial Participation with example.

Solution

Table:

Table 2. Participation Types

Participation Type	Definition	Symbol	Example
Total Participation	Every entity must participate	Double line	Student-Course enrollment
Partial Participation	Some entities may not participate	Single line	Employee-Department management

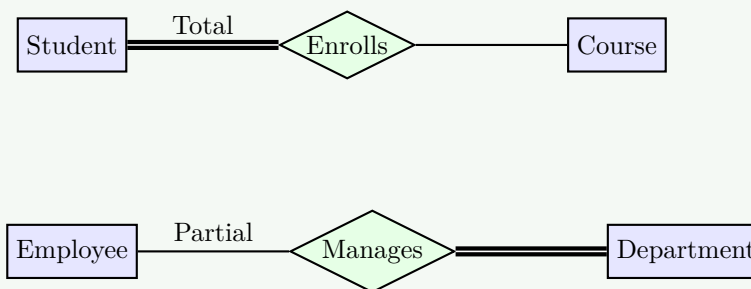


Figure 2. Total vs Partial Participation

- **Total Participation:** All students must be enrolled in at least one course
- **Partial Participation:** Not all employees manage a department
- **Double lines** indicate total participation constraints
- **Single lines** show partial participation relationships

Mnemonic

“Total = Two lines, Partial = Plain line”

Question 1(c) [7 marks]

Explain advantages of DBMS over file management systems.

Solution

Table:

Table 3. Advantages of DBMS

Advantage	File System	DBMS
Data Redundancy	High duplication	Controlled redundancy
Data Inconsistency	Common problem	Data integrity maintained
Data Sharing	Limited sharing	Concurrent access support
Security	File-level security	User-level access control
Backup & Recovery	Manual process	Automatic mechanisms

- **Reduced Data Redundancy:** Eliminates duplicate data storage across applications
- **Data Consistency:** Ensures uniform data across all applications

- **Data Independence:** Applications independent of data structure changes
- **Concurrent Access:** Multiple users can access data simultaneously
- **Security Control:** User authentication and authorization mechanisms
- **Backup and Recovery:** Automatic data protection and restoration
- **Data Integrity:** Constraint enforcement maintains data quality

Mnemonic

“RDCCSBI - Really Don’t Copy, Control, Secure, Backup, Integrate”

Question 1(c OR) [7 marks]

List out various data models. Explain any two in brief.

Solution

Data Models List:

- Hierarchical Data Model
- Network Data Model
- Relational Data Model
- Object-Oriented Data Model
- Entity-Relationship Model

Table:

Table 4. Relational vs Network Model

Model	Structure	Advantages	Disadvantages
Relational Model	Tables with rows/columns	Simple, flexible	Performance overhead
Network Model	Graph with records/links	Efficient navigation	Complex structure

Relational Data Model:

- **Structure:** Data organized in tables (relations)
- **Components:** Tuples (rows), attributes (columns), domains
- **Operations:** Select, project, join operations available

Network Data Model:

- **Structure:** Graph-based with owner-member relationships
- **Navigation:** Explicit links between record types
- **Flexibility:** Many-to-many relationships supported naturally

Mnemonic

“HNROE - Have Network Relational Object Entity”

Question 2(a) [3 marks]

Explain Mapping Cardinalities.

Solution

Table:

Table 5. Cardinality Types

Cardinality	Symbol	Description	Example
One-to-One	1:1	Each entity relates to one other	Person-Passport
One-to-Many	1:M	One entity relates to many	Department-Employee
Many-to-One	M:1	Many entities relate to one	Student-Course
Many-to-Many	M:N	Many relate to many	Student-Subject

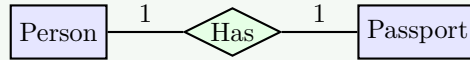


Figure 3. Cardinality Examples

- **Cardinality constraints:** define relationship participation limits
- **Maximum cardinality:** specifies upper bound of associations
- **Importance:** Helps in database design and relationship modeling

Mnemonic

“OMOM - One, One-Many, One-Many, Many-Many”

Question 2(b) [4 marks]

Explain Outer Join operation in Relational Algebra.

