

DBMS IN 100 MINUTES

Complete Placement
Revision

FREE NOTES

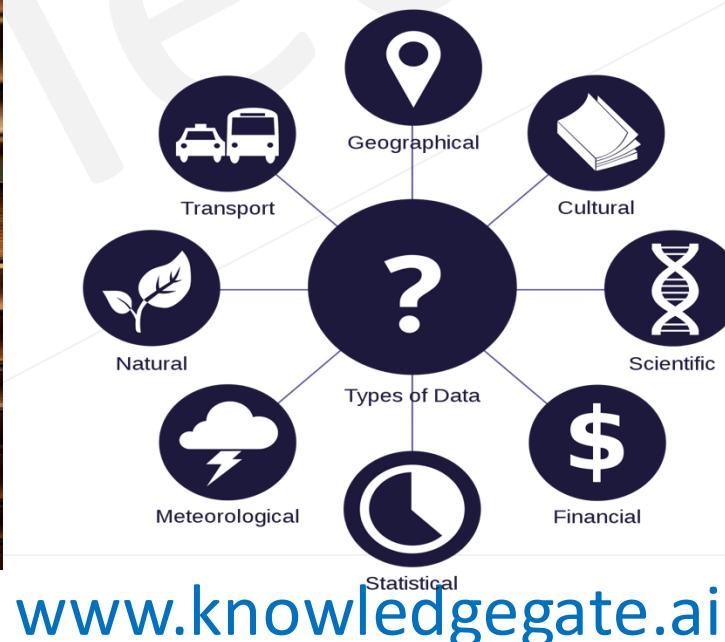


Database

- **Introduction:** “Database Management Systems (DBMS) is a highly scoring and crucial subject for campus placements, particularly with top recruiters like TCS, Infosys, Wipro, Cognizant, and Accenture. After Data Structures, DBMS requires relatively less preparation time, featuring straightforward numerical and objective questions. With growing opportunities in Data Science, AI, and Data Analytics, mastering database fundamentals is essential for a successful IT career.”
- **Who Should Take This Course:** This database course is specially designed by **Sanchit Sir**, who has **more than 10 years of experience** in the computer science field. It is suitable for all **B.Tech & other students**, including those from **CS and non-CS branches**. Sanchit Sir teaches each topic from the basics, so **no previous knowledge is needed**. He makes sure to cover **every important topic** required for placements and avoids teaching topics that aren’t necessary. As a result, the course is **efficient, complete, sufficient, and focused**, helping you save time while preparing effectively for your placement exams.

What is Data

- Data are facts and figures collected through observation. Technically, data is a set of values (qualitative or quantitative) related to people, objects, or events. A single value from data is called **datum**. Any facts or details about something or someone are called **data**.
- **Example:**
 - Student name, roll number, and exam marks (Rahul, 101, 85 marks).
 - Daily temperatures recorded (22°C, 24°C, 25°C).



Q Data typically refers to raw facts and figures that have not yet been processed to reveal any meaning. Which of the following accurately represents data? **(TCS)**

- (A)** Annual sales analysis report
- (B)** Names and addresses collected from customers
- (C)** Decisions made after sales analysis
- (D)** Strategies formulated based on customer feedback

What is Information

- **Information** is the processed form of data that is meaningful and useful for making decisions. When raw data is organized, analyzed, or structured, it becomes information. Processed data is called **information**.
- **Example:**
 - Raw data: Marks obtained by students in a class test (e.g., 78, 56, 89, 90).
 - Information: The average marks of the class or the list of students who passed or failed the test.



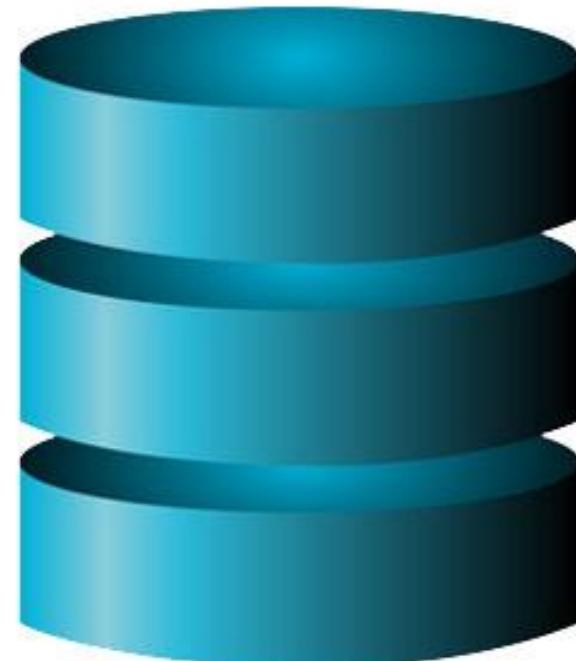
Q Information is created from data through a specific process. Which of the following processes converts data into information? **(Infosys)**

- (A) Data deletion
- (B) Data encryption
- (C) Data processing
- (D) Data storage

Answer: C, **Explanation:** Processing [transforms raw data into meaningful information.](http://www.knowledgigate.ai)

What is Data Base?

- A **database** is an organized collection of related data stored electronically. It allows data to be easily accessed, managed, and updated.
- **Example:** A college database stores student names, roll numbers, courses, and grades in a structured way.



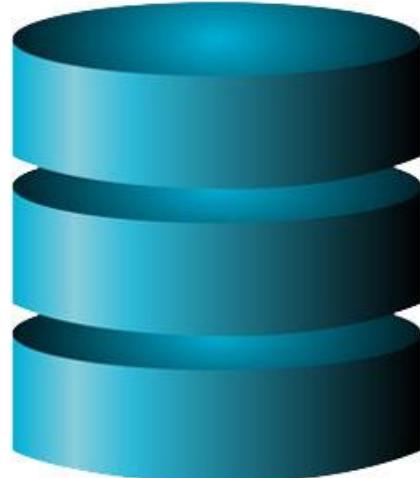
Q A database is a systematically organized collection of data. Which of the following best defines a database? **(Wipro)**

- (A)** An unstructured file system
- (B)** A collection of tables without relationships
- (C)** Interrelated data organized for easy retrieval and management
- (D)** A single spreadsheet used for analysis

Answer: C, **Explanation:** Databases store structured, interrelated data for efficient access.

What is Data Base Management System?

- A **Database Management System (DBMS)** is software that helps users create, maintain, and manage databases efficiently. It provides functionalities like data storage, retrieval, security, and concurrency control.
- **Example:** MySQL, Oracle, and Microsoft SQL Server are popular DBMS software used by companies to handle their data securely and efficiently.



Q The primary function of a Database Management System (DBMS) is best described by which of the following? **(Cognizant)**

- (A) Creating user interfaces for applications
- (B) Managing databases and controlling data access effectively
- (C) Writing programming code for software applications
- (D) Operating system management

Answer: B, **Explanation:** DBMS effectively manages and controls data access.

Problems With File System

- **Data Redundancy and Inconsistency**
 - **Definition:** Same data stored multiple times causing confusion. **Example:** Student address saved separately in exam and fee sections, creating confusion if address changes.
- **Difficulty in Accessing Data**
 - **Definition:** Difficulty retrieving data due to lack of proper structure. **Example:** Finding a specific student's old marks in multiple Excel sheets takes too much time.
- **Data Isolation**
 - **Definition:** Data stored in different files/formats, making it hard to combine. **Example:** Combining student attendance data stored separately by different teachers becomes difficult.
- **Integrity Problem**
 - **Definition:** Difficulty enforcing rules for correct data. **Example:** A student's marks entered as 150 out of 100 by mistake, due to lack of proper checks.
- **Atomicity Problem**
 - **Definition:** Partial updates causing inconsistent data after failures. **Example:** Online payment fails after deducting money but before confirming admission.
- **Concurrent Access Anomalies**
 - **Definition:** Issues occurring when multiple users access data simultaneously. **Example:** Two students booking the last seat in a class simultaneously, causing conflict.

- Q** Which of the following is a significant disadvantage of the file system compared to a database system? **(Accenture)**
- (A)** High levels of data security
 - (B)** Reduced data redundancy
 - (C)** Increased risk of data inconsistency
 - (D)** Efficient concurrent data access

Answer: C, **Explanation:** File systems often lead to data inconsistency.

Characteristics of Database Approach

- The **database approach** means managing data using a Database Management System (DBMS). It helps to solve many problems faced with traditional file systems and makes data handling easier, efficient, and organized. Here are some important characteristics:
 - **Data Independence**: Changing database structures doesn't affect existing applications. **Example:** College moves student data from Excel sheets to MySQL, but student login stays the same.
 - **Reduced Data Redundancy**: Storing each piece of data only once, avoiding unnecessary duplication. **Example:** Your name and address stored only once in the college database instead of repeating in every attendance or exam table.
 - **Concurrency**: Multiple users can access data at the same time without any issues. **Example:** Many students checking and registering for elective subjects simultaneously online without conflicts.
 - **Security**: Protecting data by giving access only to authorized users. **Example:** Only teachers can update student marks, while students can only see their own results.
 - **Integrity**: Ensuring correctness and consistency of data by applying rules.
Example: A system doesn't allow entering marks greater than 100, ensuring valid entries.

Q Concurrency in database systems enables multiple users to access data simultaneously without conflicts. Which real-world scenario illustrates concurrency best? (HCL)

- (A) A single user editing a document alone
- (B) Multiple customers booking flight tickets simultaneously
- (C) Backing up a database at midnight
- (D) Encrypting data files periodically

Answer: B, **Explanation:** Booking systems manage concurrency efficiently.

Types of Databases

- **Relational Database:** **Definition:** Stores data in tables with rows and columns, connected by keys and relationships. **Industrial Example:** Oracle, MySQL, Microsoft SQL Server. **Commercial Use:** Banks, colleges, and online stores use relational databases for structured, organized data management.
- **Hierarchical Database:** **Definition:** Stores data in a tree-like structure with parent-child relationships; each child has only one parent. **Industrial Example:** IBM's Information Management System (IMS). **Commercial Use:** Telecom and banking sectors use it for clear parent-child data structures, like customer accounts and their transactions.
- **Network Database:** **Definition:** Stores data in a network of linked records, allowing multiple relationships, like a graph structure. **Industrial Example:** Integrated Data Store (IDS) by Honeywell/Bull. **Commercial Use:** Manufacturing and logistics companies historically used it to manage complex interconnected data, like products and suppliers.
- **NoSQL Database:** **Definition:** Handles large amounts of unstructured or semi-structured data, offering flexibility and scalability without fixed tables. **Industrial Example:** MongoDB, Cassandra, Redis.
Commercial Use: Social media (like Facebook), e-commerce (like Amazon), and real-time analytics platforms use NoSQL to manage rapidly changing, vast data efficiently.

Q MongoDB is categorized under which type of database? (Capgemini)

- (A) Relational database
- (B) Hierarchical database
- (C) Network database
- (D) NoSQL database

Answer: D, Explanation: MongoDB is a popular NoSQL database.

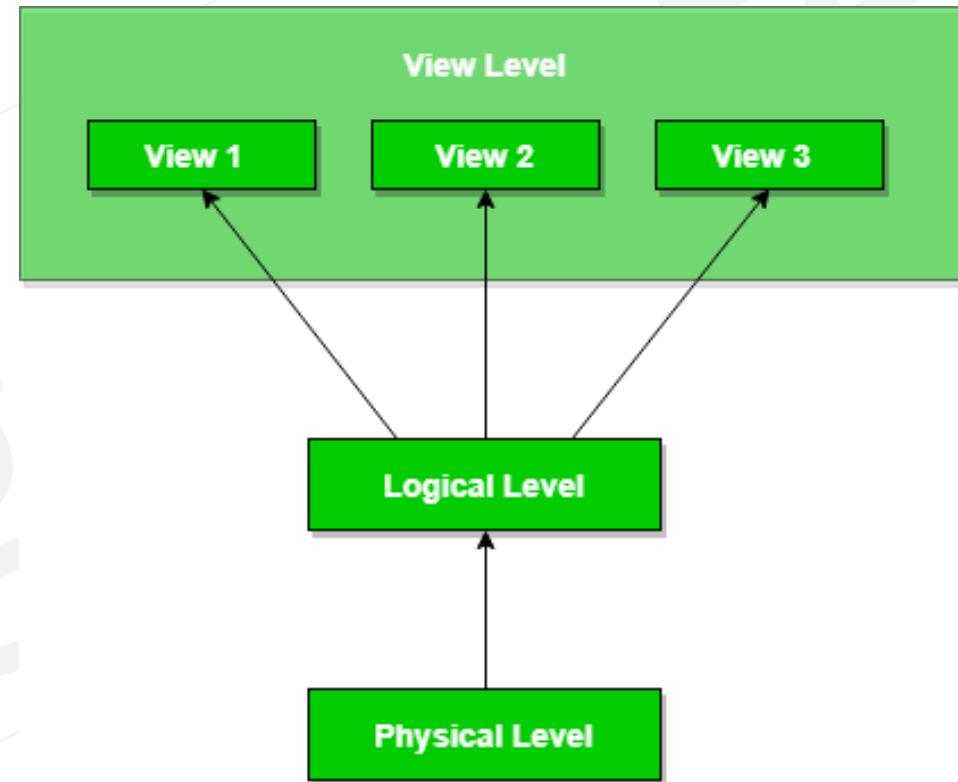
Views of Database

- A database can be viewed at three levels, each showing different levels of data detail and abstraction.

- **Physical Level (Internal Level)**: **Definition:** Lowest level describing exactly how data is stored physically in the hardware, including complex data structures and file organization. **Who uses it:** Database Administrator (DBA) only. **Example:** Exact file storage locations, indexing methods, and data compression techniques.

- **Conceptual Level (Logical Level)**: **Definition:** Describes the overall logical structure of the database, including entities, data types, and relationships, without physical details. **Who uses it:** Database designers and developers. **Example:** Tables like Student, Course, and Teacher and the relationships between these tables.

- **External Level (View Level)**: **Definition:** Highest level, showing only the necessary part of the database to the users. It helps users access only relevant data. **Who uses it:** End users and applications. **Example:** Student portal displaying only your grades, attendance, and schedule without showing other students' data.



Q Which database view describes how the data is physically stored on storage media like hard disks? **(IBM)**

- (A)** External view
- (B)** Logical view
- (C)** Physical view
- (D)** Conceptual view

Answer: C, **Explanation:** Physical view shows the data storage format.

Instance and Schemas

- **Instance:** An instance of a database is the actual collection of data present at a particular point of time. It's essentially a snapshot of the database showing what data currently exists. **Example:** At the end of a semester, the database instance might include all current students' grades, attendance records, and exam results for that specific semester.
- **Schema:** A database schema defines the overall logical structure and design of the database. It describes how data is organized, including tables, relationships, attributes, and constraints, but does not include the actual data. **Example:** A student database schema would specify tables like Students (StudentID, Name, Address), Courses (CourseID, CourseName), and Grades (StudentID, CourseID, Grade), along with their relationships and constraints, even before entering actual student data.

Q A database instance represents which of the following? (Tech Mahindra)

- (A) Logical structure of database
- (B) Database snapshot at a specific time
- (C) User access control policies
- (D) Database backup procedures

Answer: B, Explanation: Instance represents database state at a moment in time.

Q The schema of a database refers to: **(Mindtree)**

- (A)** Data entries at a specific time
- (B)** Physical storage devices
- (C)** Structural definition and organization of data
- (D)** Database management software tools

Answer: C, **Explanation:** Schema outlines the structure and organization of data.

OLAP Vs OLTP

- In real-world applications, databases handle two very different types of tasks:
 - **Daily transactions**, which must be quick and accurate, like banking or ticket booking.
 - **Analytical tasks**, where historical data is analyzed to make decisions, like sales analysis or trend prediction.
- These two tasks have entirely different requirements. Using a single database for both makes the system slow and inefficient. Therefore, databases are divided into two specialized categories: **OLTP (Online Transaction Processing)** for daily operations, and **OLAP (Online Analytical Processing)** for analyzing data and making decisions.

Basis of Difference	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Definition	Handles daily transactions efficiently and accurately.	Used for analyzing large volumes of historical data to make strategic decisions.
Main Purpose	Ensuring fast and smooth daily operations.	Analyzing data to help businesses make informed decisions.
Type of Queries	Short, simple queries involving small datasets.	Long, complex queries involving large historical datasets.
Commercial Example	ATM operations, online shopping checkout.	Amazon analyzing customer purchases to suggest new products.
Typical Users	Customers, operational employees (e.g., cashier, bank teller).	Business analysts, executives, managers.

Q OLTP systems are primarily designed for handling which type of database operations? (LTI)

- (A) Complex analytical queries
- (B) Daily operational transactions
- (C) Large-scale historical data analysis
- (D) Multidimensional data aggregations

Answer: B, **Explanation:** OLTP focuses on www.knowledgigate.com daily operational transactions.

DBA

- **Who is a DBA:** A Database Administrator is the person responsible for managing, securing, and maintaining databases to ensure they run smoothly and efficiently.
- **Basic Roles of a DBA:**
 - Ensuring database **security** and protecting data from unauthorized access.
 - Regularly performing **backup and recovery** to prevent data loss.
 - **Tuning and optimizing** database performance for faster data access.
- **Why is DBA Important in Academics?**
 - DBA concepts are crucial because they teach students how databases are practically managed, maintained, and secured in real-world scenarios. **Example:** A DBA ensures student marks and attendance data remain safe, backed up, and easily accessible at all times.
- **Commercial Example:** Large companies like **TCS, Infosys, Oracle, and Amazon** employ specialized DBAs to handle vast amounts of sensitive data securely and efficiently.
- **Famous Database Administrators:** **Larry Ellison** (Oracle co-founder) is known globally for his pioneering role in database management systems.



Q What is one of the primary responsibilities of a Database Administrator (DBA)? **(Deloitte)**

- (A)** Writing application software
- (B)** Ensuring database security and integrity
- (C)** Installing operating systems
- (D)** Performing hardware maintenance

Data Warehousing

- **Why Do We Need Data Warehousing:** Businesses gather data from various sources, making it challenging to analyze it efficiently if stored separately. To overcome this, data warehousing emerged in the 1980s–90s, enabling organizations to collect and analyze historical data from multiple sources at a single place, improving decision-making processes.
- **What is a Data Warehouse:** A **Data Warehouse** is a centralized storage system that integrates large volumes of historical data from different sources, specifically structured to support analysis, reporting, and decision-making.
- **Example (Simple & Commercial):** A university combines attendance, marks, and student feedback into one database (data warehouse) to analyse overall student performance. Companies like Amazon and Flipkart use data warehouses to study customer buying behaviour, making effective product recommendations.
- **Importance:**
 - Helps organizations make informed, data-driven decisions.
 - Enables efficient analysis of historical trends, improving business strategies.

Data Mining

- **Why Do We Need Data Mining:** Companies collect massive amounts of data every day. But simply storing data isn't enough. Businesses need ways to discover hidden patterns and valuable insights from this data. Data mining, popular since the 1990s, helps organizations analyse large datasets effectively to predict future trends and make better decisions.
- **What is Data Mining:** **Data mining** is the process of discovering hidden patterns, relationships, and useful information from large databases to support decision-making and predict future behavior.
- **Example (Simple & Commercial):** A supermarket notices that customers buying bread also often buy milk, so they place these items together to increase sales. Banks (like HDFC or ICICI) use data mining to detect suspicious transactions or potential fraud based on unusual customer activity.

