Solution for SPRING MID SEMESTER EXAMINATION-2025

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Database Management System
CS20006/ IT20006/ CC20006/ CM20006/ CS2004

Time: 1 1/2 Hours **Full Mark: 20**

Q1	Answer all the questions.	1X5

a.	Using Armstrong's axioms, prove that if A→B and BC→D then AC→D.
	We are given: ● A→B ● BC→D We need to prove: AC→D. Using Armstrong's axioms: 1. **Augmentation**: Since A→B, we can add C on both sides: AC→BC 2. **Transitivity**: From AC→BC and BC→D, we apply transitivity: AC→D Thus, we have proved that AC→D.
b.	We can convert weak entity sets into strong entity sets by simply adding the appropriate attributes. Then, why are weak entity sets needed?
	Although we can convert weak entity sets into strong entity sets by adding the appropriate attributes, weak entity sets are still useful because: ● Data Redundancy Reduction: Converting a weak entity set into a strong entity set can lead to data duplication if the key attributes of the identifying entity set are large. ● Logical Data Modeling: Weak entities help model real-world relationships where some entities naturally

depend on others (e.g., a "Dependent" entity in an insurance system depends on the "Employee" entity).
- Foreign Key Constraints: Weak entities inherently ensure that their existence is dependent on another entity, which cannot be enforced naturally in a strong entity model.

c. What are recursive relationships? Give one example where this is required.

A recursive relationship occurs when an entity set has a relationship with itself.

Example:
In an Employee entity set, an employee can be a manager of another employee. The "Manages" relationship is recursive since both the manager and subordinate are part of the same entity set (Employee).

d. Explain the difference between two-tier and three-tier application architectures. Which is better suited for web applications?

Feature	Two-Tier Architecture	Three-Tier Architecture
Structure	Client ↔ Database	Client ↔ Application Server ↔ Database
Scalability	Less scalable	Highly scalable
Security	Less secure	More secure (middle layer hides direct DB access)
Performance	Can be slower due to direct DB access	More efficient with load balancing
Suitability	Used for small applications	Ideal for web applications

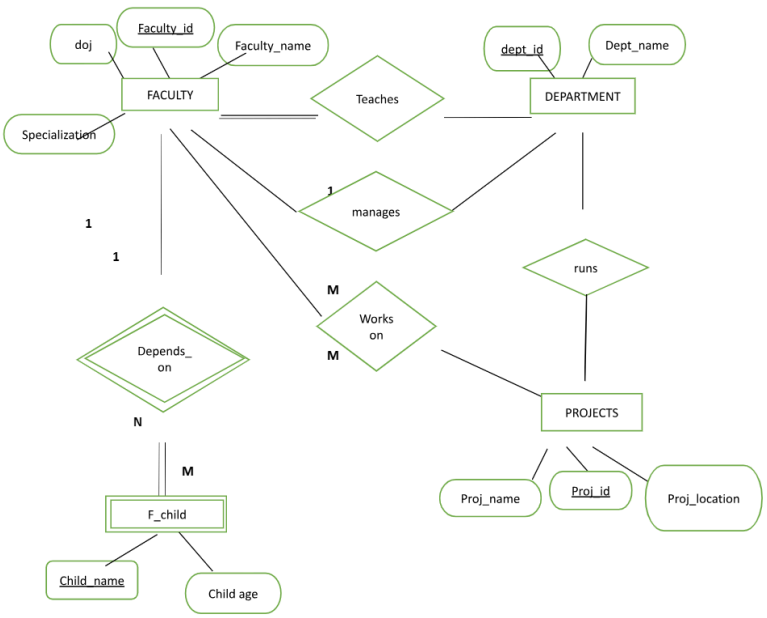
Three-tier architecture is better suited for web applications since it enhances scalability, security, and performance by introducing an intermediate application layer.

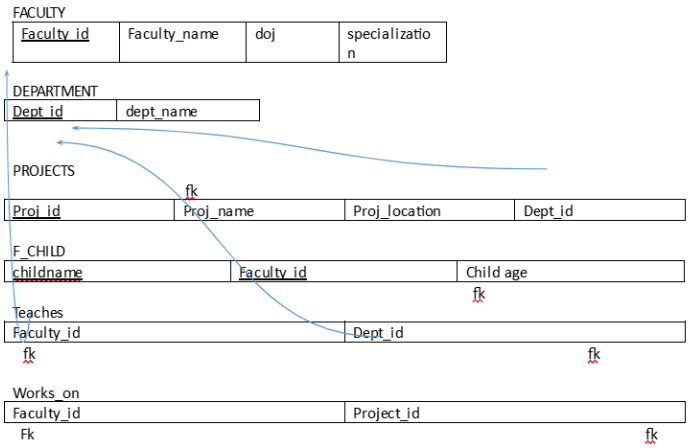
e. Consider a relation R with attributes $\{a_1, a_2, a_3, a_4, a_5\}$ and a_4 as the primary key. determine the total number of super keys for relation R.