Solution

Table:

Table 6. Outer Join Types

Join Type	Symbol	Result	NULL Handling
Left Outer Join	\bowtie_L	All left + matching right	NULLs for unmatched right
Right Outer Join	\bowtie_R	All right + matching left	NULLs for unmatched left
Full Outer Join	\bowtie_F	All from both tables	NULLs for unmatched

Example:

- EMPLOYEE LEFT OUTER JOIN DEPARTMENT
- Includes all employees
- NULL values for employees without departments
- **Preserves tuples:** Keeps unmatched tuples from specified relation(s)
- **NULL values:** Fills missing attribute values
- **Usage:** Useful for reporting incomplete data relationships

Mnemonic

“LRF - Left Right Full outer joins”

Question 2(c) [7 marks]

Explain concept of Specialization and Generalization with example.

Solution

Table:

Table 7. Specialization vs Generalization

Concept	Direction	Process	Example
Specialization	Top-Down	General to Specific	Vehicle → Car, Truck
Generalization	Bottom-Up	Specific to General	Car, Truck → Vehicle

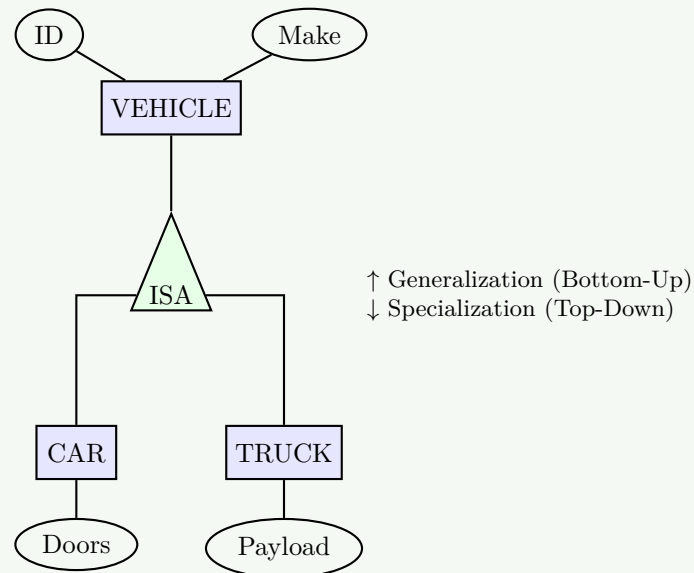


Figure 4. ISA Hierarchy Example

Specialization:

- **Process:** Creating subclasses from superclass (Top-Down)
- **Inheritance:** Subclasses inherit all superclass attributes
- **Extension:** Subclasses have specific additional attributes

Generalization:

- **Process:** Creating superclass from common subclass features (Bottom-Up)
- **Abstraction:** Identifies common attributes and relationships
- **Simplification:** Reduces complexity through hierarchy

Mnemonic

“SG-TD-BU - Specialization General-To-Detail, Bottom-Up”

Question 2(a OR) [3 marks]

Explain different types of Keys in Relational Algebra.

Solution

Table:

Table 8. Types of Keys

Key Type	Definition	Uniqueness	Example
Super Key	Any attribute set that uniquely identifies	Yes	{ID, Name, Phone}
Candidate Key	Minimal super key	Yes	{ID}, {Email}
Primary Key	Chosen candidate key	Yes	{StudentID}
Foreign Key	References primary key	No	{DeptID}

- **Super Key:** Uniquely identifies tuples, may have extra attributes
- **Candidate Key:** Minimal super key without redundant attributes
- **Primary Key:** Selected candidate key for entity identification
- **Foreign Key:** Establishes referential integrity between tables

Mnemonic

“SCPF - Super Candidate Primary Foreign”

Question 2(b OR) [4 marks]