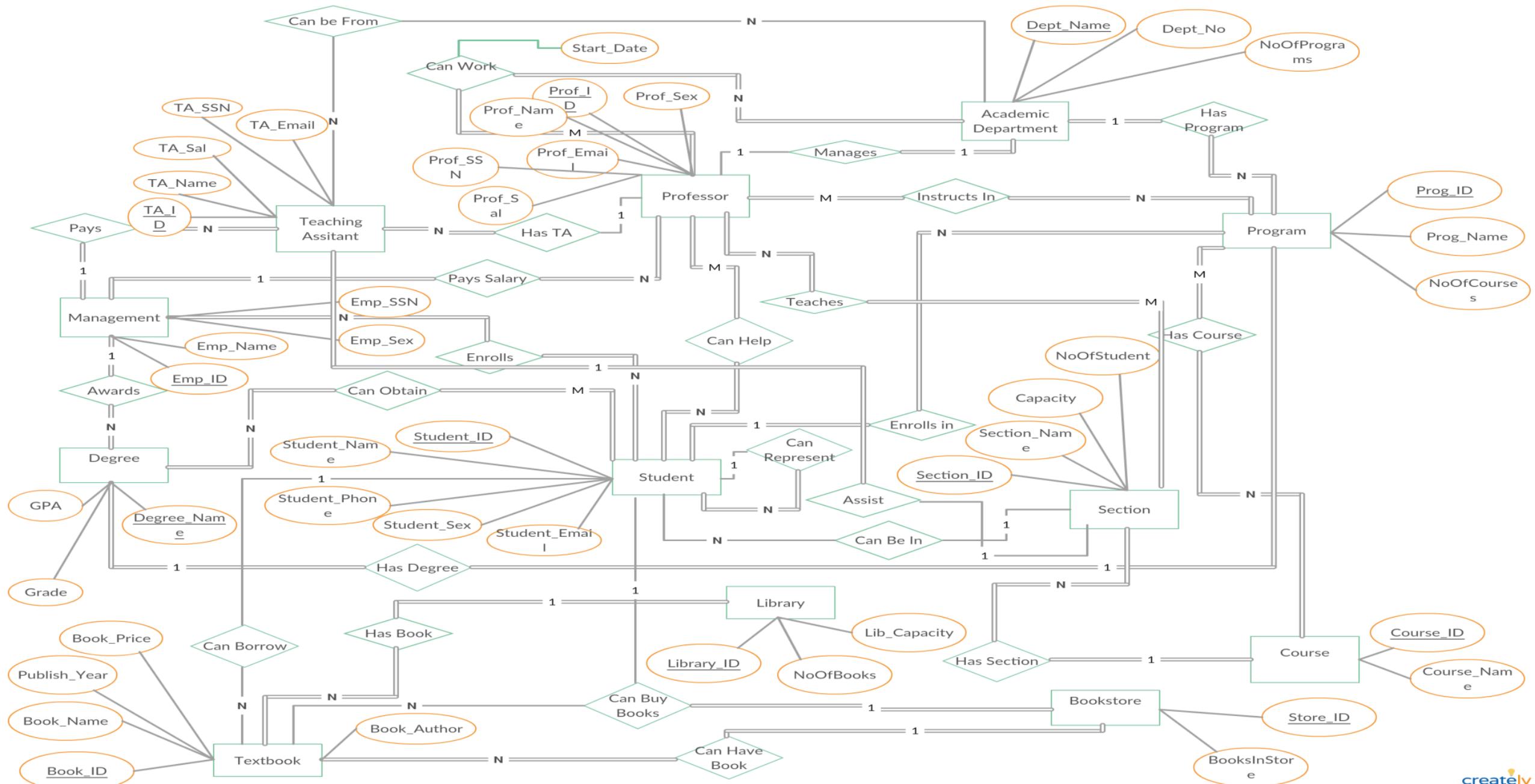
Data Science

- **Why Do We Need Data Science:** In today's digital world, massive amounts of data are generated every day. Businesses need to analyze and use this data effectively to make smarter decisions and predictions. This need gave rise to Data Science, an interdisciplinary field that emerged prominently in the 2000s, combining statistics, computing, and domain expertise to extract valuable insights from data.
- **What is Data Science:** **Data Science** is the field of extracting meaningful insights from data using scientific methods, algorithms, and statistical techniques to solve real-world problems.
- **Example (Simple & Commercial):** Netflix analyzes viewers' watching history to recommend movies and series that match individual tastes. Swiggy and Zomato use Data Science to predict delivery times and optimize routes, ensuring faster deliveries.
- In short:
 - **Data Warehouse** stores data.
 - **Data Mining** finds hidden patterns.
 - **Data Science** combines all these to predict and solve problems.

E-R DIAGRAM/MODEL

- **Introduction:**
 - The **Entity-Relationship (E-R) Diagram** was introduced by **Dr. Peter Chen** in **1976**. It provides a simple, visual way to design databases by representing real-world scenarios.
- **Purpose and Use:**
 - The E-R model helps in designing databases by clearly defining entities (basic objects), their attributes (properties), and relationships among entities. It maps real-world scenarios onto a conceptual database design.
- **Benefits:**
 - Easy to understand as it uses simple diagrams.
 - Removes confusion by clearly showing data structure and interactions.
 - Suitable even for non-technical users, making database design intuitive and straightforward.





Q: Entity-Relationship diagrams, widely used in database design, were introduced in 1976. Who proposed this concept initially? (TCS)

- (A) Edgar F. Codd
- (B) Dr. Peter Chen
- (C) Charles Bachman
- (D) James Gosling

Answer: B, **Explanation:** ER diagrams were introduced by Dr. Peter Chen.

Q: Why are ER diagrams considered beneficial in designing databases, especially when dealing with non-technical stakeholders? **(Infosys)**

- (A)** They are highly technical.
- (B)** They use programming languages.
- (C)** They provide an intuitive visual representation of data.
- (D)** They require extensive technical training.

ENTITY

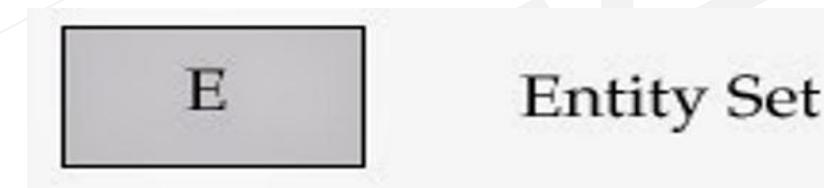
- An entity is a thing or an object in the real world that is distinguishable from other object based on the values of the attributes it possesses. An entity may be **concrete**, such as a person or a book, or it may be **abstract**, such as a course, a course offering, or a flight reservation.
- *Types of Entity*
 - **Tangible** - *Entities which physically exist in real world. E.g. – Car, Pen, locker*
 - **Intangible** - *Entities which exist logically. E.g. – Account, video.*
- In ER diagram we cannot represent an entity, as entity is an instant not schema, and ER diagram is designed to understand schema.
- In a relational model entity is represented by a row or a tuple or a record in a table.

Name	FName	City	Age	Salary
Smith	John	3	35	\$280
Doe	Jane	1	28	\$325
Brown	Scott	3	41	\$265
Howard	Shemp	4	48	\$359
Taylor	Tom	2	22	\$250



- **ENTITY SET**- Collection of same type of entities that share the same properties or attributes.
- In an ER diagram an entity set is represented by a rectangle
- In a relational model it is represented by a separate table

Name	FName	City	Age	Salary
Smith	John	3	35	\$280
Doe	Jane	1	28	\$325
Brown	Scott	3	41	\$265
Howard	Shemp	4	48	\$359
Taylor	Tom	2	22	\$250



**Q: An entity within an ER diagram is typically defined as which of the following?
(Wipro)**

- (A) A specific type of database query**
- (B) A uniquely identifiable object or concept**
- (C) An action or event**
- (D) A property of an object**

Answer: B, Explanation: Entities represent identifiable objects or concepts.

Q: What is the correct definition of an entity set in ER diagrams? (Cognizant)

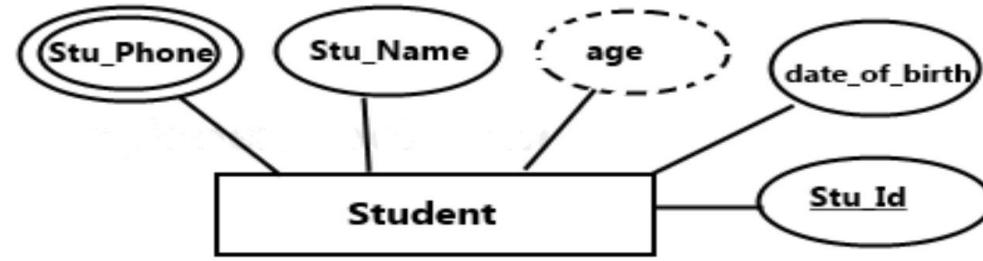
- (A) A single attribute
- (B) A set of different relationships
- (C) A group of similar entities
- (D) A complex database operation

Answer: C,Explanation: An entity set is a collection of entities of the same type.

ATTRIBUTES

- Attributes are the units defines and describe properties and characteristics of entities. Attributes are the descriptive properties possessed by each member of an entity set. for each attribute there is a set of permitted values called domain.

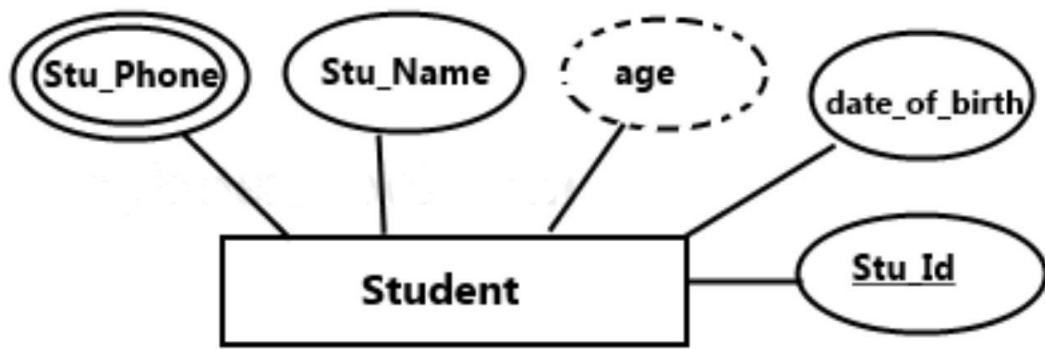
Name	FName	City	Age	Salary
Smith	John	3	35	\$280
Doe	Jane	1	28	\$325
Brown	Scott	3	41	\$265
Howard	Shemp	4	48	\$359
Taylor	Tom	2	22	\$250



- In an ER diagram attributes are represented by ellipse or oval connected to rectangle. While in a relational model they are represented by independent column. e.g. Instructor (ID, name, salary, dept_name)

Types of Attributes

- **Single valued**- Attributes having single value at any instance of time for an entity. E.g. – Aadhar no, dob.
- **Multivalued** - Attributes which can have more than one value for an entity at same time. E.g. - Phone no, email, address.
 - A multivalued attribute is represented by a double ellipse in an ER diagram and by an independent table in a relational model.
 - Separate table for each multivalued attribute, by taking mva and pk of main table as fk in new table



Customer

<u>Customer ID</u>	<u>First Name</u>	<u>Surname</u>	<u>Telephone Number</u>
123	Pooja Singh	Singh	555-861-2025, 192-122-1111
456	San Zhang	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	John Doe	Doe	555-808-9633

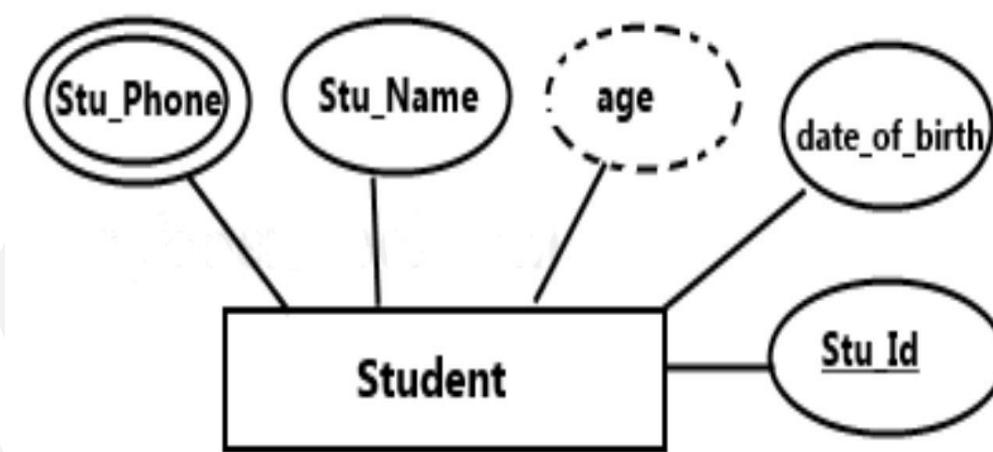
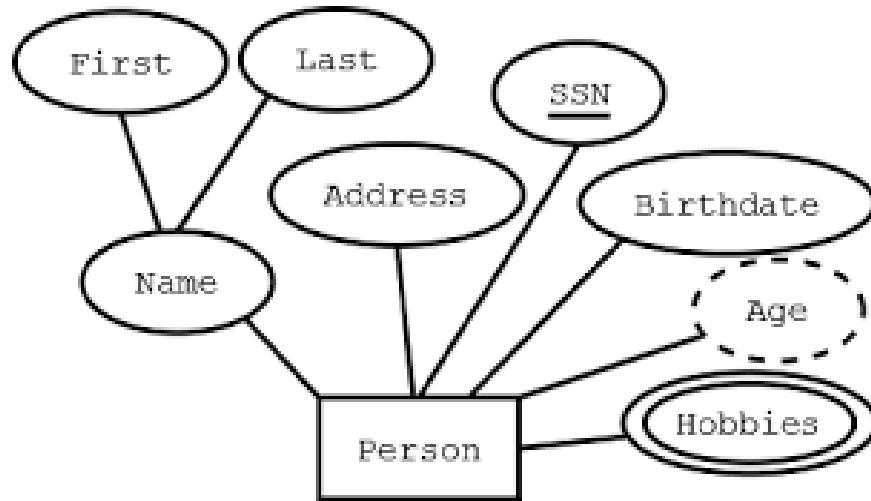


Customer Name		
<u>Customer ID</u>	<u>First Name</u>	<u>Surname</u>
123	Pooja	Singh
456	San	Zhang
789	John	Doe



Customer Phone Number	
<u>Customer ID</u>	<u>Telephone Number</u>
123	555-861-2025
123	192-122-1111
456	(555) 403-1659 Ext. 53
456	182-929-2929
789	555-808-9633

- **Simple** - Attributes which cannot be divided further into sub parts. E.g. Age. **Composite** - Attributes which can be further divided into sub parts, as simple attributes. A composite attribute is represented by an ellipse connected to an ellipse and in a relational model by a separate column.
- **Stored** - Main attributes whose value is permanently stored in database. E.g. date_of_birth. **Derived** - The value of these types of attributes can be derived from values of other Attributes. E.g. - Age attribute can be derived from date_of_birth and Date attribute.



**Q: Which type of attribute can hold multiple values for a single entity in ER modeling?
(Accenture)**

- (A) Simple attribute**
- (B) Composite attribute**
- (C) Derived attribute**
- (D) Multi-valued attribute**

Answer: D, Explanation: Multi-valued attribute can store multiple values, e.g., multiple phone numbers.

Q: Which attribute type is calculated or inferred from other stored attributes in an ER model? (HCL)

- (A) Derived attribute**
- (B) Composite attribute**
- (C) Simple attribute**
- (D) Multi-valued attribute**

Answer: A, Explanation: Derived attributes are computed from other stored attributes (e.g., age from birthdate).

Relationship / Association

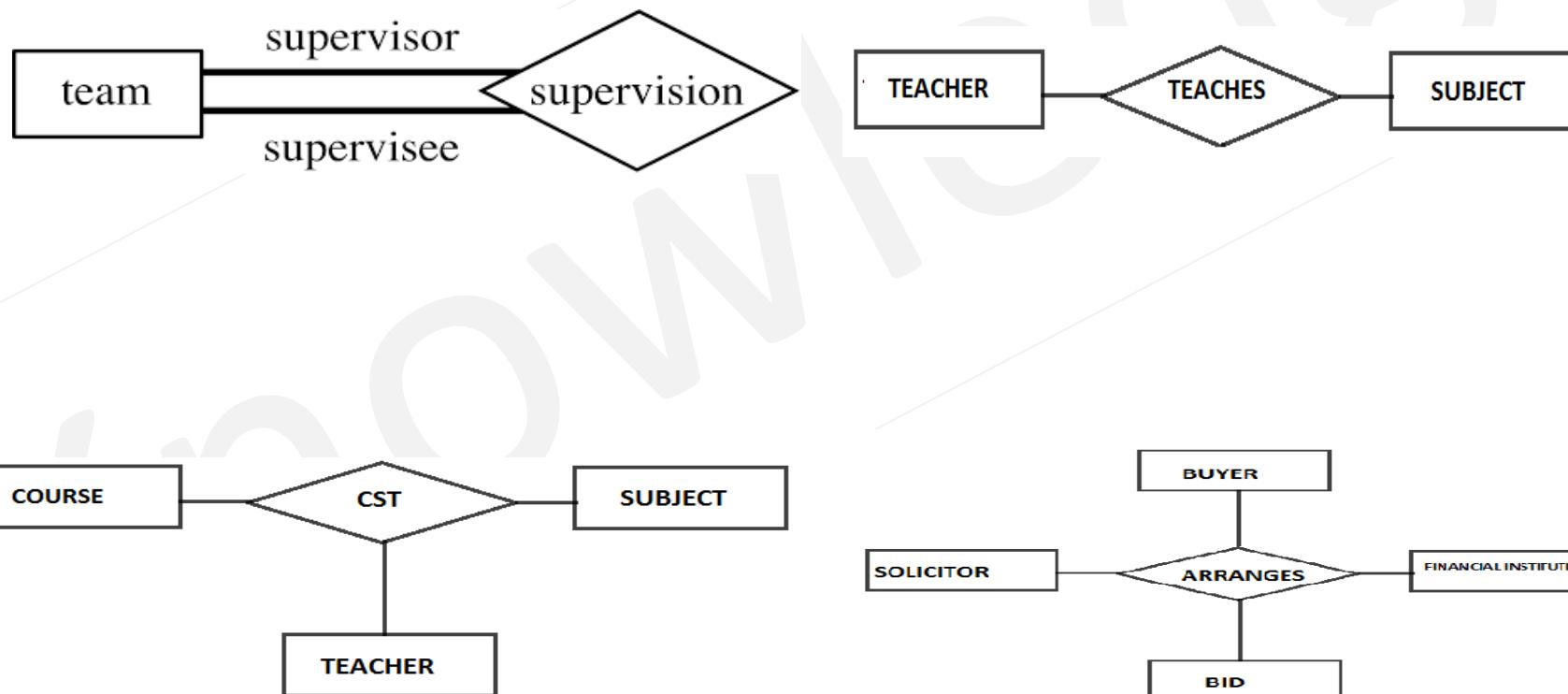
- Is an association between two or more entities of same or different entity set. In ER diagram we cannot represent individual relationship as it is an instance or data.



- In an ER diagram it is represented by a diamond, while in relational model sometimes through foreign key and other time by a separate table.
Note: - normally people use word relationship for relationship type so don't get confused.
- Every relationship type has three components: Ubique Name, Degree, Structural constraints (cardinalities ratios, participation)

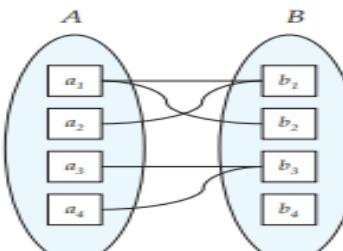
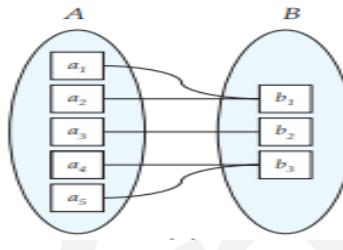
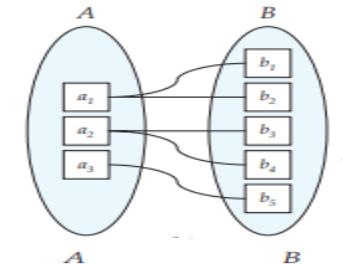
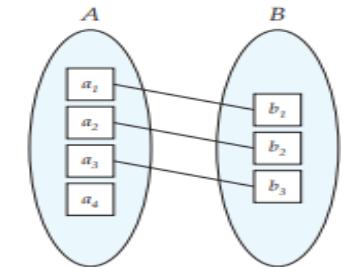
Degree of a relationship/relationship set

- Means number of entities set(relations/tables) associated(participate) in the relationship set. Most of the relationship sets in a data base system are binary. Occasionally however relationship sets involve more than two entity sets. Logically, we can associate any number of entity set in a relationship called N-ary Relationship.



Structural constraints (Cardinalities Ratios, Participation)

- An E-R enterprise schema may define certain constraints to which the contents of a database must conform. Express the number of entities to which another entity can be associated via a relationship set. Four possible categories are- One to One (1:1) Relationship, One to Many (1: M) Relationship, Many to One (M: 1) Relationship, Many to Many (M: N) Relationship.



Participation Constraints

- Participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. These constraints specify the minimum and maximum number of relationship instances that each entity must/can participate in.
- **Max cardinality** – it defines the maximum no of times an entity occurrence participating in a relationship. **Min cardinality** - it defines the minimum no of times an entity occurrence participating in a relationship.
- **PARTICIPATION CONSTRAINTS**- it defines participations of entities of an entity type in a relationship.
 - **PARTIAL PARTICIPATION (min cardinality zero)** - In Partial participation only some entities of entity set participate in Relationship set, that is there exists at least one entity which do not participate in a relation.
 - **TOTAL PARTICIPATION (min cardinality at least one)** - In total participation every entity of an entity set participates in at least one relationship in Relationship set.

Q What is a binary relationship in an ER diagram context? **(Capgemini)**

- (A)** A relationship involving exactly one entity
- (B)** A relationship involving exactly two entities
- (C)** A relationship involving exactly three entities
- (D)** A relationship with unlimited entities

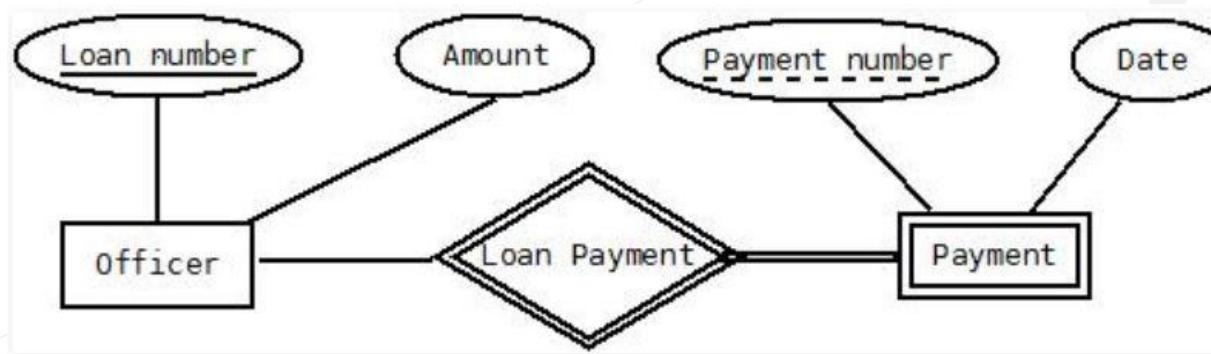
Answer: B, **Explanation:** Binary relationships involve exactly two entities.