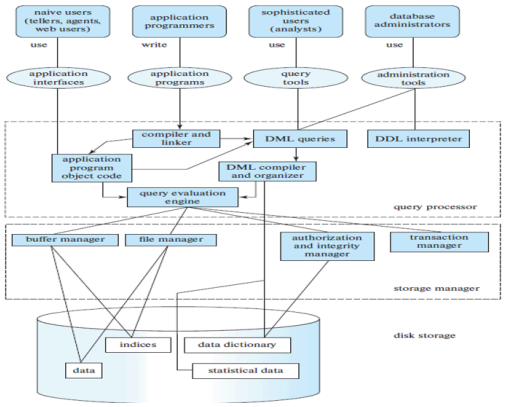
Given relation R(a1,a2,a3,a4,a5) with **primary key = {a4}**, the number of superkeys is calculated as follows:

A **superkey** is any superset of the primary key. Since the primary key is {a4}, any combination of {a4} with other attributes is also a superkey.

- There are 4 other attributes (a1, a2, a3, a5), each of which can be either included or not in the superkey.

	• The number of ways to choose any subset of {a1, a2, a3, a5} is 2^4 = 16. Thus, the total number of superkeys is **16**.	
2.	An educational institute database needs to store information about faculty members (identified by faculty-id, with faculty-name, doj, and specialization as attributes); departments (identified by dept-id, with dept-name as attributes); projects (identified by proj-id, with proj-name, proj-location as attributes) and children of faculty members (with child-name and child-age as attributes). A department can have many faculty members and faculty member can teach in more than one department. Faculty members can work on different projects. A department can have many projects and a project can belong to at most one department. Each department is managed by one HOD,who is a faculty member. A child must be identified uniquely by name when the parent (who is a faculty member; assume that only one parent works for institute) is known. We are not interested in information about a child one the parent leaves the institute. Answer the following questions i) Draw the ER diagram that captures the above information. ii) Translate the ER diagram into relations. Also identify the primary key and foreign keys. 	3+2 marks



3.	Explain the Database System Architecture and its significance of each level, support your explanation with a diagram.	5 marks
	Database System Architechture (Diagram)  	
4.	Consider the following relational database schema consisting of the four relation schemas: passenger (pid, pname, pgender, pcity) agency (aid, aname, acity) flight (fid, fdate, time, src, dest) booking (pid, aid, fid, fdate) Answer the following questions using relational algebra queries or SQL; A. Find the agency names for agencies that located in the same city as passenger with passenger id 123.	1X5 Marks

	SQL : SELECT DISTINCT a.aname FROM agency a JOIN passenger p ON a.acity = p.pcity WHERE p.pid = 123; **RA**: $\Pi_{aname}(\text{agency} \bowtie_{acity = pcity} (\sigma_{pid = 123}(\text{passenger})))$
B.	Get the details about all flights from Chennai to New Delhi.
	SQL: SELECT * FROM flight WHERE src = 'Chennai' AND dest = 'New Delhi'; **RA**: $\Pi_{fid,fdate,time,src,dest}(\sigma_{src = "Chennai" \wedge dest = "New Delhi"}(\text{flight}))$
C.	Find only the flight numbers for passenger with pid 123 for flights to Chennai before 06/11/2020.
	SQL: SELECT DISTINCT f.fid FROM flight f JOIN booking b ON f.fid = b.fid AND f.fdate = b.fdate WHERE b.pid = 123 AND f.dest = 'Chennai' AND f.fdate < '2020-11-06'; **RA**: $\Pi_{fid}(\sigma_{pid = 123}(\text{booking}) \bowtie \sigma_{dest = "Chennai" \wedge fdate < 06/11/2020}(\text{flight}))$
D.	Find the passenger names for passengers who have bookings on at least one flight.
	SQL : SELECT DISTINCT p.pname FROM passenger p JOIN booking b ON p.pid = b.pid; **RA**: Π pname (passenger $\bowtie$ booking)
E.	Find the passenger names for those who do not have any bookings in any flights.
	SQL: SELECT p.pname FROM passenger p LEFT JOIN booking b ON p.pid = b.pid WHERE b.pid IS NULL;

	RA: Π pname ((Π pid (passenger) - Π pid (booking)) ⋈ passenger)	
5.	consider two entity sets A and B that both have the attribute X (among others whose names are not relevant to this question). A. If the two X's are completely unrelated. How should the design be improved? B. If the two X's represent the same property and it is one that applies both to A and to B. How should the design be improved? consider three subcases. a. X is the primary key for A but not B b. X is the primary key for both A and B c. X is not the primary key for A nor for B	2+3 Marks
	a. Rename the attribute X to something that describes more info in both entity sets, so that it can have a different name in both the entity sets. b. Case 1: X is the primary key for A but not B In this case we would create a foreign-key constraint on the attribute X of the entity set B referencing the entity set A. Case 2: X is the primary key for both A and B In this case we merge the two entity sets A and B into one entity set C. The new entity set C will have the attribute X as a primary key and moreover it will have all the other attributes of A and B (making proper attribute name modifications, if needed). Case 3: X is not the primary key for A nor for B In this case we can create a new entity set called C having a single attribute X. And the attribute X in A and B will be a foreign-key referencing the attribute X in the entity set C. This also gives room for storing more information about the attribute X in side of the entity set C.	