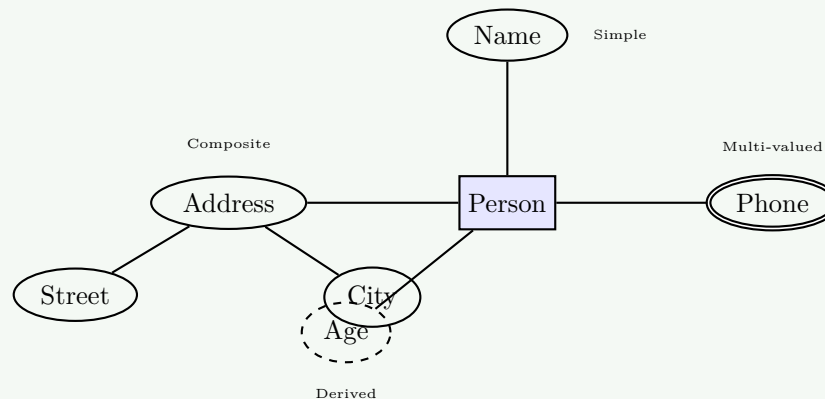
Explain types of attributes in ER-diagram with suitable example.

Solution

Table:

Table 9. Attribute Types

Attribute Type	Symbol	Description	Example
Simple	Oval	Cannot be subdivided	Age, Name
Composite	Tree-like	Can be subdivided	Address (Street, City)
Derived	Dashed oval	Calculated from others	Age from Birth_Date
Multi-valued	Double oval	Multiple values	Phone_Numbers

**Figure 5.** Attribute Types in ER

- **Simple attributes:** atomic and indivisible
- **Composite attributes:** have meaningful sub-parts (e.g., Address broken into Street, City)
- **Derived attributes:** computed from other attribute values (e.g., Age from DOB)
- **Multi-valued attributes:** store multiple values per entity (e.g., Phone numbers)

Mnemonic

“SCDM - Simple Composite Derived Multi-valued”

Question 2(c OR) [7 marks]

Explain SELECT, PROJECT, UNION and SET-INTERSECTION operation with suitable example.

Solution

Table:

Table 10. Relational Operations

Operation	Symbol	Purpose	Example
SELECT	σ	Filter rows	$\sigma_{salary > 50000}(Employee)$
PROJECT	π	Select columns	$\pi_{name, age}(Employee)$
UNION	\cup	Combine relations	$R \cup S$
INTERSECTION	\cap	Common tuples	$R \cap S$

1. SELECT Operation (σ):

- Filters rows based on condition.
- Example: $\sigma_{age > 25}(STUDENT)$
- Returns students older than 25 years.

2. PROJECT Operation (π):

- Selects specific columns.
- Example: $\pi_{name, course}(STUDENT)$
- Returns only name and course columns.

3. UNION Operation (\cup):

- Combines tuples from two relations (removing duplicates).
- Example: $SCIENCE_STUDENTS \cup ARTS_STUDENTS$
- Combines students from both streams.

4. INTERSECTION Operation (\cap):

- Returns tuples present in BOTH relations.
- Example: $MALE_STUDENTS \cap SPORTS_STUDENTS$
- Returns male students who play sports.

Mnemonic

“SPUI - Select Project Union Intersection”

Question 3(a) [3 marks]

Differentiate Primary Key and Foreign Key constraint.

Solution

Table:

Table 11. Primary Key vs Foreign Key

Aspect	Primary Key	Foreign Key
Purpose	Unique identification	Referential integrity
NULL Values	Not allowed	Allowed
Uniqueness	Must be unique	Can be duplicate
Count	Only one per table	Multiple allowed

- **Primary Key:** Ensures entity integrity within table (Unique + Not Null).
- **Foreign Key:** Maintains referential integrity between tables (links to PK).
- **Uniqueness:** Primary keys are always unique; foreign keys can repeat.

Mnemonic

“PU-FN - Primary Unique, Foreign Nullable”

Question 3(b) [4 marks]

Explain DUAL table and SYSDATE with example.

Solution

Table:

Table 12. DUAL and SYSDATE

Component	Type	Purpose	Example
DUAL	Virtual table	Test expressions	SELECT 2+3 FROM DUAL
SYSDATE	System function	Current date/time	SELECT SYSDATE FROM DUAL

DUAL Table:

- **Virtual table:** Special one-row, one-column table present in Oracle.
- **Testing:** Used to select system functions or calculations not linked to a physical table.

SYSDATE Function:

- **System Time:** Returns current date and time from the database server.
- **Operations:** Supports date arithmetic (e.g., SYSDATE + 1 for tomorrow).