Q Which type of participation constraint implies that every entity must participate in the relationship? **(IBM)**

- (A)** Partial participation
- (B)** Complete participation
- (C)** Optional participation
- (D)** Selective participation

STRONG AND WEAK ENTITY SET

- A strong entity set is one that has its own primary key, allowing each entity (tuple) to be uniquely identified. A weak entity set, however, doesn't have sufficient attributes for a primary key and thus relies on another strong entity set (identifying or owner entity set) for unique identification. Weak entity sets contain discriminator attributes (partial keys), which, combined with the primary key of the identifying entity set, uniquely identify each tuple.
- In ER diagrams, weak entity sets are shown as double rectangles, and their relationship with identifying entities (identifying relationship) is represented by double diamonds. This relationship is always many-to-one, with total participation from the weak entity set.
- Weak entity sets are important because they accurately represent entities that logically depend on others. They help maintain database consistency by automatically deleting dependent entities when the strong entity is deleted, avoiding data duplication and inconsistency issues.



Q A weak entity set in an ER diagram is defined as: **(Tech Mahindra)**

- (A)** An entity set without attributes
- (B)** An entity set that cannot exist without a strong entity
- (C)** An entity set with its own independent existence
- (D)** An entity set always used for temporary storage

Answer: B, **Explanation:** Weak entities www.knowledgigate.ai depend on strong entities for their existence.

**Q Strong entities in ER diagrams differ from weak entities primarily because strong entities:
(Mindtree)**

- (A) Require another entity for their existence**
- (B) Lack attributes entirely**
- (C) Can independently exist and have a unique identifier**
- (D) Can never participate in relationships**

Conversion From ER Diagram To Relational Model

- **Entity Set**
 - Convert every strong entity set into a separate table.
 - Convert every weak entity set into a separate table, by making it dependent into one strong entity set (**identifying or owner entity set**).
- **Relationship(Unary)**
 - No separate table is required, add a new column as fk which refer the pk of the same table.
- **Relationship(Binary)**
 - 1:1 No separate table is required, take pk of one side and put it as fk on other side, priority must be given to the side having total participation.
- **Conversion of 1-1 relationship(binary)**
 - No separate table is required, take pk of one side as pk on other side, priority must be given to the side having total participation.
- **Conversion of 1-n or n-1 relationship (binary)**
 - No separate table is required, modify n side by taking pk of 1 side a foreign key on n side.
- **Conversion of n-n relationship (binary)**
 - Separate table is required take pk of both table and declare their combination as a pk of new table.

- **Relationship(3 or More)**
 - Take the pk of all participating entity sets as fk and declare their combinations as pk in the new table.
- **Multivalued Attributes**
 - A separate table must be taken for all multivalued attributes, where we take pk of the main table as fk and declare combination of fk and multivalued attribute are pk in the new table.
- **Composite Attributes**
 - A separate column must be taken for all simple attributes of the composite attribute.

Q: During ER diagram conversion to relational tables, how is a Many-to-Many relationship typically handled? **(LTI)**

- (A)** Ignored completely
- (B)** Represented by adding extra attributes
- (C)** Represented by creating an additional junction table
- (D)** Represented by combining entities into a single table

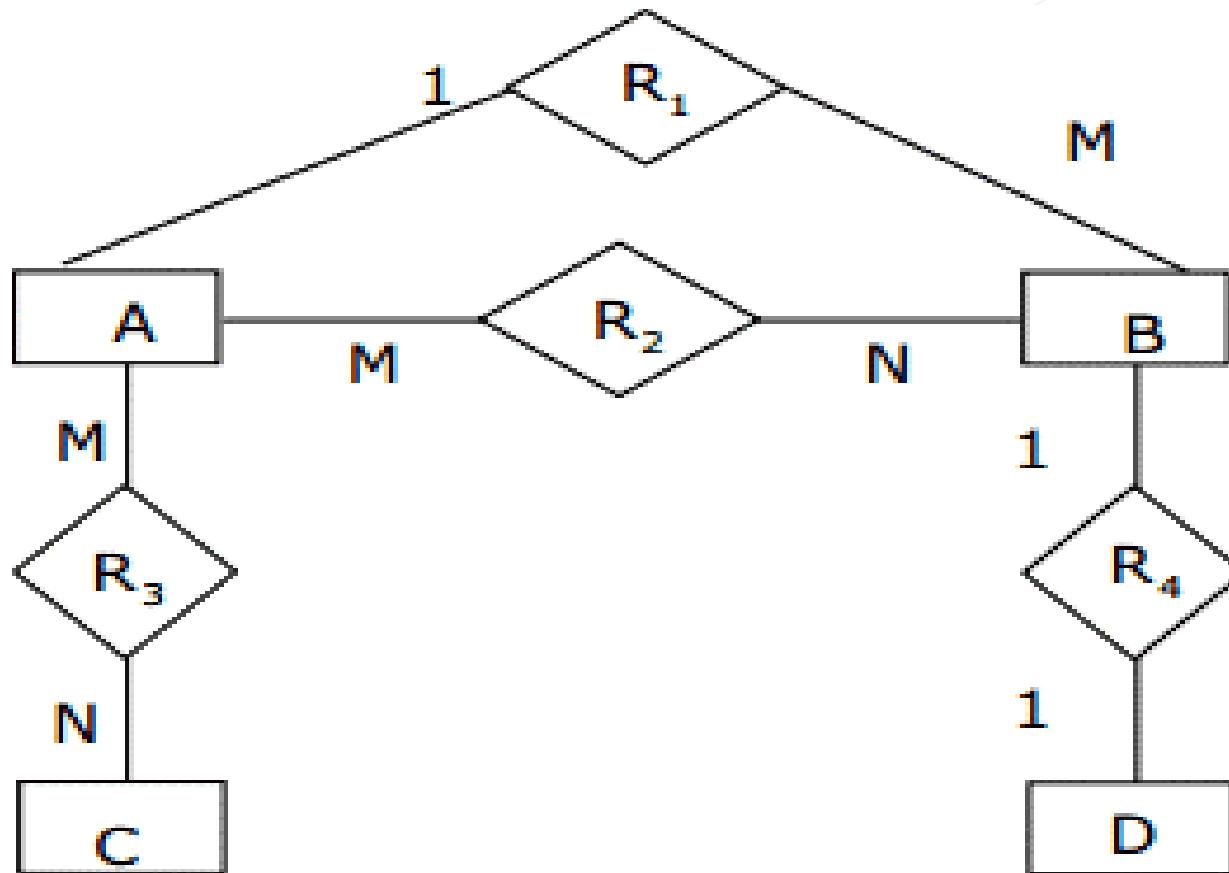
Answer: C, Explanation: Many-to-Many relationships require a separate junction table.

Q When converting a weak entity set into relational tables, which of the following is correct?
(Deloitte)

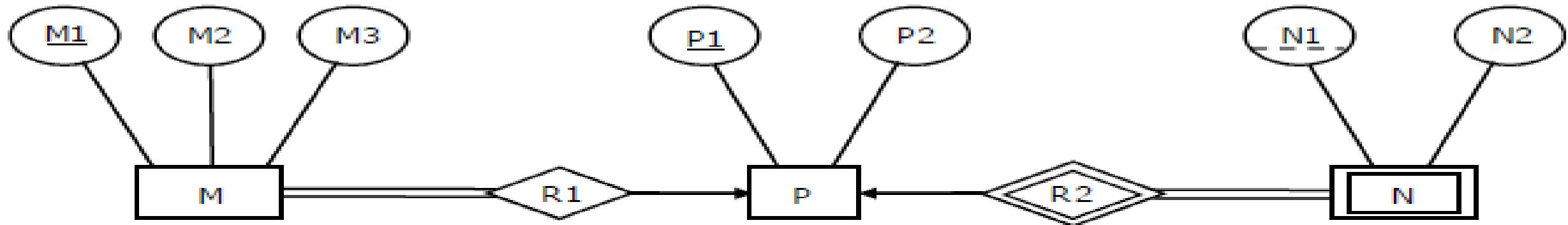
- (A)** Weak entities always merge into the strong entity table
- (B)** Weak entities are not converted into tables
- (C)** Weak entities become separate tables containing the key of the strong entity
- (D)** Weak entities always form relationship tables

Answer: C, **Explanation:** Weak entities are converted into separate tables that include the key from the associated strong entity. www.knowledgegate.ai

Q The minimum number of tables required to convert the following ER diagram to Relational model is _____



Consider the following ER diagram

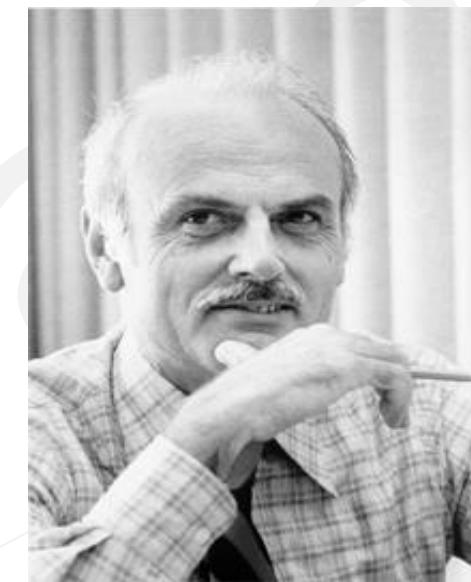


The minimum number of tables needed to represent M, N, P, R₁, R₂ is

- **ADVANTAGES OF E-R DIGRAM**
 - Constructs used in the ER diagram can easily be transformed into relational tables.
 - It is simple and easy to understand with minimum training.
- **DISADVANTAGE OF E-R DIGRAM**
 - Loss of information content.
 - Limited constraints representation.
 - It is overly complex for small projects.

RELATIONAL DATABASE MANAGEMENT SYSTEM

- Relational Database Management System (RDBMS) is a database system based on the relational model, originally introduced by Edgar F. Codd in 1970, who is considered the father of modern relational database design. Most popular databases today, both commercial and open-source (like MySQL, Oracle, and Microsoft SQL Server), follow this relational model. An RDBMS stores data efficiently in tables and uses clearly defined relationships among these tables to ensure data consistency and integrity.



BASICS OF RDBMS

- **Domain (set of permissible value in particular column)** is a set of atomic values. By **atomic** we mean that each value in the domain is indivisible as far as the formal relational model is concerned. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn.
 - E.g. Names: The set of character strings that represent names of persons.
- **Table (Relation)** - A Relation is a set of tuples/rows/entities/records.
- **Tuple** - Each row of a Relation/Table is called Tuple.
- **Arity/Degree** - No. of columns/attributes of a Relation. E.g. - Arity is 5 in Table Student.
- **Cardinality** - No of rows/tuples/record of a Relational instance. E.g. - Cardinality is 4 in table Student.

Domain/
Field/
Column/
Arity/
Degree

NAME	ID	CITY	COUNTRY	HOBBY
NISHA	1	AGRA	INDIA	PLAYING
NIKITA	2	DELHI	INDIA	DANCING
AJAY	3	AGRA	INDIA	CHESS
ARPIT	4	PATNA	INDIA	READING

Rows/Tuples/Record/
Cardinality

Q) In a relational schema, each tuple is divided into fields called [TCS NINJA].

- (a) Relations**
- (b) Domains**
- (c) Queries**
- (d) All of the above**
- (e) None of the above**

Ans=B.

Properties of Relational tables

1. Cells contains atomic values
2. Values in a column are of the same kind
3. Each row is unique
4. No two tables can have the same name in a relational schema.
5. Each column has a unique name
6. The sequence of rows is insignificant
7. The sequence of columns is insignificant.

Problems in relational database

- **Update Anomalies**- Anomalies that cause redundant work to be done during insertion into and Modification of a relation and that may cause accidental loss of information during a deletion from a relation
 - **Insertion anomalies**: An independent piece of information cannot be recorded into a relation unless an irrelevant information must be inserted together at the same time.
 - **Modification anomalies**: The update of a piece of information must occur at multiple locations.
 - **Deletion Anomalies**: The deletion of a piece of information unintentionally removes other information.

Roll no	name	Age	Br_code	Br_name	Br_hod_name
1	A	19	101	Cs	Abc
2	B	18	101	Cs	Abc
3	C	20	101	Cs	Abc
4	D	20	102	Ec	Pqr

Roll no	name	Age	Br_code	Br_name	Br_hod_name
1	A	19	101	Cs	Abc
2	B	18	101	Cs	Abc
3	C	20	101	Cs	Abc
4	D	20	102	Ec	Pqr

PK
↓

FK
↓

PK
↓

Roll no	name	Age	Br_code
1	A	19	101
2	B	18	101
3	C	20	101
4	D	20	102

Br_code	Br_name	Br_hod_name
101	Cs	Abc
102	Ec	Pqr

Purpose of Normalization

- Normalization is a systematic process of organizing database tables clearly and efficiently. Its main goal is to remove redundancy (duplicate data) and inconsistent dependencies, ensuring data accuracy and improving database performance.
- Without normalization, databases become slow, inefficient, and can produce incorrect results due to repeated or conflicting data. Normalization solves this by ensuring each table stores data about only one idea or concept. If a table contains multiple ideas, it is decomposed into smaller, clearly-focused tables.
- To perform normalization effectively, especially in larger databases, we rely on a powerful concept called Functional Dependencies. These dependencies help identify how data items relate, guiding us in properly structuring tables for clarity, accuracy, and efficiency.

FUNCTIONAL DEPENDENCY

- A formal tool for analysis of relational schemas.
- In a Relation R, if ' α ' \sqsubseteq R AND ' β ' \sqsubseteq R, then attribute or a Set of attribute ' α ' Functionally derives an attribute or set of attributes ' β ', iff each ' α ' value is associated with precisely one ' β ' value.
- For all pairs of tuples t_1 and t_2 in R such that
 - If $T_1[\alpha] = T_2[\alpha]$
 - Then, $T_1[\beta] = T_2[\beta]$



X	Y	Z
1	4	2
1	4	3
2	6	3
3	2	2

Q Consider the following relation instance, which of the following dependency doesn't hold

A) $A \rightarrow b$

B) $BC \rightarrow A$

C) $B \rightarrow C$

D) $AC \rightarrow B$

A	B	C
1	2	3
4	2	3
5	3	3

Q Which of the following dependency doesn't hold good?

A) $A \rightarrow BC$

B) $DE \rightarrow C$

C) $C \rightarrow DE$

D) $BC \rightarrow A$

A	B	C	D	E
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

- Trivial Functional dependency - If β is a subset of α , then the functional dependency $\alpha \rightarrow \beta$ will always hold.

जिसका होना न होना बराबर हो



X	Y	Z
1	4	2
1	4	3
2	6	3
3	2	2

Q) A functional dependency of the form $x \rightarrow y$ is trivial if. [Amcat].

- (A) $y \subseteq x$
- (B) $y \subset x$
- (C) $x \subset y$
- (D) $x \subset y$ and $y \subset x$

Ans=A.

ATTRIBUTES CLOSURE/CLOSURE ON ATTRIBUTE SET/ CLOSURE SET OF ATTRIBUTES

- Attribute closure (also known as closure of attribute set) is the set of all attributes that can be functionally determined from a given set of attributes using the provided functional dependencies (FDs). It includes attributes that can be directly determined by the given dependencies, as well as those that can be logically derived. If we have an attribute set “X,” its attribute closure is represented as X^+ .

$R(A, B, C, D)$

$A \rightarrow B$
 $B \rightarrow C$
 $AB \rightarrow D$

$A^+ =$

$R(ABCDE)$

$A \rightarrow BC$
 $CD \rightarrow E$
 $B \rightarrow D$
 $E \rightarrow A$

$(B)^+ =$

$R(ABCDEF)$

$AB \rightarrow C$
 $BC \rightarrow AD$
 $D \rightarrow E$
 $CF \rightarrow B$

$(AB)^+ =$

ARMSTRONG'S AXIOMS



- Armstrong's Axioms are a set of fundamental rules or principles used to find all possible functional dependencies in a relational database. Introduced by William W. Armstrong in 1974, these axioms act as basic inference rules, helping us logically determine new dependencies from existing ones. Armstrong's Axioms are considered “sound,” meaning they correctly generate only valid functional dependencies within the closure (denoted as F^+) of a given set of dependencies (denoted as F).
- **Armstrong Axioms(Sound and Complete)**
 - **Reflexivity:** If Y is a subset of X , then $X \rightarrow Y$
 - **Augmentation:** If $X \rightarrow Y$, then $XZ \rightarrow YZ$
 - **Transitivity:** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

From these rules, we can derive these secondary rules-

- **Union:** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- **Decomposition:** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- **Pseudo transitivity:** If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

Q Which of the following is not Armstrong's Axiom? [Asked in Accenture]

- a) Reflexivity rule
- b) Transitivity rule
- c) Pseudotransitivity rule
- d) Augmentation rule

Ans=C

Q Consider a relation R(A, B, C), which of the following statements is not true according to inference rules for functional dependencies? [Asked in TCS NQT]

- a) If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
- b) If $AB \rightarrow C$, then $A \rightarrow B$ and $B \rightarrow C$
- c) If $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$
- d) If $A \rightarrow B$, then $AC \rightarrow BC$

Ans: B

Key

- For identifying the uniqueness of a tuple, we take help of an attribute or set of attributes. Uniqueness of tuple is needed as a Relation is a Set and Set disallows duplicity of elements. Various Keys used in database System are as follows-



RegNo	Name	Email	Phone	Aadhaar	Address
101	Alice Rao	alice@abc.edu	9123456780	1111-2222-3333	12 Park Street
102	Bob Singh	bob@abc.edu	9123456781	2222-3333-4444	55 Lake Road
103	Charlie Das	charlie@abc.edu	9123456782	3333-4444-5555	12 Park Street
104	Bob Singh	diana@abc.edu	9123456783	4444-5555-6666	78 Hill Avenue
105	Eva Nair	eva@abc.edu	9123456784	5555-6666-7777	99 River Lane

Super key

- A Super Key is a set of one or more attributes in a relation that uniquely identifies each tuple (row) within that relation. In other words, no two tuples can have the same values for these attributes.
- Formally, if we have a set of attributes X in a relation R, and if the attribute closure of X (denoted as X^+) includes all the attributes of R, then X is called a Super Key.
- Every relation must have at least one super key that doesn't contain null values. For a relation with 'n' attributes, the maximum number of possible super keys is $2^n - 1$. The largest super key possible includes all attributes of the relation.
- A **super key** can contain attributes that individually allow null values, but to uniquely identify each tuple, at least one attribute within the super key must always have a **NOT NULL** constraint. In simpler terms, a super key cannot be entirely null because, if it were, it wouldn't be able to uniquely identify records. At least one attribute within a super key must always have valid (non-null) data to maintain uniqueness and integrity.

Q. Consider a relation R(A,B,C,D,E) with the following functional dependencies:

$ABC \rightarrow DE$ and

$D \rightarrow AB$

The number of super keys of R is: [Hexaware]

- (A) 2
- (B) 7
- (C) 10
- (D) 12

Answer : C

- **Candidate key:** A candidate key is the smallest set of attributes required to uniquely identify every tuple (row) within a relation. It is also known as a minimal super key, meaning no subset of a candidate key can act as a super key. Every relation must have at least one candidate key with a NOT NULL constraint to maintain data integrity. Attributes that form part of any candidate key are known as **prime attributes**.
- **Primary key:** A primary key is one of the candidate keys selected by the database administrator to uniquely identify tuples in a relation. Attributes chosen as the primary key cannot contain NULL values, ensuring each row remains distinct and clearly identifiable. Each table can have at most one primary key. Any candidate keys not selected as the primary key are called alternate keys.
- **Composite key** – Composite key is a key composed of more than one column sometimes it is also known as concatenated key.
- **Secondary key** – Secondary key is a key used to speed up the search and retrieval contrary to primary key, a secondary key does not necessarily contain unique values.

Q) The subset of super key is a candidate key under what condition? [Amcat].

- (a) No proper subset is a super key
- (b) All subsets are super keys
- (c) Subset is a super key
- (d) Each subset is a super key

Q Which of the following options are correct regarding these three keys (Primary Key, Super Key, and Candidate Key) in a database? [Asked in E-Litmus]

- I. Minimal super key is a candidate key
 - II. Only one candidate key can be a primary key
 - III. All super keys can be a candidate key
 - IV. We cannot find a primary key from the candidate key
- a) I and II
- b) II and III
- c) I and III
- d) II and IV

Q Class (course id, title, dept name, credits, sec id, semester, YEAR, building, room NUMBER, capacity, TIME slot id)

The SET OF functional dependencies that we require TO hold ON class are:

course id->title, dept name, credits

building, room number->capacity

course id, sec id, semester, year->building, room NUMBER, TIME slot id

A candidate KEY FOR this schema IS {course id, sec id, semester, YEAR}

Consider the above conditions. Which of the following relation holds?

a) Course id-> title, dept name, credits

b) Title-> dept name, credits

c) Dept name-> credits

d) Cannot be determined

[Asked in TCS NQT]

Q) A _____ is a property of the entire relation, rather than of the individual tuples in which each tuple is unique. [Amcat]

- (a) Rows
- (b) Key
- (c) Attribute
- (d) Fields

Foreign Keys

- A foreign key is a column or group of columns in a relational database table that refers the primary key of the same table or some other table to represent relationship. The ***concept of referential integrity*** is derived from foreign key theory.