Examples:

- SELECT SYSDATE FROM DUAL; (Current Time)
- SELECT SYSDATE + 30 FROM DUAL; (30 days later)

Mnemonic

“DT-ST - DUAL Testing, SYSDATE Time”

Question 3(c) [7 marks]

Write SQL queries to use various numeric functions.

Solution

Numeric Functions Overview:

- **TRUNC:** Truncates number to specified precision
- **ABS:** Absolute value
- **CEIL:** Smallest integer $\geq n$
- **FLOOR:** Largest integer $\leq n$

- MOD: Remainder
- POWER: Exponentiation

SQL Queries:

- (a) Display integer value of 125.25
SELECT TRUNC(125.25) FROM DUAL; → 125
- (b) Display absolute value of (-15)
SELECT ABS(-15) FROM DUAL; → 15
- (c) Display ceil value of 55.65
SELECT CEIL(55.65) FROM DUAL; → 56
- (d) Display floor value of 100.2
SELECT FLOOR(100.2) FROM DUAL; → 100
- (e) Display the square root of 16
SELECT SQRT(16) FROM DUAL; → 4
- (f) Show value of e^3
SELECT EXP(3) FROM DUAL;
- (g) Display result of 12 raised to 6
SELECT POWER(12, 6) FROM DUAL;
- (h) Display result of 24 mod 2
SELECT MOD(24, 2) FROM DUAL; → 0
- (i) Show output of sign(-25), sign(25), sign(0)
SELECT SIGN(-25), SIGN(25), SIGN(0) FROM DUAL; → -1, 1, 0

Mnemonic

“TACFSEPM - TRUNC ABS CEIL FLOOR SQRT EXP POWER MOD”

Question 3(a OR) [3 marks]

Explain Unique and Check Constraint with suitable example.

Solution

Table:

Table 13. Relationship Constraints

Constraint	Purpose	Duplicates	Example
UNIQUE	Prevent duplicates	Not allowed	Email address
CHECK	Validate data	Value restrictions	Age > 0

Examples:

- **UNIQUE Constraint:**
email VARCHAR(50) UNIQUE ensures no two students have the same email.
- **CHECK Constraint:**
age NUMBER CHECK (age >= 18) ensures age is valid.

Mnemonic

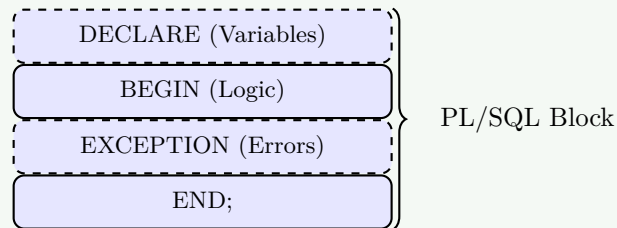
“UC-DV - Unique no Copy, Check Validates”

Question 3(b OR) [4 marks]

Explain structure of PL/SQL block.

Solution**Table:****Table 14.** PL/SQL Block Sections

Section	Required	Purpose	Example
DECLARE	Optional	Variable declarations	v_name VARCHAR2(20);
BEGIN	Mandatory	Executable statements	SELECT ... INTO ...
EXCEPTION	Optional	Error handling	WHEN OTHERS THEN ...
END	Mandatory	Block termination	END;

**Figure 6.** PL/SQL Block Structure

- **DECLARE section:** Variable and cursor declarations (Optional).
- **BEGIN-END:** Mandatory executable section containing SQL and logic.
- **EXCEPTION section:** Handles runtime errors (Optional).

Mnemonic

“DBE-E - Declare Begin Exception End”

Question 3(c OR) [7 marks]

Consider the following table and solve queries:

Solution**I) Create the BRANCH table:**

```

1 CREATE TABLE BRANCH (
2     branchid VARCHAR2(10) PRIMARY KEY,
3     branchname VARCHAR2(50) NOT NULL,
4     address VARCHAR2(100)
5 );
  
```

II) Create the EMPLOYEE table:

```

1 CREATE TABLE EMPLOYEE (
2     empid VARCHAR2(10) PRIMARY KEY,
3     name VARCHAR2(50) NOT NULL,
4     post VARCHAR2(30),
5     gender CHAR(1) CHECK (gender IN ('M', 'F')),
6     birthdate DATE,
7     salary NUMBER(10,2),
8     branchid VARCHAR2(10),
9     FOREIGN KEY (branchid) REFERENCES BRANCH(branchid)
10 );
  
```

III) Find employees in Ahmedabad branch:

```

1 SELECT e.* FROM EMPLOYEE e, BRANCH b
2 WHERE e.branchid = b.branchid
3 AND b.branchname = 'Ahmedabad';

```

IV) Find employees born in 1998:

```

1 SELECT * FROM EMPLOYEE
2 WHERE EXTRACT(YEAR FROM birthdate) = 1998;

```

V) Find female employees with salary > 5000:

```

1 SELECT * FROM EMPLOYEE
2 WHERE gender = 'F' AND salary > 5000;

```

VI) Find address where Ajay works:

```

1 SELECT b.address FROM EMPLOYEE e, BRANCH b
2 WHERE e.branchid = b.branchid
3 AND e.name = 'Ajay';

```

Mnemonic

“CBEFFA - Create Branch Employee Find Female Address”

Question 4(a) [3 marks]

Explain Referential Integrity with suitable example.

Solution

Table:

Table 15. Referential Integrity

Aspect	Description	Example
Definition	Foreign key must reference existing primary key	Employee.deptid → Department.deptid
Purpose	Maintain data consistency	Prevent orphan records
Actions	CASCADE, SET NULL, RESTRICT	ON DELETE CASCADE

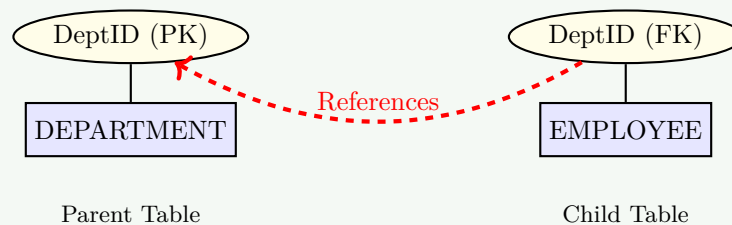


Figure 7. Foreign Key Reference

- **Referential integrity:** ensures valid relationships between tables.
- **Orphan records:** prevented; cannot add employee to non-existent department.
- **Cascade:** operations automatically propagate updates/deletes.

Mnemonic

“RIO - Referential Integrity prevents Orphans”

Question 4(b) [4 marks]

Differentiate Partial and Full Functional Dependency.

Solution

Table:

Table 16. Dependency Types

Type	Definition	Example	Requirement
Partial	Depends on part of composite key	$(\text{StudentID}, \text{CourseID}) \rightarrow \text{Name}$	Composite primary key
Full	Depends on entire key	$(\text{StudentID}, \text{CourseID}) \rightarrow \text{Grade}$	Complete key needed

Example: Key is (StudentID, CourseID)

- **Partial:** StudentName depends only on StudentID. Violates 2NF.
- **Full:** Grade depends on BOTH StudentID and CourseID. Good.
- **Partial dependency** causes data redundancy and anomalies.
- **2NF eliminates** partial functional dependencies.

Mnemonic

“PF-CF - Partial Few, Complete Full”

Question 4(c) [7 marks]

Explain 3rd Normal Form with example.

Solution

3rd Normal Form Requirements:

- Must be in 2NF.
- No transitive dependencies (Non-key \rightarrow Non-key).
- Non-key attributes depend **ONLY** on the primary key.

Problem (Before 3NF):

StudID	Name	CourseID	CName	InstID	InstName
S1	John	C1	Math	I1	Dr. Smith

Issue: $\text{StudentID} \rightarrow \text{InstID} \rightarrow \text{InstName}$ (Transitive Dependency)

3NF Solution (Decomposition):

- STUDENT Table:**

StudID	Name	CourseID
S1	John	C1
- COURSE Table:**

CourseID	CName	InstID
C1	Math	I1
- INSTRUCTOR Table:**

InstID	InstName
I1	Dr. Smith

Mnemonic

“3NF-NT - 3rd Normal Form No Transitives”

Question 4(a OR) [3 marks]

Explain Importance of Normalization.