Roll no	CR
1	1
2	2
3	1
4	2
5	1
6	2
7	1
8	2
9	1
10	2
11	1
12	2
13	1
14	2
15	null

Roll no	name	Age	Br_code	Br_name	Br_hod_name
1	A	19	101	Cs	Abc
2	B	18	101	Cs	Abc
3	C	20	101	Cs	Abc
4	D	20	102	Ec	Pqr

Roll no	name	Age	Br_code
1	A	19	101
2	B	18	101
3	C	20	101
4	D	20	102

Br_code	Br_name	Br_hod_name
101	Cs	Abc
102	Ec	Pqr

Q) An attribute in a relation is a foreign key if the _____ key from one relation is used as an attribute in that relation. [Amcat].

- (a) Candidate
- (b) Primary
- (c) Super
- (d) Sub

Q) A _____ integrity constraint, requires that the values appearing in specified attributes of any tuple in the referencing relation also appear in specified attributes of at least one tuple in the referenced relation.[Amcat].

(a) Referential

(b) Referencing

(c) Specific

(d) Primary

Q Consider the following statements S_1 and S_2 about the relational data model:

- S_1 : A relation scheme can have at most one foreign key.
- S_2 : A foreign key in a relation scheme R cannot be used to refer to tuples of R.

Which one of the following choices is correct? [Asked in Hexaware]

(a) Both S_1 and S_2 are true

(b) S_1 is true and S_2 is false

(c) S_1 is false and S_2 is true

(d) Both S_1 and S_2 are false

Q) Which is correct difference between primary key and foreign key.

[Tech Mahindra]

- (a) A table can have multiple primary key and single foreign key.
- (b) A primary key cannot ignore NULL value but foreign key can.
- (c) A primary key can have duplicate value but foreign does not.
- (d) None of the above.

Q Consider the following table consisting of two attributes A and B, where 'B' is the foreign key referring the candidate key 'A' with on – delete cascade option. When we delete the tuple (3 2), we need to delete few tuples additionally in order to preserve the referential integrity. The number of tuples that are remaining in the table when we delete (3 2) and additional tuples if necessary is _____

A	B
8	9
4	6
7	6
3	2
6	5
5	1
1	2
2	3

Q) A combination of two or more columns used to identify particular rows in a relation is _____. [Amcat]

- (a) Record**
- (b) Field**
- (c) Composite key**
- (d) Foreign key**

Q) Consider the tables store and sales given below.[infosys]

Store_id	City	Region
S001	New York	East
S002	Chicago	Central
S003	Atlanta	East
S004	Los Angeles	West
S005	SanFrancisco	West
S006	Philadelphia	East

Table: sales

Productid	Desc	Store_id
P204	biscuits	S001
P205	shampoo	S004
P204	biscuits	S002
P203	soap	S003
P206	rice	S005
P201	wheat	S001

Which is the best primary key for the sales table from the following?

- (A) desc
- (B) {productid,store_id}
- (C) productid
- (D) store_id

Answer : C

Q R(ABCD)

A → B

B → C

C → A

A

B

C

D

Q R(ABCDEFGHIJ)

AB → C

A B C D E F G H I J

A → DE

B → F

F → GH

D → IJ

Q R(ABCDE)

A → B

A

B

C

D

E

BC → E

DE → A

Q R(ABCD)

AB → CD

C → A

D → B

A

B

C

D

Q R(WXYZ)

Z → W

Y → XZ

XW → Y

W

X

Y

Z

Q Suppose relation R(A,B,C,D,E) has the following functional dependencies: [Asked in Tech Mahindra]

$A \rightarrow B$

$B \rightarrow C$

$BC \rightarrow A$

$A \rightarrow D$

$E \rightarrow A$

$D \rightarrow E$

A B C D E

Which of the following is not a key?

- a) A
- b) E
- c) B, C
- d) D

Q Given a relation R(A,B,C,D,E,F) and set of functional dependency (FD)
 $F = \{A \rightarrow BC, C \rightarrow E, E \rightarrow F, F \rightarrow AB\}$. How many candidate keys does the relation R have?

a) 1

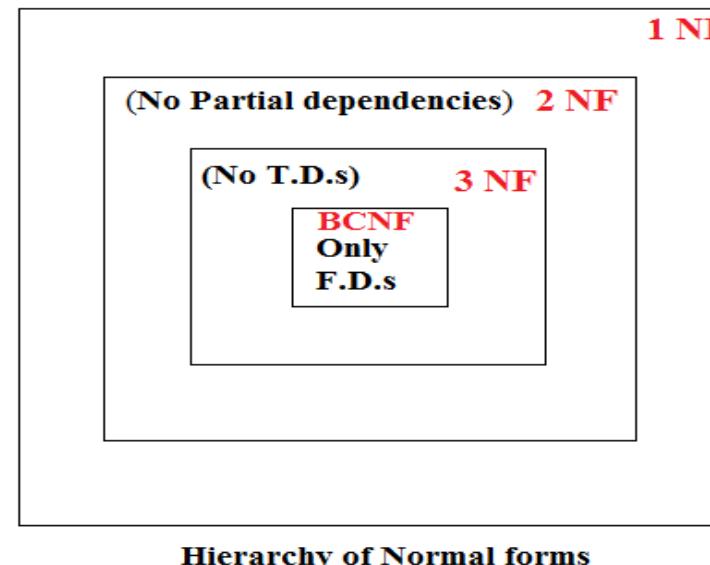
b) 3

c) 4

d) 5

A B C D E F

- Normalization (or decomposition of relations) is a structured process used in database design to minimize redundancy and avoid data inconsistencies. It involves breaking down larger tables into smaller, clearly focused tables while preserving all essential relationships and information.
- To perform normalization, we primarily use tools such as functional dependencies and candidate keys. Functional dependencies help us analyze the relationships among attributes, guiding us through the normalization steps. Using functional dependencies, we can effectively normalize a database schema up to Boyce-Codd Normal Form (BCNF).
- Normalization proceeds through a series of standard forms, each improving database efficiency and clarity:
 - First Normal Form (1NF) → Second Normal Form (2NF) → Third Normal Form (3NF) → Boyce-Codd Normal Form (BCNF).



FIRST NORMAL FORM

- A table (relation) is said to be in **First Normal Form (1NF)** if every cell contains only a single (atomic) value. In other words, attributes should not hold multiple values or composite (combined) values within one cell.
- To satisfy 1NF:
 - Each row must be unique; no two rows should be identical.
 - The table must have a primary key to uniquely identify each row.
 - Each column (attribute) should have a distinct and meaningful name.
 - The order of rows and columns does not matter in a relational table.

Q Relations produced from E - R Model will always be in _____. [Asked in Tech Mahindra]

- (a) 1 NF
- (b) 2 NF
- (c) 3 NF
- (d) 4 NF

- **Prime attribute**: - A attribute is said to be prime if it is part of any of the candidate key
- **Non-Prime attribute**: - A attribute is said to be non-prime if it is not part of any of the candidate key
 - $R(ABCD)$
 - $AB \rightarrow CD$
- Here candidate key is AB so, A and B are prime attribute, C and D are non-prime attributes.
- **PARTIAL DEPENDENCY**- When a non – prime attribute is dependent only on a part (Proper subset) of candidate key then it is called partial dependency. (PRIME > NON-PRIME)
- **TOTAL DEPENDENCY**- When a non – prime attribute is dependent on the entire candidate key then it is called total dependency.
- e.g. $R(ABCD)$ $AB \rightarrow D$, $A \rightarrow C$

SECOND NORMAL FORM

- Relation R is in 2NF if,
 - R should be in 1 NF.
 - R should not contain any Partial dependency. (that is every non-prime attribute should be fully dependent upon candidate key)

$R(A, B, C) \quad B \rightarrow C$

A	B	C
a	1	X
b	2	Y
a	3	Z
C	3	Z
D	3	Z
E	3	Z

A	B
A	1
B	2
A	3
C	3
D	3
E	3

B	C
1	X
2	Y
3	Z

TRANSITIVE DEPENDENCY – A functional dependency from non-Prime attribute to non-Prime attribute is called transitive

E.g.- R(A, B, C) with A as a candidate key

$A \rightarrow B$

$B \rightarrow C$ [transitive dependency]

THIRD NORMAL FORM

- Let R be the relational schema, it is said to be in 3 NF
 - R should be in 2NF
 - It must not contain any transitive dependency
- A relational schema R is said to be 3 NF if every functional dependency in R from $\alpha \rightarrow \beta$, either α is super key or β is the prime attribute

A	B	C
A	1	P
B	2	Q
C	2	Q
D	2	Q
E	3	R
F	3	R
G	4	S

A	B
A	1
B	2
C	2
D	2
E	3
F	3
G	4

B	C
1	P
2	Q
3	R
4	S

Q Which normal form is considered as adequate for usual database design?(Infosys)

- (A)** 2NF
- (B)** 3NF
- (C)** 4NF
- (D)** 5NF

Q Third normal form is based on the concept of _____. (TCS)

- (A)** Closure Dependency
- (B)** Transitive Dependency
- (C)** Normal Dependency
- (D)** Functional Dependency

R(A, B, C)

AB → C

C → B

A	B	C
A	C	B
B	B	C
B	A	D
A	A	E
C	C	B
D	C	B
E	C	B
F	C	B

A	B
A	B
B	B
B	A
A	A
C	C
D	C
e	C
f	c

C	B
B	C
C	B
D	A
E	A

BCNF (BOYCE CODD NORMAL FORM)

A relational schema R is said to be BCNF if every functional dependency in R from

$A \rightarrow \beta$

α must be a super key

E.g.- R (A, B, C, D)

{

AB->C

C->D

D->A

} Candidate key= {AB}, {DB}, {CB}

$Q \ R(ABC) \ (AB, BC)$

$AB \rightarrow C$

$C \rightarrow A$

A

B

C

Q R(ABCD)(AB)

AB → C

B → D

A B C D

Q R(ABCDE)(CE)

CE → D

D → B

C → A

A B C D E

Q R(ABCDE)(AE)

A → B

B → E

C → D

A B C D

Q A relation is in _____ if an attribute of a composite key is dependent on an attribute of other composite key. [Asked in Amcat]

- a) 2NF
- b) 3NF
- c) BCNF
- d) 1NF

Q In 2NF [Asked in Tech Mahindra]

- a) No functional dependencies (FDs) exist
- b) No multivalued dependencies (MVDs) exist
- c) No partial FDs exist
- d) No partial MVDs exist

Q. Amit has some knowledge about database normalization. He has created a table "customer", which has the following characteristics.

- 1)** Table has transitive dependencies.
- 2)** There are no partial dependencies in the table.
- 3)** There is no column with redundant data in the table In which normal form is the table.[Amcat].

- (a)** 1NF
- (b)** 2NF
- (c)** 3NF
- (d)** BCNF

Q) Third normal form is inadequate in situations where the relation : [Amcat].

- (a) Has multiple candidate keys**
- (b) Has candidate keys that are composite**
- (c) Has overlapped candidate keys**
- (d) None of the above**

Q) Consider the relation given below and find the maximum normal form applicable to them :-

R(A, B) with productions { A \rightarrow B }

R(A, B) with productions { B \rightarrow A }

R(A, B) with productions {A \rightarrow B, B \rightarrow A }

R(A, B, C) with productions {A \rightarrow B, B \rightarrow A, AB \rightarrow C } **[Amcat]**

(a) 1, 2 and 3 are in 3NF and 4 is in BCNF

(b) 1 and 2 are in BCNF and 3 and 4 are in 3NF

(c) All are in 3NF wrong

(d) All are in BCNF

Q) What is normalization? [Hexaware].

- (1). Minimizing redundancy.
- (2). Minimizing insertion, deletion and update anomalies.

How many statement are correct.

- (a) 1
- (b) Both are correct.
- (c) 2
- (d) None of the above.

Q. A table has fields F1, F2, F3, F4, and F5, with the following functional dependencies:

$F_1 \rightarrow F_3$

$F_2 \rightarrow F_4$

$(F_1, F_2) \rightarrow F_5$

in terms of normalization, this table is [Asked in Wipro NLTH]

- (A) 1NF
- (B) 2NF
- (C) 3NF
- (D) None of the mentioned

Answer : A

Q. Let $R(A,B,C,D,E,P,G)$ be a relational schema in which the following FDs are known to hold:

$AB \rightarrow CD$

$DE \rightarrow P$

$C \rightarrow E$

$P \rightarrow C$

$B \rightarrow G$

The relation schema R is [Asked in Amcat 2020]

(A) in BCNF

(B) in 3NF, but not in BCNF

(C) in 2NF, but not in 3NF

(D) not in 2NF

Answer : D

Q. In which normal form an attribute can't hold multiple values : (INFYTQ)

- (A) 1NF
- (B) 2NF
- (C) 3NF
- (D) BCNF

Answer : A

Q. A relation in which every non-key attribute is fully functionally dependent on the primary key and which has no transitive dependencies, is said to be in

_____ [Asked in InfyTQ]

- (A) BCNF
- (B) 1NF
- (C) 2NF
- (D) 3NF

Q. Consider a relation R(A, B, C), which of the following statements is not true according to inference rules for functional dependencies? [Asked in TCS NQT 2021]

- (A) If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
- (B) If $AB \rightarrow C$, then $A \rightarrow B$ and $B \rightarrow C$
- (C) If $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$
- (D) If $A \rightarrow B$, then $AC \rightarrow BC$

Answer : B

Q. Tables in second normal form (2NF): [Asked in L&T InfoTech (LTI) 2021]

- (A) Eliminate all hidden dependencies**
- (B) Eliminate the possibility of insertion anomalies**
- (C) Have a composite key**
- (D) Have all non key fields depend on the whole primary key**

2 NF DECOMPOSITION

Q R(ABCDEFGHIJ)

$AB \rightarrow C$

$AD \rightarrow GH$

$BD \rightarrow EF$

$A \rightarrow I$

$H \rightarrow J$

A B C D E F G H I J

BCNF DECOMPOSITION

Q R(ABCD)

$AB \rightarrow C$

$BC \rightarrow D$

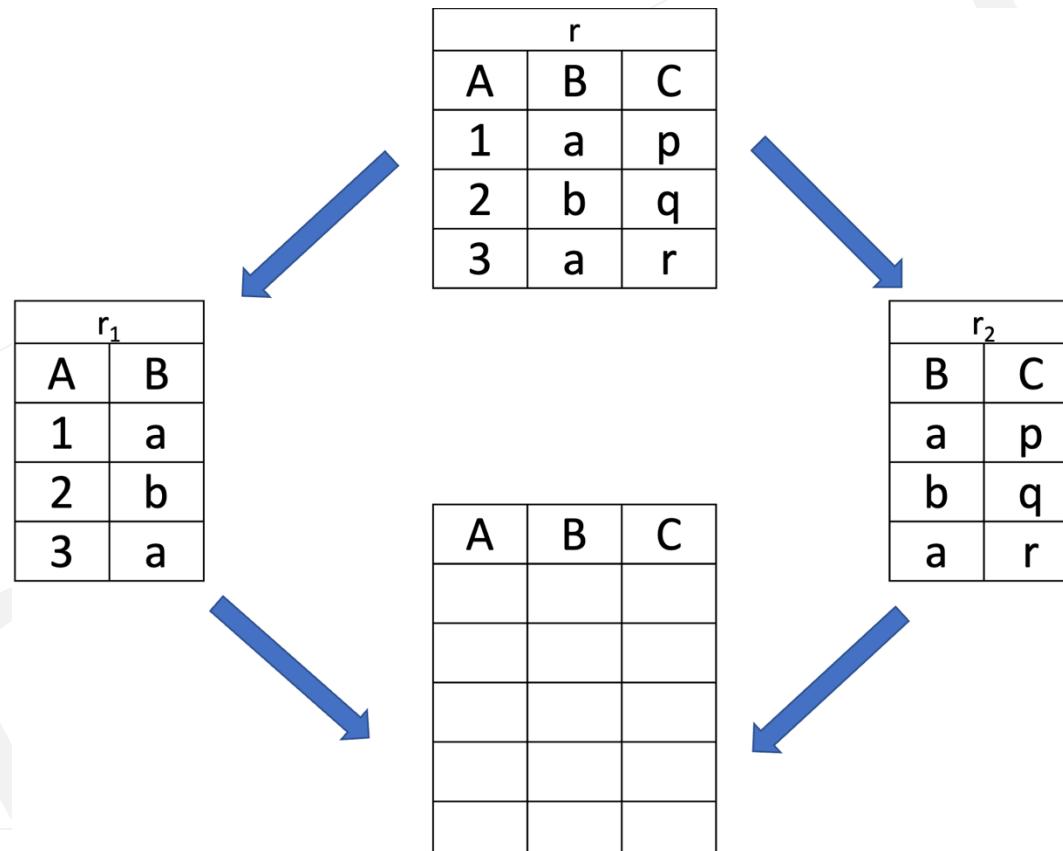
A B C D

Q) Consider the schema $R(S,T,U,V)$ and the dependencies $S \rightarrow T$, $T \rightarrow U$, $U \rightarrow V$, $V \rightarrow S$. Let $R = \{R_1, R_2\}$ such that $R_1 \cap R_2 = \emptyset$. Then the decomposition is :[Amcat].

- (a) Not in 2NF
- (b) In 2NF but not in 3NF
- (c) In 3NF but not in 2NF
- (d) In both 2NF and 3NF

Lossy/Lossless-Dependency Preserving Decomposition

- When a table is decomposed into multiple tables during normalization, it's essential to ensure the decomposition is done properly. The most important property to check is called the **lossless join property** (also called non-additive join decomposition).
- In simple terms, after decomposition, if we later perform a natural join of these smaller tables, we should get exactly the original table back, without gaining or losing any tuples (rows). If after joining, we get additional or fewer tuples than the original, then the decomposition is called **lossy**.



- **Lossy decomposition** occurs if:
 - $R_1 \bowtie R_2$ produces more tuples (extra) than R , or fewer tuples (missing) than R .
 - (Formally, if $R_1 \bowtie R_2 \supset R$ or $R \supset R_1 \bowtie R_2$)
- **Lossless decomposition** occurs if:
 - $R_1 \bowtie R_2 = R$ (exactly same tuples, no more, no less)
- To verify lossless decomposition using functional dependencies, these conditions must hold true:
 - The union of attributes in decomposed relations R_1 and R_2 should exactly cover all attributes of the original relation R .
 - $\text{Att}(R_1) \cup \text{Att}(R_2) = \text{Att}(R)$
 - Intersection (common attributes) between R_1 and R_2 should not be empty.
 - $\text{Att}(R_1) \cap \text{Att}(R_2) \neq \emptyset$
 - The common attribute(s) between R_1 and R_2 must form a candidate key (or at least a key) for at least one of these relations.
 - $\text{Att}(R_1) \cap \text{Att}(R_2) \rightarrow \text{Att}(R_1)$ or $\text{Att}(R_1) \cap \text{Att}(R_2) \rightarrow \text{Att}(R_2)$
- These conditions guarantee the decomposition is lossless and preserve the correctness of database information.

A	B	C	D	E
A	122	1	W	A
E	236	4	X	B
A	199	1	Y	C
B	213	2	Z	D

Q R (A, B, C, D)

AB → CD, D → A

R₁(A, D), R₂(B, C, D)

Q R(ABCDE)(NF) R₁(AB) R₂(BC) R₃(ABCD) R₄(EG)

A → BC

C → DE

D → E

Q R (A,B,C,D) is a relation. Which of the following does not have a lossless join dependency preserving BCNF decomposition? [Asked in CoCubes]

- a) A->B, B->CD
- b) A->B, B->C, C->D
- c) AB->C, C->AD
- d) A->BCD

Answer : D

Q Inst_dept (ID, name, salary, dept name, building, budget) is decomposed into
instructor (ID, name, dept name, salary)
department (dept name, building, budget)

This comes under [Asked in L&T Infotech (LTI)]

- a) Lossy decomposition
- b) Lossless-join decomposition
- c) None
- d) Both

Answer : B

Indexing (Need for Indexing)

- Theoretically, relational databases are based on set theory, where the order of elements doesn't matter. However, practically, when we store data, the order in which records are stored significantly impacts the efficiency of database operations such as searching, insertion, and deletion.
- Records in large files can be stored either in an ordered (sorted) or unordered (unsorted) manner. If records are stored unordered, new records are typically inserted at the end, making insertions and deletions easy, but searches become slow, as only linear search is possible. On the other hand, ordered storage of records enables faster search methods, such as binary search—similar to quickly locating a page number in a book—but makes insertions and deletions costly due to necessary reorganization.
- As databases grow larger, occupying numerous blocks on storage, accessing records becomes slow and inefficient if we depend solely on the direct storage method (ordered or unordered). To address these issues, **indexing** is introduced.

```
-rw-r----- 1 mysql mysql 10M ul 16 20:10 node_field_meta_tags.ibd
-rw-r----- 1 mysql mysql 10M ul 15 22:50 comment_field_data.ibd
-rw-r----- 1 mysql mysql 11M ul 16 20:18 search_total.ibd
-rw-r----- 1 mysql mysql 11M ul 16 20:21 cache_discovery.ibd
-rw-r----- 1 mysql mysql 12M ul 16 20:11 node_field_data.ibd
-rw-r----- 1 mysql mysql 14M ul 16 20:11 url_alias.ibd
-rw-r----- 1 mysql mysql 14M ul 16 20:11 taxonomy_index.ibd
-rw-r----- 1 mysql mysql 15M ul 16 19:27 node_tags.ibd
-rw-r----- 1 mysql mysql 15M ul 15 22:47 comment_comment_body.ibd
-rw-r----- 1 mysql mysql 16M ul 16 19:27 node_revision_tags.ibd
-rw-r----- 1 mysql mysql 19M ul 16 20:23 sessions.ibd
-rw-r----- 1 mysql mysql 22M ul 16 20:18 search_dataset.ibd
-rw-r----- 1 mysql mysql 31M ul 16 20:11 node_body.ibd
-rw-r----- 1 mysql mysql 32M ul 16 20:24 watchdog.ibd
-rw-r----- 1 mysql mysql 36M ul 16 20:11 node_revision_body.ibd
-rw-r----- 1 mysql mysql 108M ul 16 20:18 search_index.ibd
-rw-r----- 1 mysql mysql 120M ul 16 20:21 cache_entity.ibd
-rw-r----- 1 mysql mysql 268M ul 16 20:25 cache_data.ibd
-rw-r----- 1 mysql mysql 1.6G ul 16 20:25 cache_dynamic_page_cache.ibd
-rw-r----- 1 mysql mysql 3.4G ul 16 20:25 cache_render.ibd
```

- Indexing creates additional data structures (indexes) based on specific key attributes, allowing rapid data retrieval without scanning the entire database. This significantly enhances the speed and efficiency of database operations, particularly searches, while balancing the costs of insertion and deletion. Indexing in database systems is similar to what we see in books.

INDEX

ABC, 164, 321n
academic journals, 262, 280–82
Adobe eBook Reader, 148–53
advertising, 36, 45–46, 127, 145–46, 167–68, 321n
Africa, medications for HIV patients in, 257–61
Agee, Michael, 223–24, 225
agricultural patents, 313n
Aibo robotic dog, 153–55, 156, 157, 160
AIDS medications, 257–60
air traffic, land ownership vs., 1–3
Akerlof, George, 232
Alben, Alex, 100–104, 105, 198–99, 295, 317n
alcohol prohibition, 200
Alice's Adventures in Wonderland (Carroll), 152–53
Anello, Douglas, 60
animated cartoons, 21–24
antiretroviral drugs, 257–61
Apple Corporation, 203, 264, 302
architecture, constraint effected through, 122, 123, 124, 318n
archive.org, 112
 see also Internet Archive
archives, digital, 108–15, 173, 222, 226–27
Aristotle, 150
Armstrong, Edwin Howard, 3–6, 184, 196
Arrow, Kenneth, 232
art, underground, 186
artists:
 publicity rights on images of, 317n
 recording industry payments to, 52, 58–59, 74, 195, 196–97, 199, 301, 329n–30n

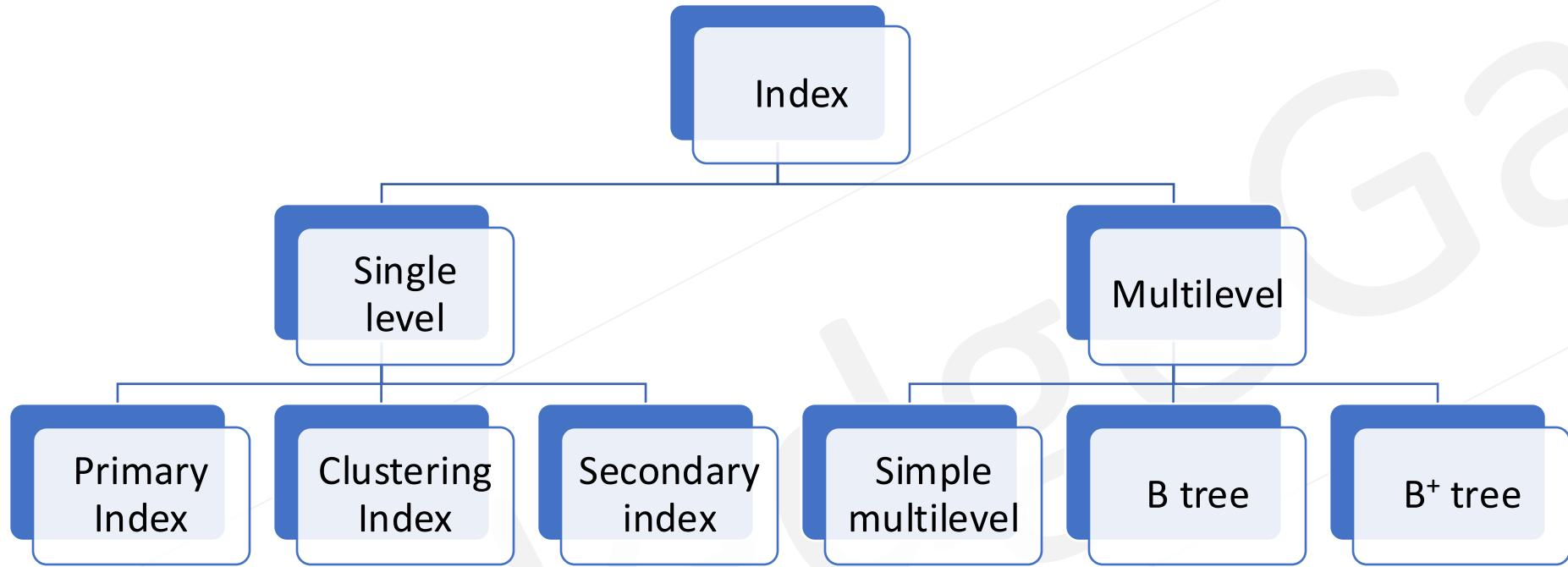
- An index provides a secondary access path, giving us an alternate, faster way to access records without affecting how records are physically stored in the main file.
- Typically, the size of an index file is much smaller compared to the main file because the index file stores only two pieces of information: the search key (the attribute used for searching) and the block pointer (the address of the block in the main file containing that record). In contrast, the main file stores complete records with all attributes.
- Indexes can be created on any attribute of the table, including both primary keys and non-key attributes. Additionally, multiple indexes can be designed for the same main file, each index based on a different attribute.
- Index files are always stored in sorted order (ordered), regardless of whether the main file is ordered or unordered. This ordered structure allows efficient searching methods, such as binary search.
- Indexing significantly reduces search time, making queries faster, but introduces extra storage overhead due to the additional space required by the index files.
- The number of accesses needed to search and locate the correct block in the main file using an index is approximately $\log_2(\text{number of blocks in the index file}) + 1$.



Q Data which improves the performance and accessibility of the database are called:

- (a)** Indexes
- (b)** User Data
- (c)** Application Metadata
- (d)** Data Dictionary

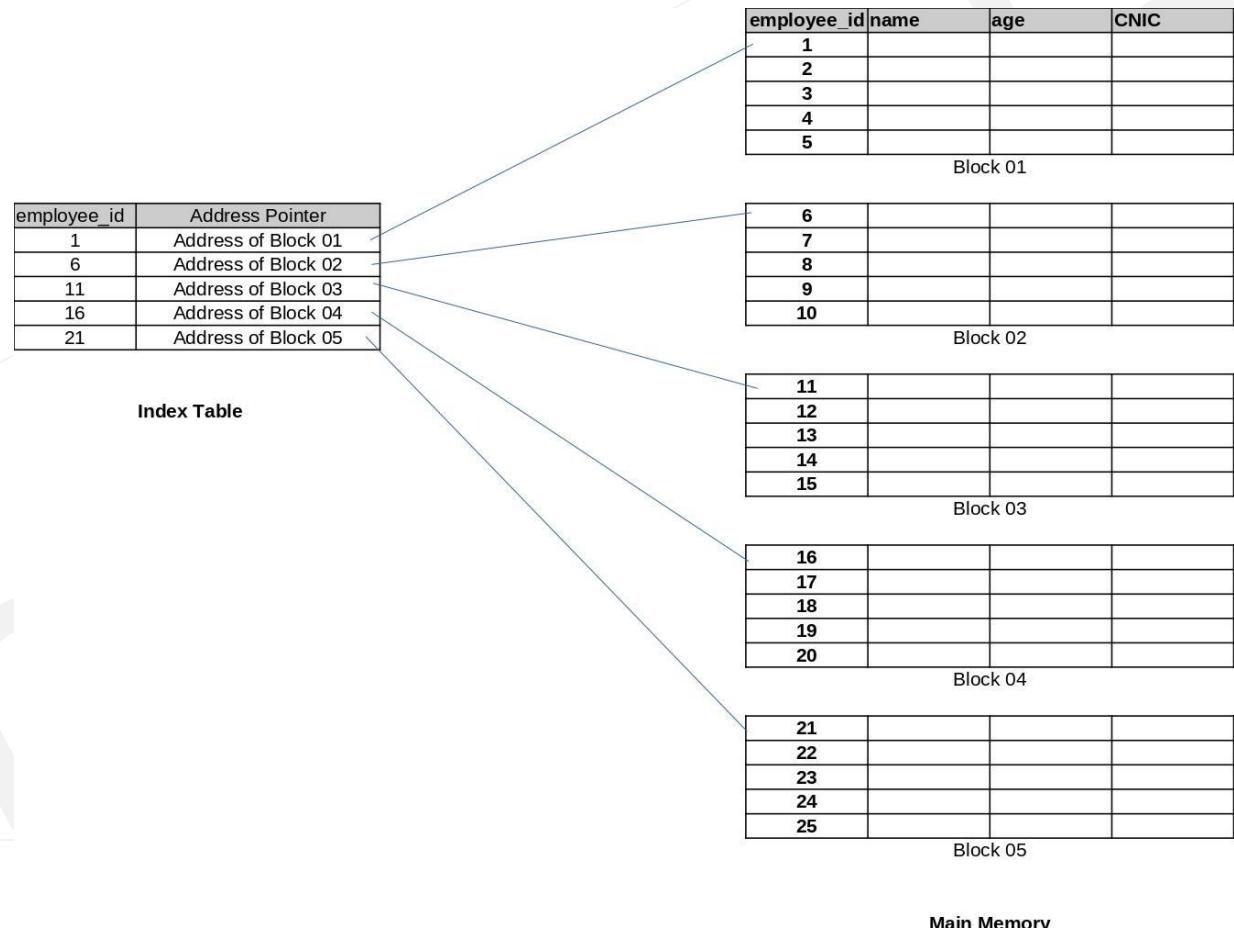
TYPES OF INDEXING



- Single level index means we create index file for the main file, and then stop the process.
- Multiple level index means, we further index the index file and keep repeating the process until we get one block.

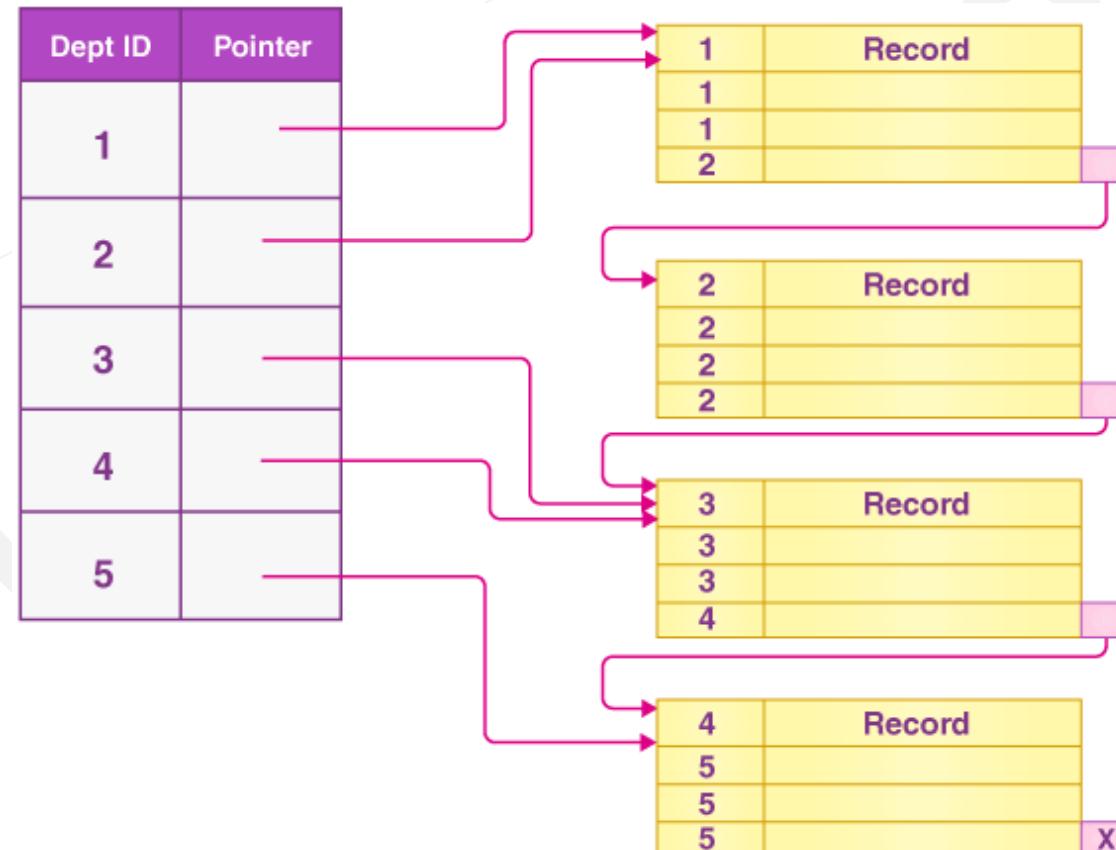
PRIMARY INDEXING

- Main file is always sorted according to primary key.
- Indexing is done on Primary Key, therefore called as primary indexing
- Index file have two columns, first primary key and second anchor pointer (base address of block)
- It is an example of Sparse Indexing.
- Here first record (anchor record) of every block gets an entry in the index file
- No. of entries in the index file = No of blocks acquired by the main file.



CLUSTERED INDEXING

- Main file will be ordered on some non-key attributes
- No of entries in the index file = no of unique values of the attribute on which indexing is done.
- It is the example of Sparse as well as dense indexing



Q A clustering index is created when _____.

- (A) primary key is declared and ordered
- (B) no key ordered
- (C) foreign key ordered
- (D) there is no key and no order

Q An index is clustered, if

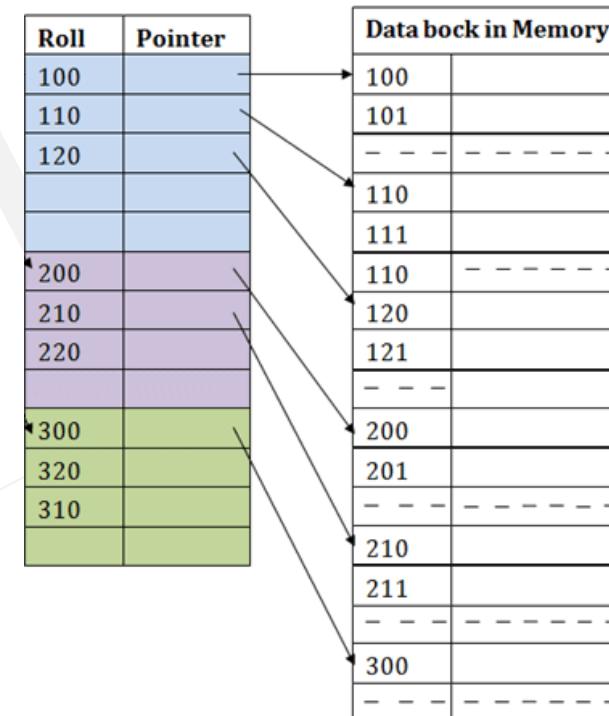
- (a)** it is on a set of fields that form a candidate key
- (b)** it is on a set of fields that include the primary key
- (c)** the data records of the file are organized in the same order as the data entries of the index
- (d)** the data records of the file are organized not in the same order as the data entries of the index

Q A clustering index is defined on the fields which are of type **(GATE-2008) (1 Marks)**

- a)** non-key and ordering
- b)** non-key and non-ordering
- c)** key and ordering
- d)** key and non-ordering

SECONDARY INDEXING

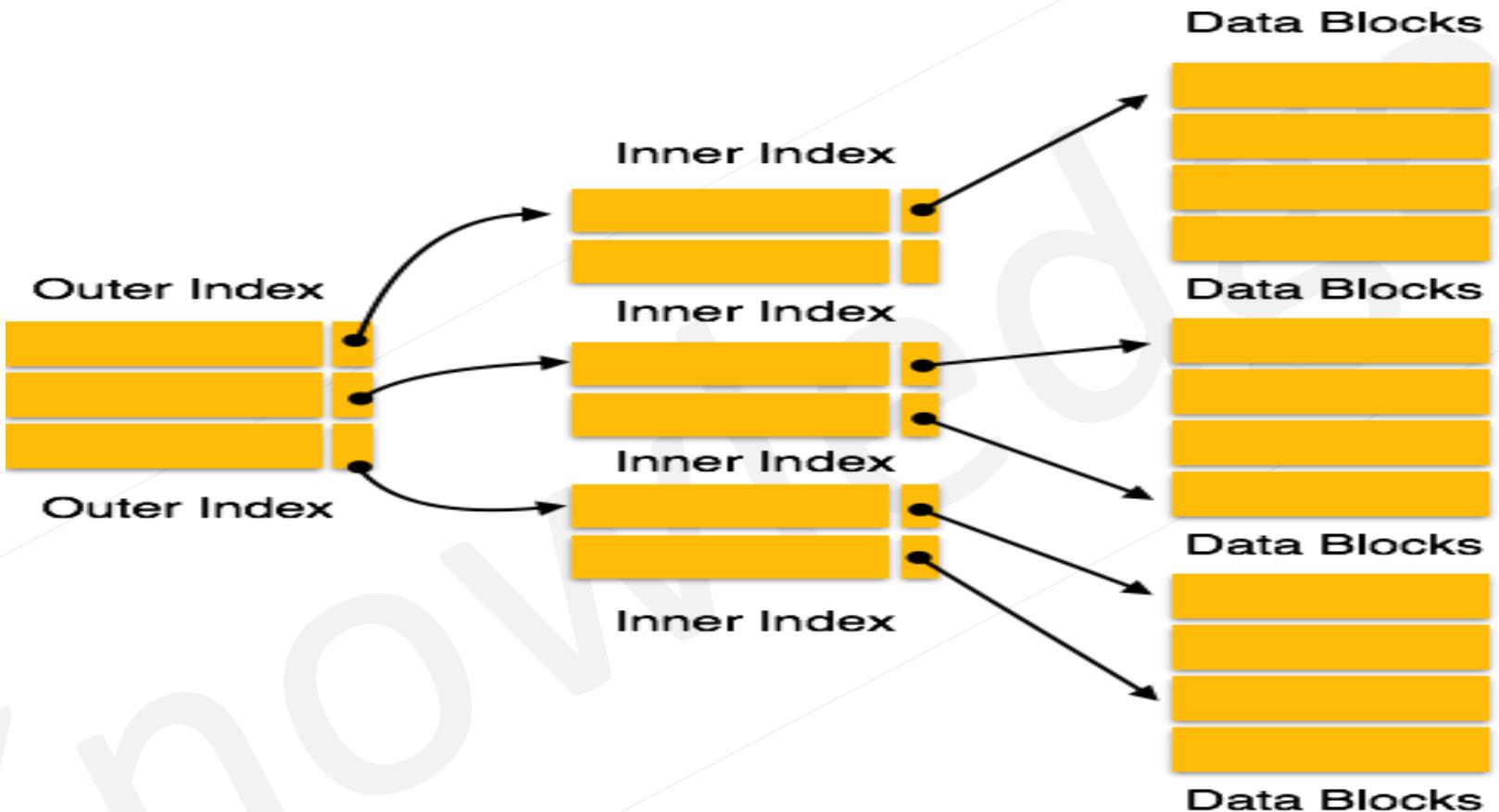
- Secondary indexing is used when we already have a primary index based on a primary key, but there are frequent queries involving other attributes as well. To speed up these queries, we create additional indexes called secondary indexes.
- Unlike primary indexing, the main file does **not** need to be ordered based on the attribute used for secondary indexing—it can be unordered. Secondary indexes can be created on both key and non-key attributes.
- In secondary indexing, the number of entries in the index file typically matches the total number of records in the main file. Thus, secondary indexing is usually an example of dense indexing, having an entry for every single record in the main file.



- Indexing can be classified based on various criteria. One important criterion is how frequently index entries are created, leading to two types: Dense Index and Sparse Index. Dense and sparse indexes are not mutually exclusive or complementary.
 - A Dense Index contains an index entry for every single search key value present in the main file. This means each distinct value used for searching is listed in the index file. Dense indexing typically allows very quick data retrieval because each search key is explicitly mapped, but it requires more storage space due to a higher number of entries. It's important to understand that a dense index has an entry for each distinct search key value, not necessarily each individual record. Therefore, if a search key is repeated, the dense index might still contain fewer entries than the total number of records.
 - In contrast, a Sparse Index contains index entries only for some selected records, usually at regular intervals. As a result, the sparse index file always has fewer entries than the number of records in the main file, which means it takes up less space. Searching with a sparse index can be slightly slower compared to dense indexing since some additional steps might be needed to locate a specific record.
 - It's worth noting clearly that Dense and Sparse Indexing approaches are not necessarily complementary or mutually exclusive. Depending on the search key used and practical requirements, both indexing methods might coexist within the same database system. Sparse indexing is more suitable for ordered files, where it's easy to locate records between two index entries, whereas dense indexing works effectively with both ordered and unordered files.

MULTILEVEL INDEXING

- Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block-0, which can easily be accommodated anywhere in the main memory.