Solution

Table:

Table 17. Benefits of Normalization

Benefit	Problem Solved	Result
Reduce Redundancy	Duplicate data	Storage efficiency
Eliminate Anomalies	Update/Insert/Delete issues	Data consistency
Improve Integrity	Data inconsistency	Reliable information

- **Minimizes redundancy:** Saves storage by storing data once.
- **Prevents anomalies:** Ensures logical data modification.
- **Simplifies maintenance:** Organized structure is easier to manage.

Mnemonic

“RESIM - Redundancy Eliminated, Storage Improved, Maintenance”

Question 4(b OR) [4 marks]

Differentiate Prime Attributes and Non-Prime Attributes.

Solution

Table:

Table 18. Prime vs Non-Prime Attributes

Type	Definition	Role	Example
Prime	Part of candidate key	Key formation	StudentID, CourseID
Non-Prime	Not part of any key	Data storage	Name, Grade

Example: Relation(StudentID, CourseID, Grade, Semester)

- **Candidate Key:** (StudentID, CourseID)
- **Prime Attributes:** StudentID, CourseID
- **Non-Prime Attributes:** Grade, Semester
- **Significance:** Functional dependency involving non-prime attributes determines Normal Form (2NF/3NF).

Mnemonic

“PN-KD - Prime in Key, Non-prime for Data”

Question 4(c OR) [7 marks]

Explain 2nd Normal Form with example.

Solution**2nd Normal Form Requirements:**

- Must be in 1NF.
- No partial functional dependencies.
- Non-key attributes must depend on the WHOLE primary key.

Problem (Before 2NF):

StudID	CourseID	SName	CName	Grade
S1	C1	John	Math	A
S1	C2	John	Physics	B

Issue: $StudID \rightarrow SName$ (Partial), $CourseID \rightarrow CName$ (Partial)

2NF Solution (Decomposition):

1. STUDENT Table:

StudID	SName
S1	John

2. COURSE Table:

CourseID	CName
C1	Math

3. ENROLLMENT Table:

StudID	CourseID	Grade
S1	C1	A
S1	C2	B

Mnemonic

“2NF-FD - 2nd Normal Form Full Dependencies”

Question 5(a) [3 marks]

Explain Transaction states with proper diagram.

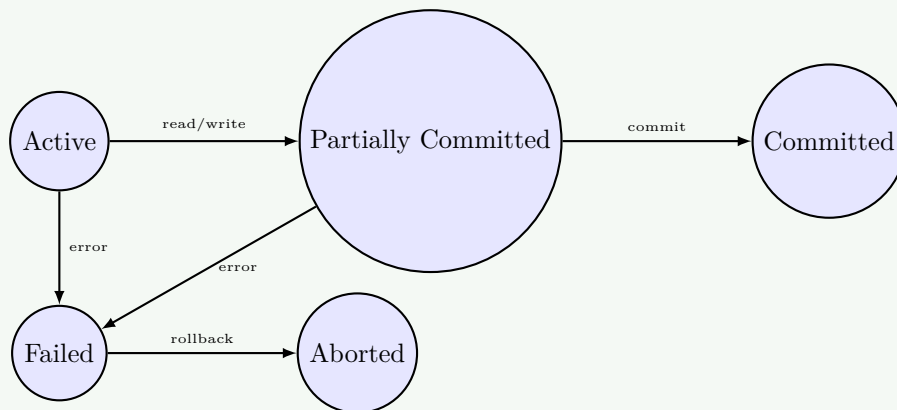
Solution

Figure 8. Transaction State Diagram

Table:

Table 19. Transaction States

State	Description	Next State
Active	Executing operations	Partial/Failed
Partially Committed	Last statement executed	Committed/Failed
Committed	Successful completion	End
Failed	Execution stopped due to error	Aborted
Aborted	Rolling back changes	End

Mnemonic

“APCFA - Active Partial Commit Fail Abort”

Question 5(b) [4 marks]

Explain any two DDL commands with a suitable example.

Solution

Table:

Table 20. DDL Commands

Command	Purpose	Example Syntax
CREATE	Create objects	CREATE TABLE T (...)
ALTER	Modify structure	ALTER TABLE T ADD ...