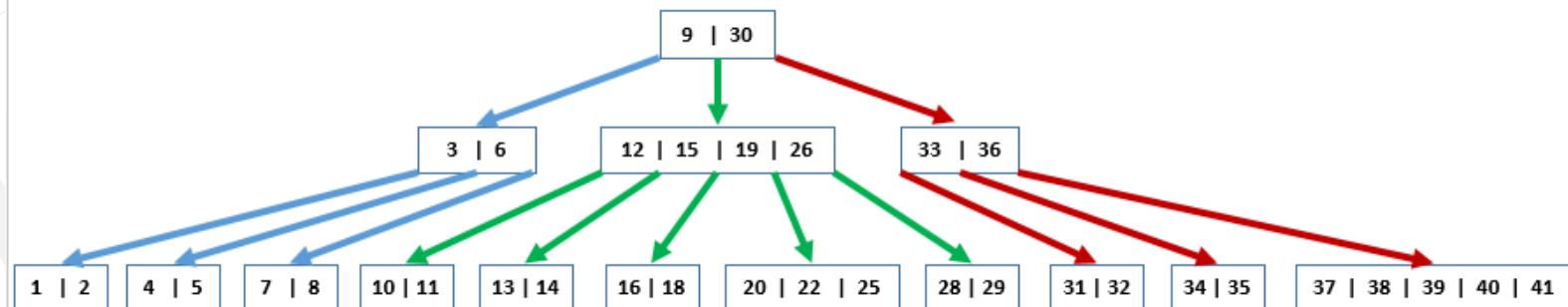
Reason to have B tree and B+ tree

- After studying indexing in detail, we understand that index files are always sorted and frequently searched. As databases grow larger, the index files themselves become so large that we sometimes need to index these indexes, known as multi-level indexing. Thus, we require a suitable data structure to efficiently support searching, insertion, and deletion operations, all while maintaining sorted order.
- Although many data structures exist—like arrays, linked lists, stacks, queues, and binary trees—each has limitations in efficiently supporting frequent insertions, deletions, and quick searches simultaneously, especially for large, sorted datasets. This led to the development of specialized tree structures called B-trees and B+ trees, specifically designed for indexing large database files.
- B-trees and B+ trees are dynamic and can efficiently handle an increasing or decreasing number of records. They inherently support multi-level indexing because their height remains minimal even as data grows, ensuring very fast searches. These trees are self-balancing, maintaining perfect balance automatically, which significantly improves database search and update performance.

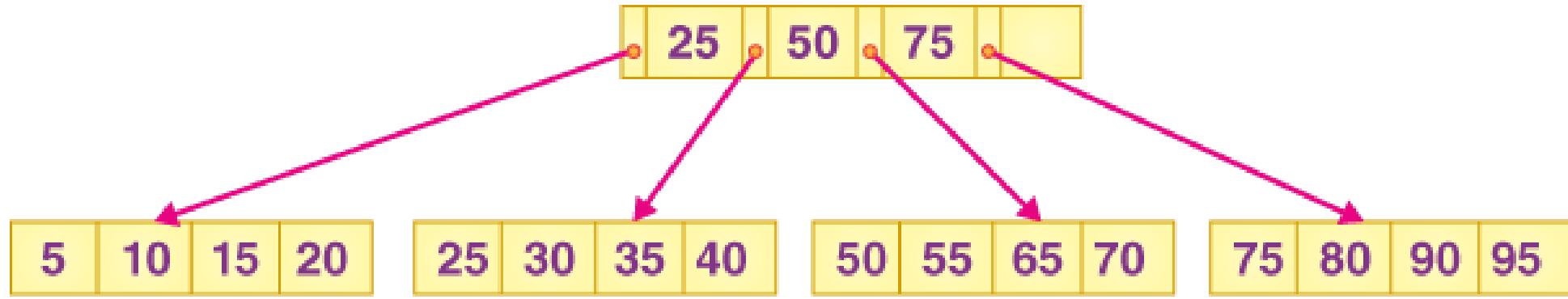
B tree

- A B-tree of order m if non-empty is an m-way search tree in which.
 - The root has at least zero child nodes and at most m child nodes.
 - The internal nodes except the root have at least $\text{ceiling}(m/2)$ child nodes and at most m child nodes.
 - The number of keys in each internal node is one less than the number of child nodes and these keys partition the subtrees of the nodes in a manner similar to that of m-way search tree.
 - All leaf nodes are on the same level(perfectly balanced).
 - Very less internal fragmentation, memory utilization is very good. Less number of nodes(blocks) are used and height is also optimized, so access will be very fast. Difficulty of traversing the key sequentially. Means B-TREE do not hold good for range-based queries of database.

Root			Internal except Root			Leaf		
Rules	MAX	MIN	Rules	MAX	MIN	Rules	MAX	MIN
CHILD	m	0	CHILD	m	$[m/2]$	CHILD	0	0
DATA	$m-1$	1	DATA	$m-1$	$[m/2] - 1$	DATA	$m-1$	$[m/2] - 1$



B+ tree



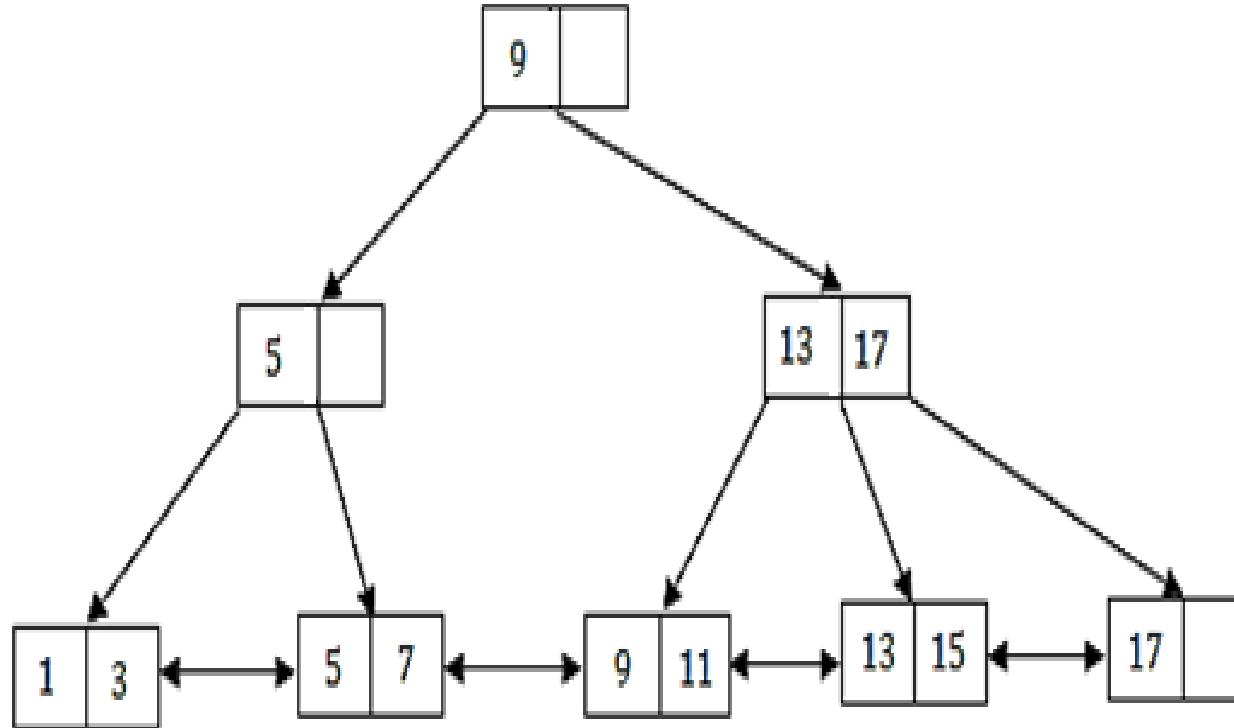
Q Which one of the following statements is NOT correct about the B⁺ tree data structure used for creating an index of a relational database table?

- (a)** Each leaf node has a pointer to the next leaf node
- (b)** Non-leaf nodes have pointers to data records
- (c)** B+ Tree is a height-balanced tree
- (d)** Key values in each node are kept in sorted order

Q B⁺ Trees are considered **BALANCED** because

- a)** the lengths of the paths from the root to all leaf nodes are all equal
- b)** the lengths of the paths from the root to all leaf nodes differ from each other by at most 1
- c)** the number of children of any two non-leaf sibling nodes differ by at most 1
- d)** the number of records in any two leaf nodes differ by at most 1

With reference to the B⁺ tree index of order 1 shown below, the minimum number of nodes (including the Root node) that must be fetched in order to satisfy the following query: "Get all records with a search key greater than or equal to 7 and less than 15" is _____.



Q) The primary indexes, secondary indexes and cluster index are all types of.[Amcat].

- (a) Ordered indexes**
- (b) Unordered indexed**
- (c) Linear indexes**
- (d) Relative search indexes**

Q) In case the indices values are larger, index is created for these values of the index. This is called.[**Amcat**].

- (a) Pointed index
- (b) Sequential index
- (c) Multilevel index
- (d) Multiple index

Q) Which one is true about clustered index.[Amcat].

- (a) Clustered index is not associated with table
- (b) Clustered index is built by default on unique key columns
- (c) Clustered index is not built on unique key columns
- (d) None of the mentioned

Q) Which of the following is true for B+ trees:[Hexaware]

- (a)** B+ trees are for storing data on disk.
- (b)** On B+ trees range queries are faster.
- (c)** B+ trees are for secondary indexes.
- (d)** B+ trees are for primary indexes.

Ans=B.

Q. Which of the following statements is FALSE with respect to B-tree and B+ trees? [Asked in InfyTQ]

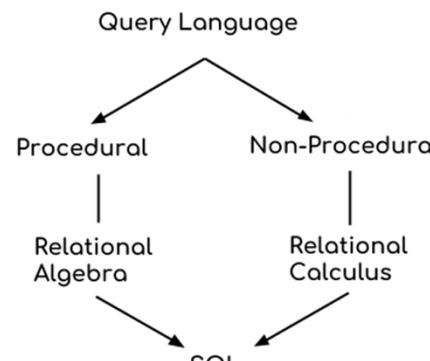
- (A) Deletion operation is easier in B-tree but complex in the case of B+ trees.**
- (B) In B+ trees, data records are stored only in the leaf nodes but in B trees data records are stored both in leaf and internal nodes.**
- (C) Search keys are repeated in the case of B+ trees but not in the case of B trees.**
- (D) Searching is faster in B+ trees compared to B trees**

Answer : A

Query Language

- After designing a database—starting with ER diagrams, converting them into relational models, normalizing, and indexing—the next important task is how to efficiently store, retrieve, and modify data. For interacting with databases, we use special languages called query languages. Although they can perform multiple operations, our main focus will be on data retrieval.
- Query languages, also known as Data Query Languages (DQLs), are computer languages that allow users to request information from a database. The most widely-used example of such a language is Structured Query Language (SQL).
- There are two main types of query languages:
 - **Procedural Query Languages:** In procedural query languages, users explicitly instruct the system on “what” data to retrieve and also “how” exactly to retrieve it. Relational Algebra is a key example of a procedural language, where the user specifies step-by-step operations to obtain the required result.
 - **Non-Procedural (Declarative) Query Languages:** In non-procedural languages, users specify only “what” data they want, without mentioning the exact steps or methods for retrieving it. Relational Calculus (including Tuple Relational Calculus and Domain Relational Calculus) is an example of a declarative query language based on mathematical logic.
- Relational Algebra and Relational Calculus provide fundamental mathematical foundations for querying relational databases. They are theoretical and not directly executed by computer systems, but SQL is practically built upon these foundations.
- Structured Query Language (SQL), which operates on RDBMS, combines both procedural and non-procedural features, making it powerful and flexible for practical database operations.

ER diagram → relational model → Normalization → Indexing → ?



- [Atomese](#), the graph query language for the [OpenCog](#) graph database, the [AtomSpace](#).
- [Attempto Controlled English](#) is a query language that is also a [controlled natural language](#).^[1]
- [AQL](#) is a query language for the [ArangoDB](#) native multi-model database system.
- [.QL](#) is a proprietary object-oriented query language for querying [relational databases](#); successor of Datalog;
- [Contextual Query Language](#) (CQL) a formal language for representing queries to [information retrieval](#) systems such as web indexes or bibliographic catalogues.
- [CQLF](#) (CODYASYL Query Language, Flat) is a query language for [CODASYL](#)-type databases;
- [Concept-Oriented Query Language](#) (COQL) is used in the concept-oriented model (COM). It is based on a novel [data modeling](#) construct, concept, and uses such operations as projection and de-projection for multi-dimensional analysis, analytical operations and inference;
- [Cypher](#) is a query language for the [Neo4j](#) graph database;
- [DMX](#) is a query language for [data mining](#) models;
- [Datalog](#) is a query language for [deductive databases](#);
- [Discovery Query Language](#) is a query language for accessing Watson Discovery Services on IBM Cloud;^[2]
- [F-logic](#) is a declarative object-oriented language for [deductive databases](#) and [knowledge representation](#).
- [FQL](#) enables you to use a [SQL](#)-style interface to query the data exposed by the [Graph API](#). It provides advanced features not available in the [Graph API](#).^[3]
- [Gellish English](#) is a language that can be used for queries in Gellish English Databases, for dialogues (requests and responses) as well as for information modeling and knowledge modeling;^[4]
- [Gremlin](#) is an [Apache Software Foundation](#) graph traversal language for OLTP and OLAP graph systems.
- [GraphQL](#) is a data query language developed by [Facebook](#) as an alternate to [REST](#) and ad-hoc [webservice](#) architectures.
- [HTSQL](#) is a query language that translates [HTTP](#) queries to [SQL](#);
- [ISBL](#) is a query language for [PRTV](#), one of the earliest relational database management systems;
- [Jaql](#) is a functional data processing and query language most commonly used for JSON query processing;
- [JSONiq](#) is a declarative query language designed for collections of [JSON](#) documents;
- [KQL](#) is a query language used in [Azure Data Explorer \(Kusto\)](#) and the CMPivot tool in Microsoft System Center Configuration Manager
- [LINQ](#) query-expressions is a way to query various data sources from [.NET](#) languages
- [LDAP](#) is an application protocol for querying and modifying directory services running over TCP/IP;
- LogiQL is a variant of Datalog and is the query language for the LogicBlox system.

- LINQ query-expressions is a way to query various data sources from .NET languages
- LDAP is an application protocol for querying and modifying directory services running over TCP/IP;
- LogiQL is a variant of Datalog and is the query language for the LogicBlox system.
- MQL is a cheminformatics query language for a substructure search allowing beside nominal properties also numerical properties;
- MDX is a query language for OLAP databases;
- N1QL is a Couchbase's query language finding data in Couchbase Servers;
- OQL is Object Query Language;
- OCL (Object Constraint Language). Despite its name, OCL is also an object query language and an OMG standard;
- OPPath, intended for use in querying WinFS Stores;
- OttoQL, intended for querying tables, XML, and databases;
- Poliqarp Query Language is a special query language designed to analyze annotated text. Used in the Poliqarp search engine;
- PQL is a special-purpose programming language for managing process models based on information about scenarios that these models describe;
- PTQL based on relational queries over program traces, allowing programmers to write expressive, declarative queries about program behavior.
- QUEL is a relational database access language, similar in most ways to SQL;
- RDQL is a RDF query language;
- Rego is a query language inspired by Datalog;
- ReQL is a query language used in RethinkDB;
- SMARTS is the cheminformatics standard for a substructure search;
- SPARQL is a query language for RDF graphs;
- SPL is a search language for machine-generated big data, based upon Unix Piping and SQL.
- SCL is the Software Control Language to query and manipulate Endevor objects
- SQL is a well known query language and data manipulation language for relational databases;
- SuprTool is a proprietary query language for SuprTool, a database access program used for accessing data in /Image/SQL (formerly TurboIMAGE) and Oracle databases;
- TMQL Topic Map Query Language is a query language for Topic Maps;
- TQL is a language used to query topology for HP products
- Tutorial D is a query language for truly relational database management systems (TRDBMS);
- U-SQL is a data processing language invented at Microsoft
- XQuery is a query language for XML data sources;
- XPath is a declarative language for navigating XML documents;
- XSPARQL is an integrated query language combining XQuery with SPARQL to query both XML and RDF data sources at once;
- YQL is an SQL-like query language created by Yahoo!
- Search engine query languages, e.g., as used by Google^[5] or Bing^[6]

Q) What is the language used by most of the DBMSs for helping their users to access data?[TCS NINJA]

- (a) High level language**
- (b) Query language**
- (c) SQL**
- (d) 4GL**
- (e) None of the above**

Ans=B.

Introduction to SQL

- Structured Query Language (SQL) is currently the most popular database query language used in relational database management systems (RDBMS). While we often refer to SQL as just a “query language,” it can actually perform many additional tasks. These include defining database structures, modifying data, managing security constraints, and handling various database-related operations.
- SQL was originally developed by IBM as “SEQUEL” (Structured English Query Language) during the System R research project in the early 1970s. Later, due to trademark reasons, its name changed from SEQUEL to SQL. Over time, SQL became the widely accepted standard language for relational databases.
- The SQL language combines concepts from both relational algebra (procedural) and relational calculus (non-procedural), giving it both flexibility and power. Its standards have evolved through multiple revisions, starting with SQL-86, and continuing with updates such as SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008, SQL:2011, SQL:2016, SQL:2016, and most recently, SQL:2023.

Parts of SQL

- Data Definition Language (DDL): This includes commands used for defining database structures like tables, modifying or deleting these structures, and specifying constraints on data (e.g., NOT NULL constraint for primary keys). It also includes commands to define database views and control authorization (user permissions, like granting read/write access).
- Data Manipulation Language (DML): DML consists of commands used for retrieving (querying) data, as well as inserting, updating, or deleting records within a database.
- Transaction Control Language (TCL): TCL includes commands that manage database transactions, ensuring data integrity during operations (e.g., COMMIT, ROLLBACK, SAVEPOINT).
- Embedded and Dynamic SQL: This part specifies how SQL commands can be integrated (embedded) into general-purpose programming languages such as C, C++, and Java, enabling programmers to use SQL commands directly in their applications.

Q Which one is correct?

- (a)** DML includes a query language based on both relation algebra and tuple calculus
- (b)** DML includes a query language based on tuple calculus
- (c)** DML includes a query language based on relational algebra
- (d)** DML includes a query language based on none of the relational algebra and tuple calculus

Q) The language associated with a database management system that is employed by end users and programmers to manipulate data in the database is the: [Amcat].

- (a) Data definition language.**
- (b) Data presentation language.**
- (c) Data manipulation language.**
- (d) Data translation language**

Ans=C.

Q DBMS provides the facility of accessing data from a database through?

(a) DDL

(b) DML

(c) DBA

(d) Schema

Q Which of the following is/are correct?

- (a)** An SQL query automatically eliminates duplicates
- (b)** An SQL query will not work if there are no indexes on the relations
- (c)** SQL permits attribute names to be repeated in the same relation
- (d)** None of the above

Q Which is the subset of SQL commands used to manipulate Oracle Database structures, including tables? [Asked in TCS NQT]

- a)** Data Definition Language(DDL)

- b)** Data Manipulation Language(DML)

- c)** Both of above

- d)** None

Q What is a view? [Asked in E-Litmus]

- A)** A view is a special stored procedure executed when certain event occurs.
- B)** A view is a virtual table which results of executing a pre-compiled query. A view is not part of the physical database schema, while the regular tables are.
- C)** A view is a database diagram.
- D)** None of these

Q Which of the following statement is correct regarding the difference between TRUNCATE, DELETE and DROP command? [Asked in Infosys]

- I. DELETE operation can be rolled back but TRUNCATE and DROP operations cannot be rolled back.
- II. TRUNCATE and DROP operations can be rolled back but DELETE operations cannot be rolled back.
- III. DELETE is an example of DML, but TRUNCATE and DROP are examples of DDL.
- IV. All are an example of DDL.

- A) I and III
- B) II and III
- C) II and IV
- D) II and IV

Q) Which of the following is generally used for performing tasks like creating the structure of the relations, deleting relation? [Tcs Ninja].

- (a) DML(Data Manipulation Language)**
- (b) Query**
- (c) Relational Schema**
- (d) DDL(Data Definition Language)**

Q) ROLLBACK in a database is _____ statement.[Wipro].

- (a) DDL
- (b) DML
- (c) DCL
- (d) TCL

Ans=D.

Q) Three DDL commands: [Amcat]

(a) CREATE, ALTER, DELETE

(b) INSERT, UPDATE, DELETE

(c) CREATE, UPDATE, DROP

(d) CREATE, ALTER, DROP

Ans=A.

Q) Major Language of databases are _____. [Amcat]

- (a)** D DL, DQL, DQL, DCL, TCL
- (b)** DLL, DML, DQL, DCL, TCL.
- (c)** DDL, DML, DQL, DCL, TCL.
- (d)** DDL, DML, DQL, DCL, TQL.

Ans=C.

Q) DQL uses select command which is used to.[Amcat].

- (a)** Retrieve data from database.
- (b)** Used to define database structure.
- (c)** Create schema.
- (d)** None of these

Basic Structure of SQL Queries

- Every SQL query reads at least **one relation (table)** listed in the FROM clause.
- It always returns **one relation** as its result; by default this result has no name, but its columns keep the names they had in the input tables.
- **Mandatory and Optional Clauses** WHERE <condition> is optional. If you omit it, the condition is treated as **true** (all rows qualify).

Select A₁, A₂,..., A_n
from r₁, r₂,..., r_m
Where P;

(Column name)
(Relation/table name)
(Condition)

- **Case-Insensitivity**
 - SQL keywords are **not case-sensitive** (select, SELECT, and SeLeCt are equivalent).
- **Duplicates in Results**
 - In the pure relational model, a relation is a **set** (no duplicates).
 - SQL keeps duplicates by default because removing them can be slow.
 - Add DISTINCT right after SELECT to drop duplicates when needed.
 - The keyword ALL can be written after SELECT to keep duplicates explicitly, but it is unnecessary because this is already the default.