1. CREATE Command:

```

1 CREATE TABLE EMPLOYEE (
2     empid NUMBER(5) PRIMARY KEY,
3     name VARCHAR2(50)
4 );

```

2. ALTER Command:

```

1 -- Add new column
2 ALTER TABLE EMPLOYEE ADD phone VARCHAR2(15);
3 -- Modify column
4 ALTER TABLE EMPLOYEE MODIFY name VARCHAR2(100);
5 -- Drop column
6 ALTER TABLE EMPLOYEE DROP COLUMN phone;

```

- **DDL:** Data Definition Language defines schema.
- **Auto-commit:** DDL commands are automatically saved.

Mnemonic

“CA-NM - CREATE Adds, ALTER Modifies”

Question 5(c) [7 marks]

Explain ACID Properties in detail.

Solution

Table:

Table 21. ACID Properties

Property	Definition	Example
Atomicity	All or nothing	Complete transfer or rollback
Consistency	Database always valid	Balance never negative
Isolation	Independent execution	Concurrent users don't interfere
Durability	Permanent changes	Data survives power loss

- **Atomicity:** Transaction is an indivisible unit.
- **Consistency:** Transforms DB from one valid state to another.
- **Isolation:** Intermediate states are invisible to other transactions.
- **Durability:** Committed data is saved permanently even after crash.

Mnemonic

“ACID - Atomicity Consistency Isolation Durability”

Question 5(a OR) [3 marks]

What is two phase locking technique?

Solution

Table:

Table 22. 2PL Phases

Phase	Action	Description	Allowed
Growing	Acquire locks	Transaction obtains all locks	LOCK
Shrinking	Release locks	Transaction releases locks	UNLOCK

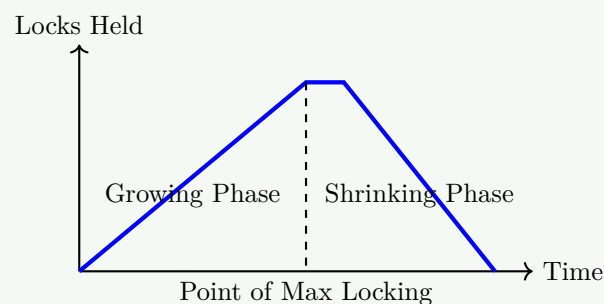


Figure 9. Two-Phase Locking Protocol

- **Growing Phase:** Transaction may obtain locks, but cannot release any.
- **Shrinking Phase:** Transaction may release locks, but cannot obtain any.
- **Guarantee:** Ensures serializability.

Mnemonic

“2PL-GS - Two Phase Locking Growing Shrinking”

Question 5(b OR) [4 marks]

Explain any two DML commands with a suitable example.

Solution

Table:

Table 23. DML Commands

Command	Purpose	Example
INSERT	Add new records	INSERT INTO Student...
UPDATE	Modify records	UPDATE Student SET...

1. INSERT Command:

```
1 INSERT INTO EMPLOYEE (empid, name, salary)
2 VALUES (101, 'John Smith', 50000);
```

2. UPDATE Command:

```
1 UPDATE EMPLOYEE
2 SET salary = 60000
3 WHERE empid = 101;
```

- **DML:** Data Manipulation Language manages data values.
- **Explicit Commit:** Changes often need COMMIT to be saved.

Mnemonic

“TU-AM - INSERT Adds, UPDATE Modifies”

Question 5(c OR) [7 marks]

List problems of concurrency control and explain any two in detail.

Solution

Problems List:

1. Lost Update Problem
2. Dirty Read Problem
3. Non-repeatable Read Problem
4. Phantom Read Problem

1. Lost Update Problem:

- **Scenario:** Two transactions update the same item, and one update overwrites the other.
- **Example:**
 - T1 reads X=100. T2 reads X=100.
 - T1 writes X=110. T2 writes X=90.
 - T1's update is lost. Final X=90 (should be 100).

2. Dirty Read Problem:

- **Scenario:** Reading data written by an uncommitted transaction.
- **Example:**
 - T1 updates X=200. T2 reads X=200.
 - T1 fails and rolls back X=100.
 - T2 has processed invalid data (200).

Solutions:

- Use **Locking Protocols** (2PL).

- Use **Isolation Levels** (Read Committed, Serializable).

Mnemonic

“LDUI - Lost Dirty Unrepeatable Inconsistent”