Select Clause

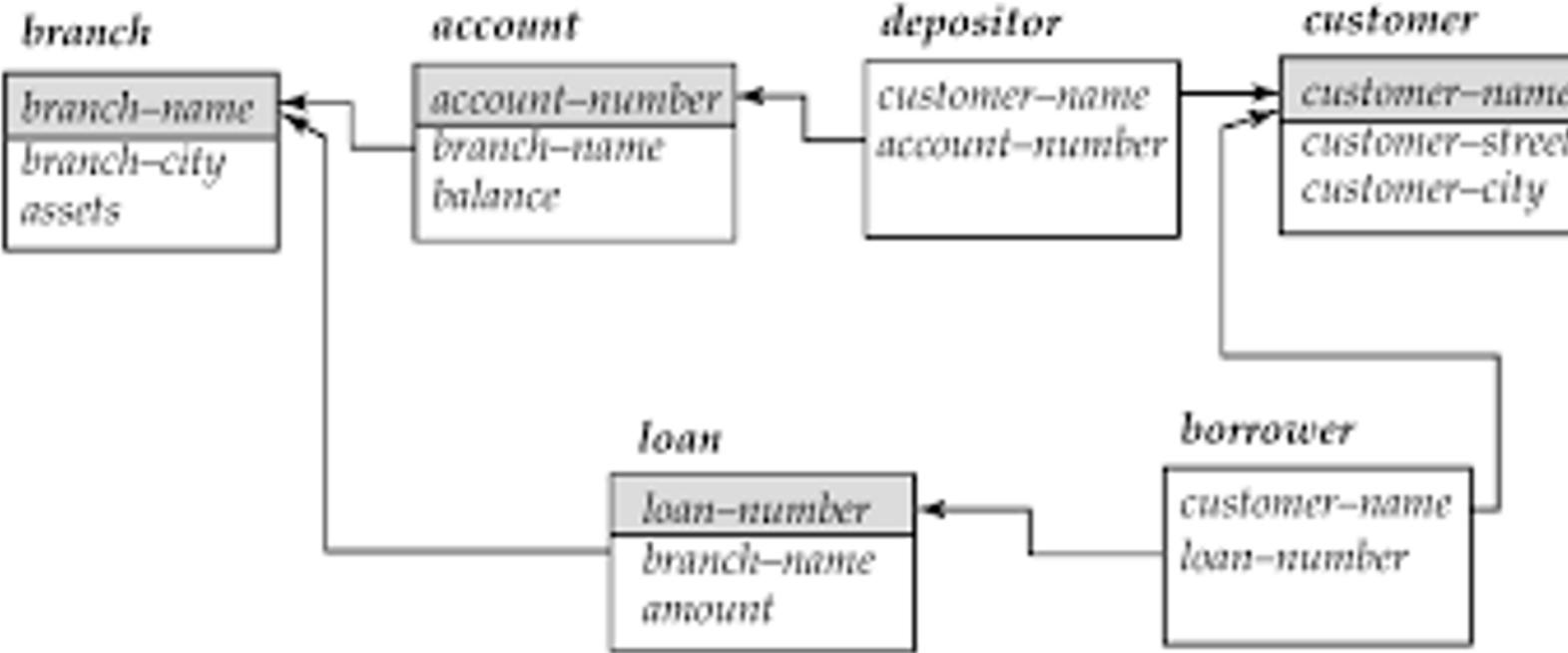
- **Purpose** – SELECT chooses which columns appear in the result, just like projection (π) in relational algebra; this is vertical filtering of a table.
- **Column order** – Columns are returned in the same order they are listed after SELECT.
- **Always required** – SQL queries must start with SELECT, even if you need every column; in relational algebra projection is optional, but SQL keeps syntax consistent.
- **All columns quickly** – Write SELECT * when you need every column without listing them one-by-one.
- **Expressions allowed** – You may include arithmetic expressions (price * 0.9, marks + 5) alongside column names; these calculate values for display only and do not alter the stored data.
 - Select A₁, A₂,..., A_n (Column name)

Q Write a SQL query to find all the details of bank branches?

Q Write a SQL query to find each loan number along with loan amount?

Q Write a SQL query to find the name of all customer without duplication having bank account?

Q Write a SQL query to find all account_no and balance with 6% yearly interest added to it?



Q Given SQL statement is example of: [Asked in Hexaware 2018]

select * from table_name

- a) Data Definition Language
- b) Data Manipulation Language
- c) Data Control Language
- d) Transaction Control Language

Q In SQL, which command is used to SELECT only one copy of each set of duplicable rows [Asked in Accenture 2020]

- A) SELECT DISTINCT
- B) SELECT UNIQUE
- C) SELECT DIFFERENT
- D) All of the above

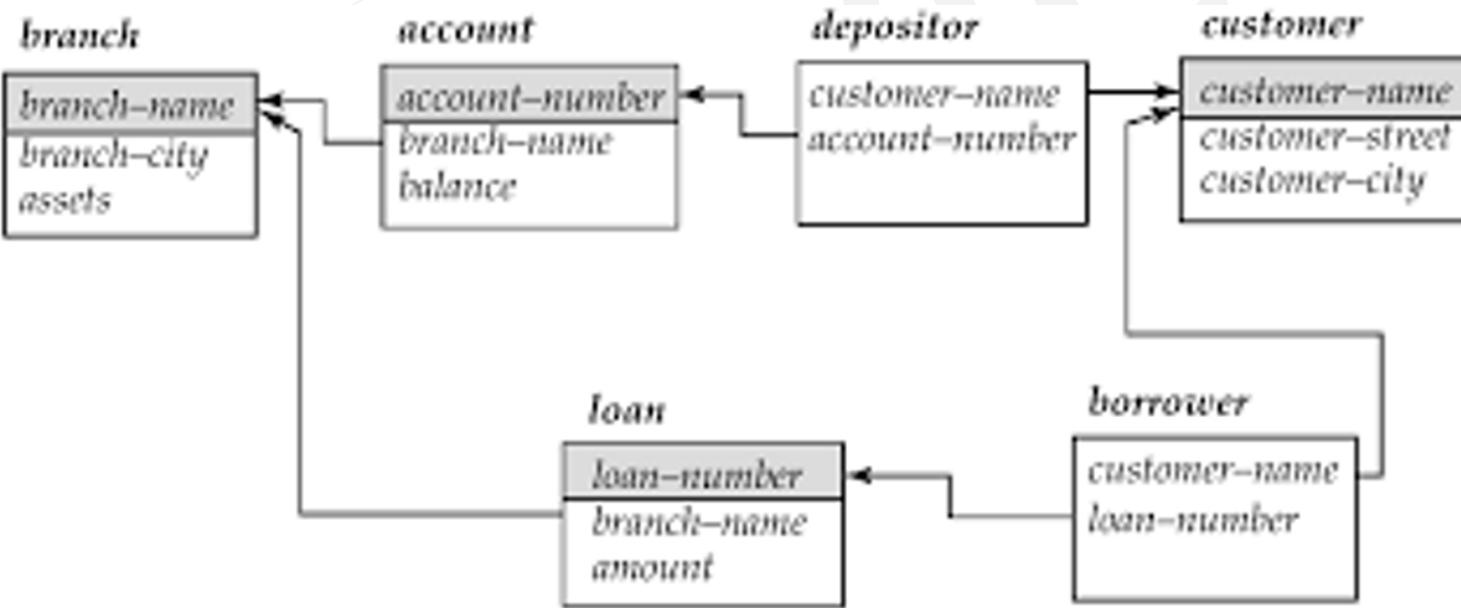
Select Clause with where clause

- Where clause in SQL is same as ‘ σ ’ sigma of relational algebra where we specify the conditions/Predicate (horizontal filtering)
- Where clause can have expressions involving the comparison operators $<$, $<$, $>$, \geq , \leq and \neq . SQL allows us to use the comparison operators to compare strings and arithmetic expressions.
- SQL allows the use of the logical connectives **and**, **or**, and **not** in the **where** clause.
- SQL includes a **between** comparison operator to simplify **where** clauses that specify that a value be less than or equal to some value and greater than or equal to some other value.
- Similarly, we can use the **not between** comparison operator.

Q Write a SQL query to find all account_no where balance is less than 1000?

Q Write a SQL query to find branch name which is situated in Delhi and having assets less than 1,00,000?

Q Write a SQL query to find branch name and account_no which has balance greater than equal to 1,000 but less than equal to 10,000?



Q Consider the following three SQL queries (Assume the data in the people table):

- (a) Select Name from people where Age > 21;
- (b) Select Name from people where Height > 180;
- (c) Select Name from people where (Age > 21) or (Height > 180);

If the SQL queries (a) and (b) above, return 10 rows and 7 rows in the result set respectively, then what is one possible number of rows returned by the SQL query ?

(a) 3

(b) 7

(c) 10

(d) 21

Q What is the output of the below query : [Asked in Infosys]

```
Select NAME  
from Student  
where BRANCH <> 'CSE'
```

- a)** find names of all students in CSE branch
- b)** find names of all students in all branches
- c)** find names of all students except of CSE branch
- d)** none of the above

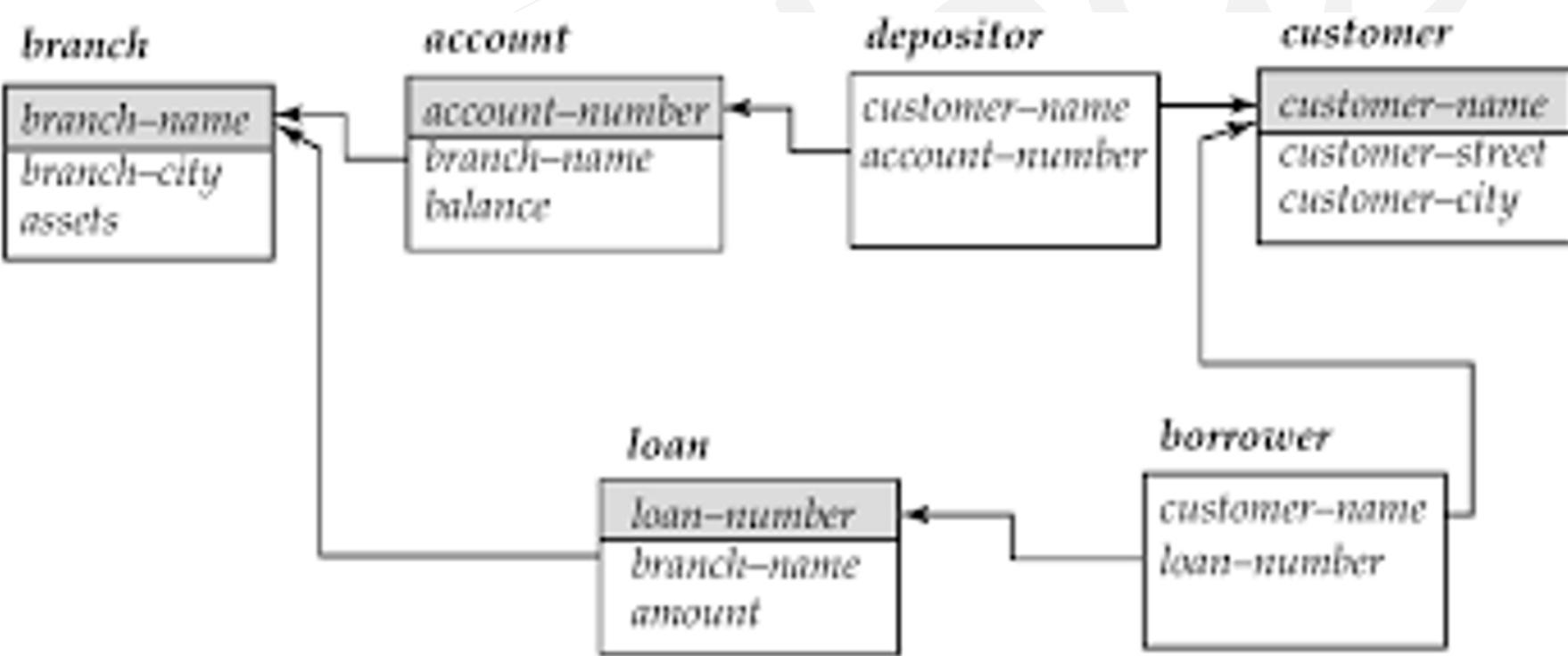
- Q** Find the cities name with the condition and temperature from table 'whether' where condition = sunny or cloudy but temperature ≥ 60 . [Asked in TCS NQT]
- A) SELECT city, temperature, condition FROM weather WHERE condition = 'cloudy' AND condition = 'sunny' OR temperature ≥ 60
- B) SELECT city, temperature, condition FROM weather WHERE condition = 'cloudy' OR condition = 'sunny' OR temperature ≥ 60
- C) SELECT city, temperature, condition FROM weather WHERE condition = 'sunny' OR condition = 'cloudy' AND temperature ≥ 60
- D) SELECT city, temperature, condition FROM weather WHERE condition = 'sunny' AND condition = 'cloudy' AND temperature ≥ 60

Set Operation

- The SQL operations **union**, **intersect**, and **except/minus** operate on relations and corresponds to the mathematical set-theory operations \cup , \cap and $-$ respectively.
- The **union** operation automatically eliminates duplicates, unlike the **select** clause, If we want to retain all duplicates, we must write **union all** in place of **union**.
- The **intersect** operation automatically eliminates duplicates. If we want to retain all duplicates, we must write **intersect all** in place of **intersect**.
- If we want to retain duplicates, we must write **except all** in place of **except**.

Q Write a SQL query to find all the customer name

- a) who have a loan or an account or both ?
- b) who have both a loan and an account?
- c) who have a loan but do not have an account?



Q Consider the following ORACLE relations:

One (x, y) = {<2, 5>, <1, 6>, <1, 6>, <1, 6>, <4, 8>, <4, 8>}

Two (x, y) = {<2, 55>, <1, 1>, <4, 4>, <1, 6>, <4, 8>, <4, 8>, <9, 9>, <1, 6>}

Consider the following two SQL queries SQ1 and SQ2:

SQ₁: SELECT * FROM One EXCEPT (SELECT * FROM Two);

SQ₂: SELECT * FROM One EXCEPT ALL (SELECT * FROM Two);

For each of the SQL queries, what is the cardinality (number of rows) of the result obtained when applied to the instances above?

- (a) 2 and 1 respectively
- (b) 1 and 2 respectively
- (c) 2 and 2 respectively
- (d) 1 and 1 respectively

Q Consider the below given query

SELECT column_name(s) FROM table₁

command

SELECT column_name(s) FROM table₂;

What should the database manager write in place of command to get distinct values from table1 and table2 of column_name(s)? [Asked in Cognizant]

- a) SET DIFFERENCE
- b) DISTINCT
- c) UNION
- d) UNION ALL

Q The UNION SQL clause can be used with _____ [Asked in Wipro NLTH]

- a) SELECT clause only
- b) DELETE and UPDATE clauses
- c) UPDATE clause only
- d) All of the mentioned

Q Suppose now that $R(A,B)$ and $S(A,B)$ are two relations with r and s tuples, respectively (again, not necessarily distinct). If m is the number of (not necessarily distinct) tuples in the result of the SQL query:

$R \text{ intersect } S;$

Then which of the following is the most restrictive, correct condition on the value of m ? [Asked in L&T Infotech (LTI)]

- a) $m = \min(r,s)$
- b) $0 \leq m \leq r + s$
- c) $\min(r,s) \leq m \leq \max(r,s)$
- d) $0 \leq m \leq \min(r,s)$

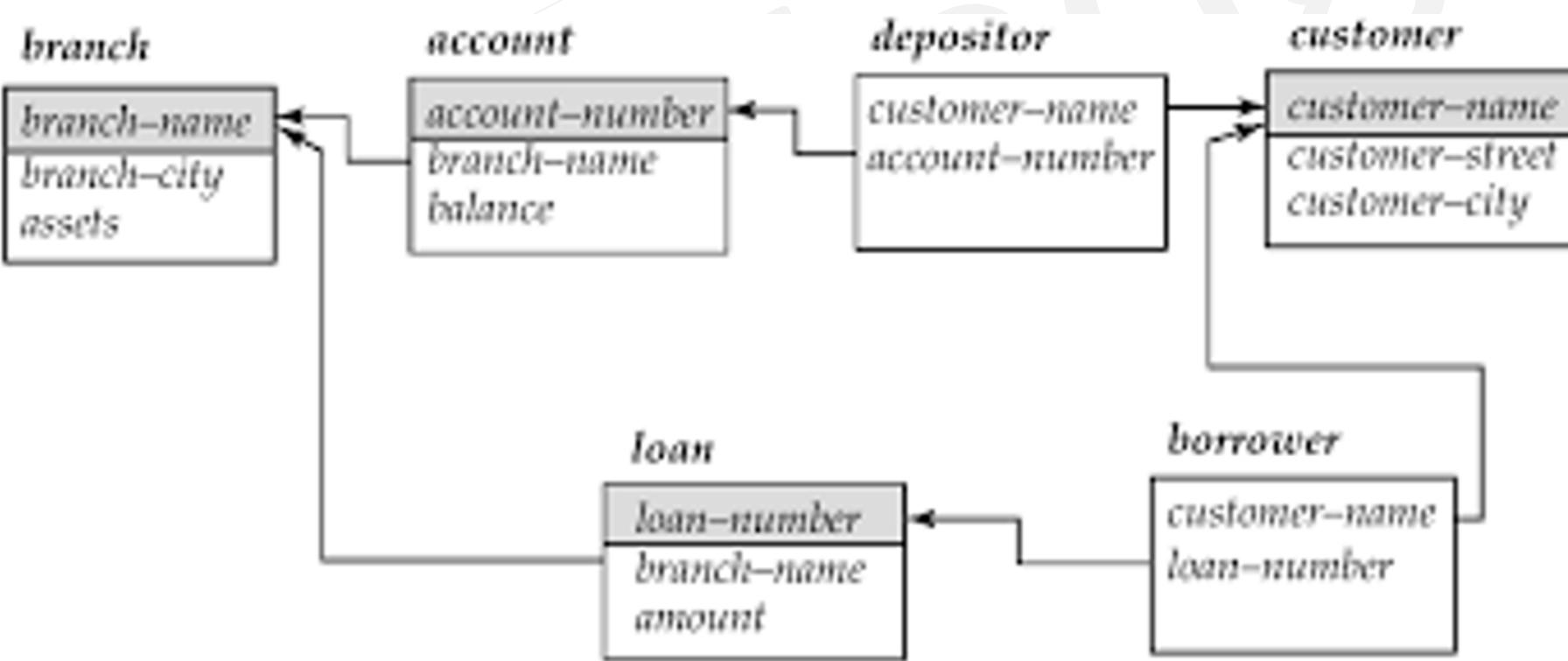
Queries on Multiple Relations

- **Need for multi-table queries** – Many real questions combine data from more than one table, not just a single relation.
 - **FROM creates a Cartesian product** – Listing two tables (r_1, r_2) forms every possible pair of their rows; filters in WHERE are then used to pick the matching pairs you actually need.
 - **Attribute name clashes** – If both tables have a column with the same name, qualify it with the table name or an alias ($r_1.id, r_2.id$) to avoid confusion.

	R₁
A	B
1	P
2	Q
3	R

	R₂
B	C
Q	X
R	Y
S	Z

- Q Write a SQL query to find the name of all the customers with account balance, who have an account in the bank?
- Q Write a SQL query to find the name of all the customers, who have a loan in the bank of less than 1000 or loan from north_delhi branch?
- Q Write a SQL query to find the name of the customer who have an account in the branch situated in Delhi?



Q) The given Query can be replaced with _____:[Tcs Ninja].

```
SELECT name  
FROM instructor1  
WHERE salary <= 100000 AND salary >= 90000;
```

a.

```
SELECT name  
FROM instructor1  
WHERE salary BETWEEN 100000 AND 90000
```

b.

```
SELECT name  
FROM instructor1  
WHERE salary BETWEEN 90000 AND 100000;
```

c.

```
SELECT name  
FROM instructor1  
WHERE salary BETWEEN 90000 AND 100000;
```

d.

```
SELECT name  
FROM instructor1  
WHERE salary <= 90000 AND salary >= 100000;
```

Ans=C.

Q) If table named Employee has 50 records and Employee2 has 0 records .what will be the output of following query? [IBM].

```
Select e1.*  
from Employee e1,Employee e2  
where e1.empno=e2.empno;
```

- (a) 4 Rows.
- (b) 6 Rows.
- (c) 2 Rows.
- (d) None of the .

Ans=D.

Q Consider the set of relations shown below and the SQL query that follows

Students: (Roll_number, Name, Date_of_birth)

Courses: (Course number, Course_name, Instructor)

Grades: (Roll_number, Course_number, Grade)

```
select distinct Name  
from Students, Courses, Grades  
where Students.Roll_number = Grades.Roll_number  
and Courses.Instructor = Korth  
and Courses.Course_number = Grades.Course_number  
and Grades.grade = A
```

Which of the following sets is computed by the above query?

- (A)** Names of students who have got an A grade in all courses taught by Korth
- (B)** Names of students who have got an A grade in all courses
- (C)** Names of students who have got an A grade in at least one of the courses taught by Korth
- (D)** None of the above

Natural Join

- **Purpose** – NATURAL JOIN lets you combine two tables without writing explicit join conditions; SQL matches rows automatically on all columns that share the same name in both tables.
- **Result formation** – Only rows whose common columns have equal values are kept; duplicate copies of the common columns are removed, so each shared attribute appears once.
- **Attribute order** – Output lists (1) the common columns, (2) the columns unique to the first table, then (3) the columns unique to the second table.
- **Commutative** – r_1 NATURAL JOIN r_2 returns the same result as r_2 NATURAL JOIN r_1 .
- **Multi-table syntax** – You can chain joins:

```
select A1, A2,..., An  
from r1 natural join r2 natural join ... natural join rm  
where P;
```

R_1	
A	B
1	P
2	Q
3	R

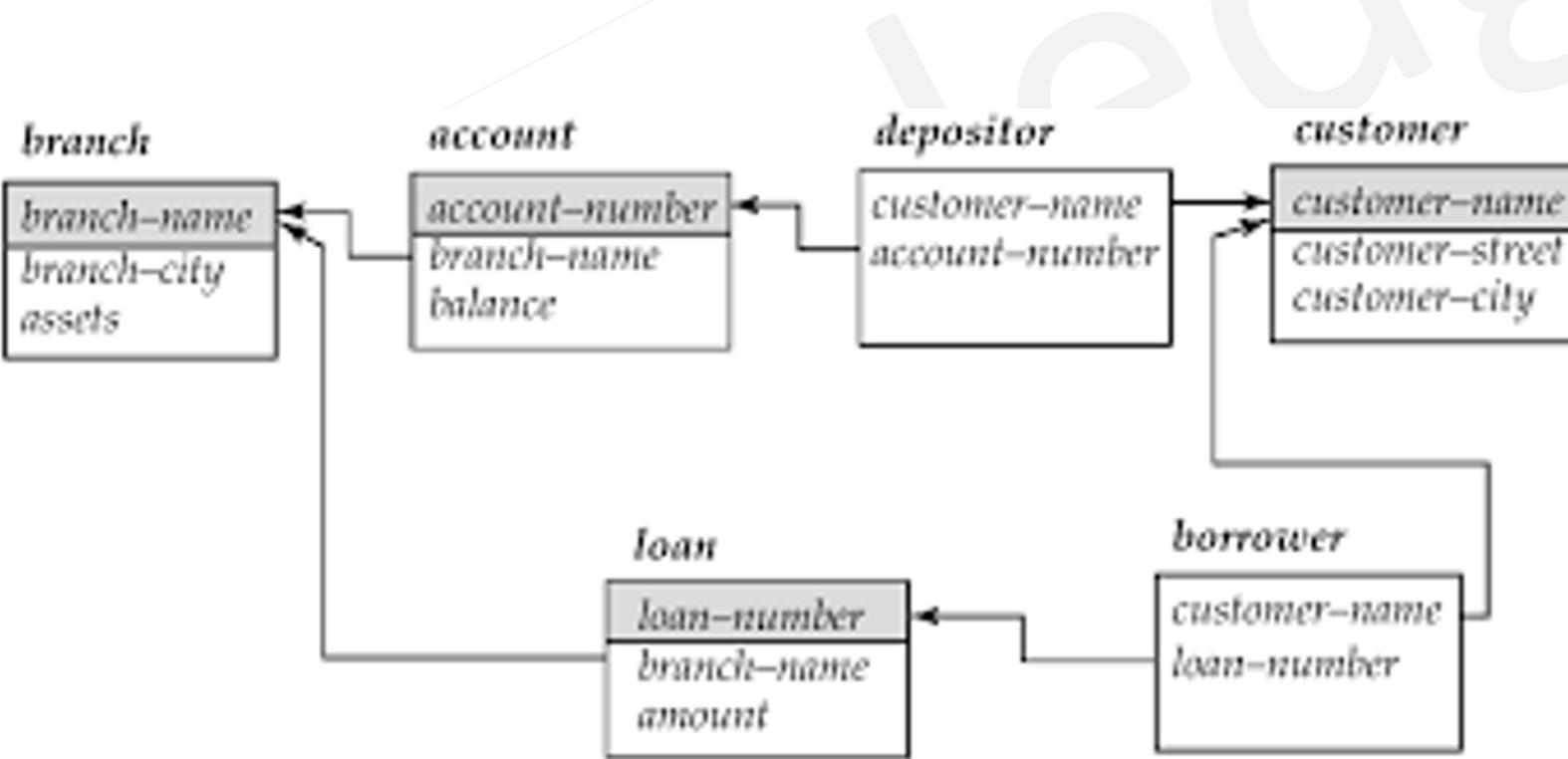
R_2	
B	C
Q	X
R	Y
S	Z

$R_1 * R_2$			
A	$R_1.B$	$R_2.B$	C
1	P	Q	X
1	P	R	Y
1	P	S	Z
2	Q	Q	X
2	Q	R	Y
2	Q	S	Z
3	R	Q	X
3	R	R	Y
3	R	S	Z

$R_1 \bowtie R_2$		
A	B	C

Q Write a SQL query to find the name of all the customers along with account balance, who have an account in the bank?

Q Write a SQL query to find the name of the customer who have an account in the branch situated in Delhi?



Q) The given Query can also be replaced with _____:[Cognizant].

```
SELECT name, course id  
FROM instructor, teaches  
WHERE instructor ID= teaches ID;
```

- (a)** Select name, course id from teaches,instructor where instructor_id=course_id;
- (b)** Select name, course_id from instructor natural join teaches;
- (c)** Select name, course_id from instructor;
- (d)** Select course_id from instructor join teaches;

Ans=B.

Outer Join

- **Inner Join vs. Outer Join:**
 - *Inner joins* (NATURAL JOIN) keep **only** the rows that match in both tables. Rows with no match are discarded.
 - *Outer joins* keep the matching rows **and** also preserve the non-matching rows by filling the missing side with NULL.
- **Why Outer Join?**
 - Prevents information loss when one table has rows with no counterpart in the other table.
- **Types of Outer Join**
 - **LEFT OUTER JOIN (LEFT JOIN)** – keeps all rows from the **left** table; unmatched right-table columns are filled with NULL.
 - **RIGHT OUTER JOIN (RIGHT JOIN)** – keeps all rows from the **right** table; unmatched left-table columns are filled with NULL.
 - **FULL OUTER JOIN** – keeps all rows from **both** tables; NULLs appear wherever a match is missing on either side.

R_1	
A	B
1	P
2	Q
3	R

R_2	
B	C
Q	X
R	Y
S	Z

$R_1 \bowtie R_2$		
A	B	C
2	Q	X
3	R	Y

$R_1 \bowtie R_2$		
A	B	C

$R_1 * R_2$			
A	$R_1.B$	$R_2.B$	C
1	P	Q	X
1	P	R	Y
1	P	S	Z
2	Q	Q	X
2	Q	R	Y
2	Q	S	Z
3	R	Q	X
3	R	R	Y
3	R	S	Z

$R_1 \bowtie R_2$		
A	B	C

$R_1 \bowtie R_2$		
A	B	C

Q Consider the following two tables and four queries in SQL.

Book (isbn, bname)

Stock (isbn, copies)

Query 1: SELECT B.isbn, S.copies FROM Book B INNER JOIN Stock S ON B.isbn = S.isbn;	Query 2: SELECT B.isbn, S.copies FROM Book B LEFT OUTER JOIN Stock S ON B.isbn = S.isbn;
Query 3: SELECT B.isbn, S.copies FROM Book B RIGHT OUTER JOIN Stock S ON B.isbn = S.isbn;	Query 4: SELECT B.isbn, S.copies FROM Book B FULL OUTER JOIN Stock S ON B.isbn = S.isbn;

Which one of the queries above is certain to have an output that is a superset of the outputs of the other three queries?

- (a) Query 1
- (b) Query 2
- (c) Query 3
- (d) Query 4

Q Suppose ORACLE relation R(A, B) currently has tuples $\{(1, 2), (1, 3), (3, 4)\}$ and relation S(B, C) currently has $\{(2, 5), (4, 6), (7, 8)\}$.

Consider the following two SQL queries SQ_1 and SQ_2 :

SQ_1 : Select * From R Full Join S On R.B = S.B;

SQ_2 : Select * From R Inner Join S On R.B = S.B;

The numbers of tuples in the result of the SQL query SQ_1 and the SQL query SQ_2 are given by:

[Asked in CoCubes]

(a) 2 and 6 respectively

(b) 6 and 2 respectively

(c) 2 and 4 respectively

(d) 4 and 2 respectively

Q Which of the following is true ?

- I. Implementation of self-join is possible in SQL with table alias.
 - II. Outer-join operation is basic operation in relational algebra.
 - III. Natural join and outer join operations are equivalent. **[Asked in Cognizant]**
- A) I and II are correct.
 - B) II and III are correct.
 - C) Only III is correct.
 - D) Only I is correct.

Q) Select the correct query/queries for cross join:[Amcat]

- (a)** Select * FROM Table1 T1 CROSS JOIN Table1 T2;
- (b)** Select * FROM Table1 T1 ALL CROSS JOIN Table1 T2;
- (c)** Select * FROM Table1 T1 CROSS Table1 T2;
- (d)** None of these.

Ans=A.

Q) Which of the join operations do not preserve non matched tuples.[Amcat].

- (a) Left outer join**
- (b) Right outer join**
- (c) Inner join**
- (d) None of these**

Ans=C.

Q) The SQL join which does Cartesian product is called. [Amcat].

- (a) Left Join**
- (b) Left Outer Join**
- (c) Right Outer Join**
- (d) Cross Join**

Ans=D.

Q) Which are the join types in join condition:[Amcat].

- (a) Cross join**
- (b) Natural join**
- (c) Join with USING clause**
- (d) All of the mentioned**

Ans=D.

Q) Which join is to be used between two tables A and B when the resultant table needs rows from A and B that matches the condition and rows from A that does not match the condition.[Amcat]

- (a) Outer Join
- (b) Cross Join
- (c) Inner Join
- (d) None of the above

Ans=A.

Q) SQL the statement select * from R, S is equivalent to.[Amcat].

- (a) Select * from R natural join S
- (b) Select * from R cross join S
- (c) Select * from R union join S
- (d) Select * from R inner join S

Ans=B.

Q Consider the tables project and allocation given below:[infosys].

Projectid	Projectna
Infy101	NAM
Infy102	All State Bank
Infy103	Suntrust
Infy104	Swiss Insure

Emepiede	Empne	Projectid
E300	Olivia	Infy102
E301	Hussy	Infy102
E302	Mike	Infy103
E303	Jack	
E304	Smith	

```
SELECT p.projectid,p.projectname,a.empid  
FROM project p FULL JOIN allocation a  
ON p.projectid=a.projectid AND a.projectid IS NOT NULL ;
```

How many rows will be fetched when the above query is executed?

- (a) 7
- (b) 5
- (c) 3
- (d) 6

Answer : C

Q) Ravi Sastri is a good coach. He wants to join 2 relational tables related to sports. He also wants that all the matched and unmatched tuples of the right table should be present in the result but only the matched rows of the left table should be present in the result after the join operation. Which of the following joins he should use.[asked in InfyTQ].

- (a) Equi Join
- (b) Left outer join
- (c) Right outer join
- (d) Full outer join

Ans=C.

Q) Which of the following type of Joins you will use when you have to join a table with itself? [L&T InfoTech].

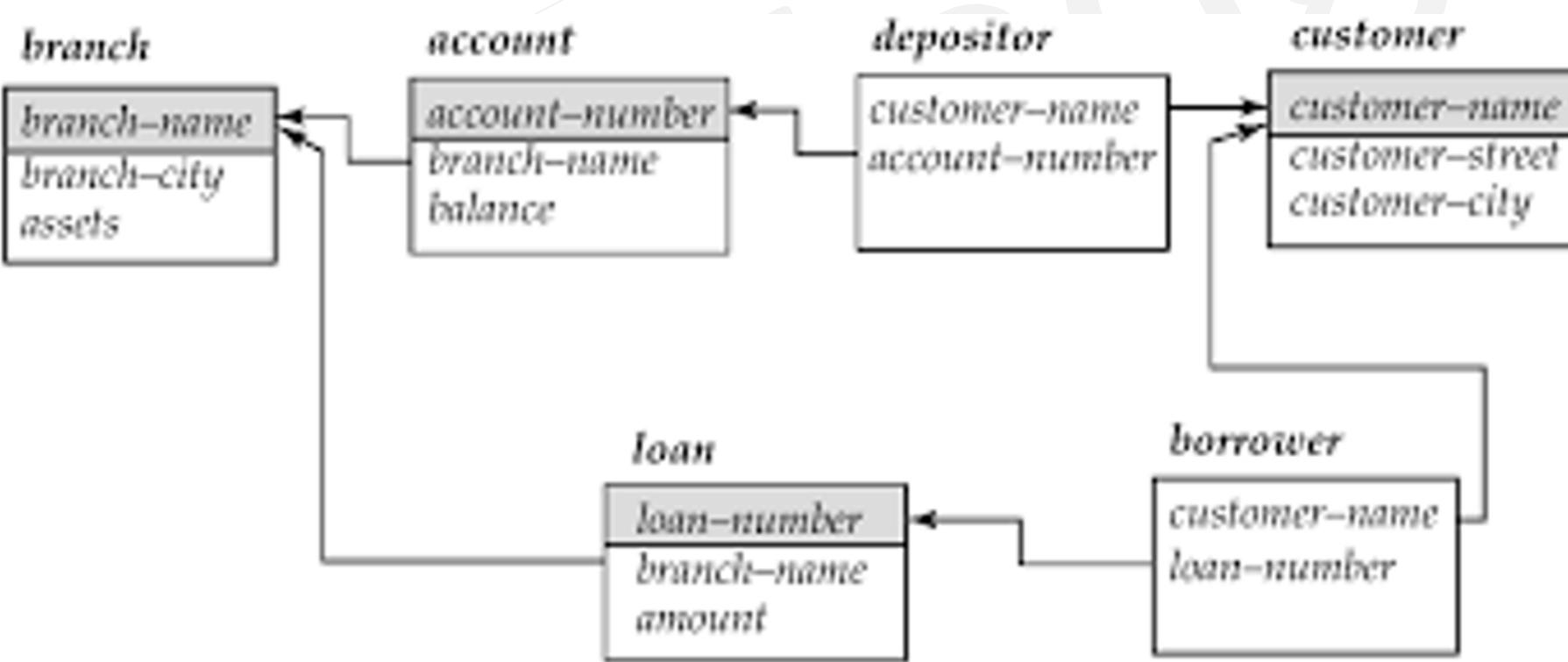
- (a) Equal Join
- (b) Outer Join
- (c) Self Join
- (d) Inner Join

Ans=C.

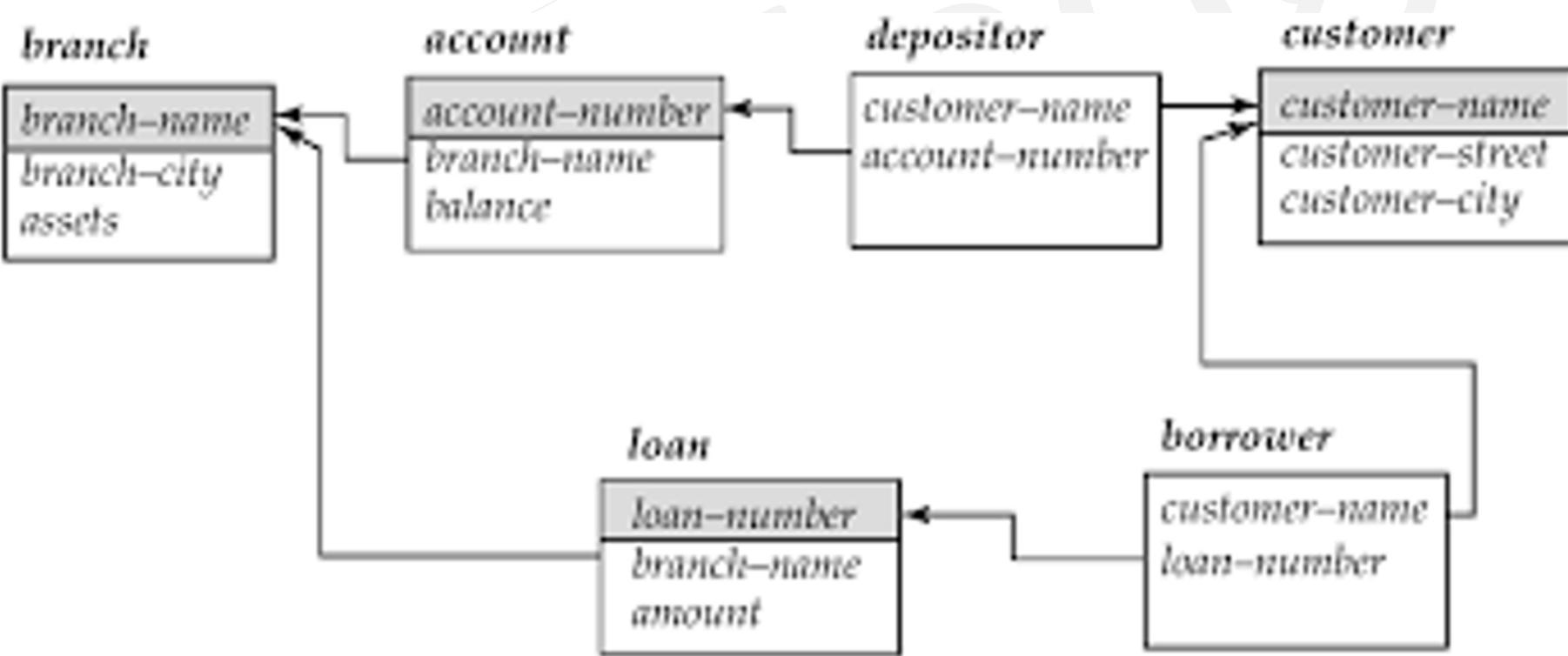
Alias Operation/ rename

- SQL aliases are used to give a table, or a column in a table, a temporary name. Just create a new copy but do not change anything in the data base.
- Aliases are often used to make column names more readable.
- It uses the as clause, taking the form: old-name as new-name. The as clause can appear in both the select and from clauses.
- An alias only exists for the duration of the query.

Q Write a SQL query to find the account_no along and balance with 8% interest, as Account, total_balance?



Q Write a SQL query to find the loan_no with maximum loan amount?



Q The SQL expression

Select distinct T.branch_name

from branch T, branch S

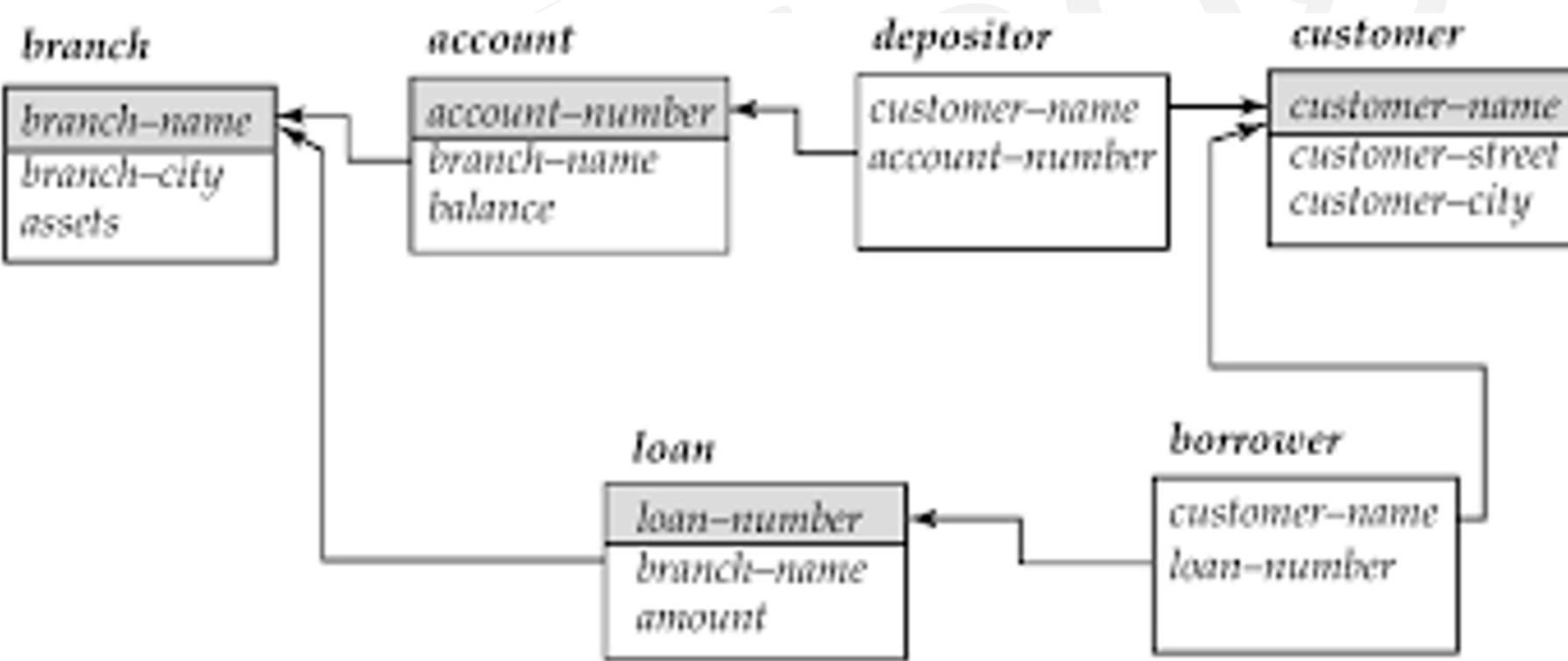
where T.assets > S.assets and S.branch_city="Mumbai" finds the names of **(NET-SEP-2013)**

- (A) All branches that have greater assets than some branch located in Mumbai.
- (B) All branches that have greater assets than all branches in Mumbai.
- (C) The branch that has greatest asset in Mumbai.
- (D) Any branch that has greater assets than any branch in Mumbai.

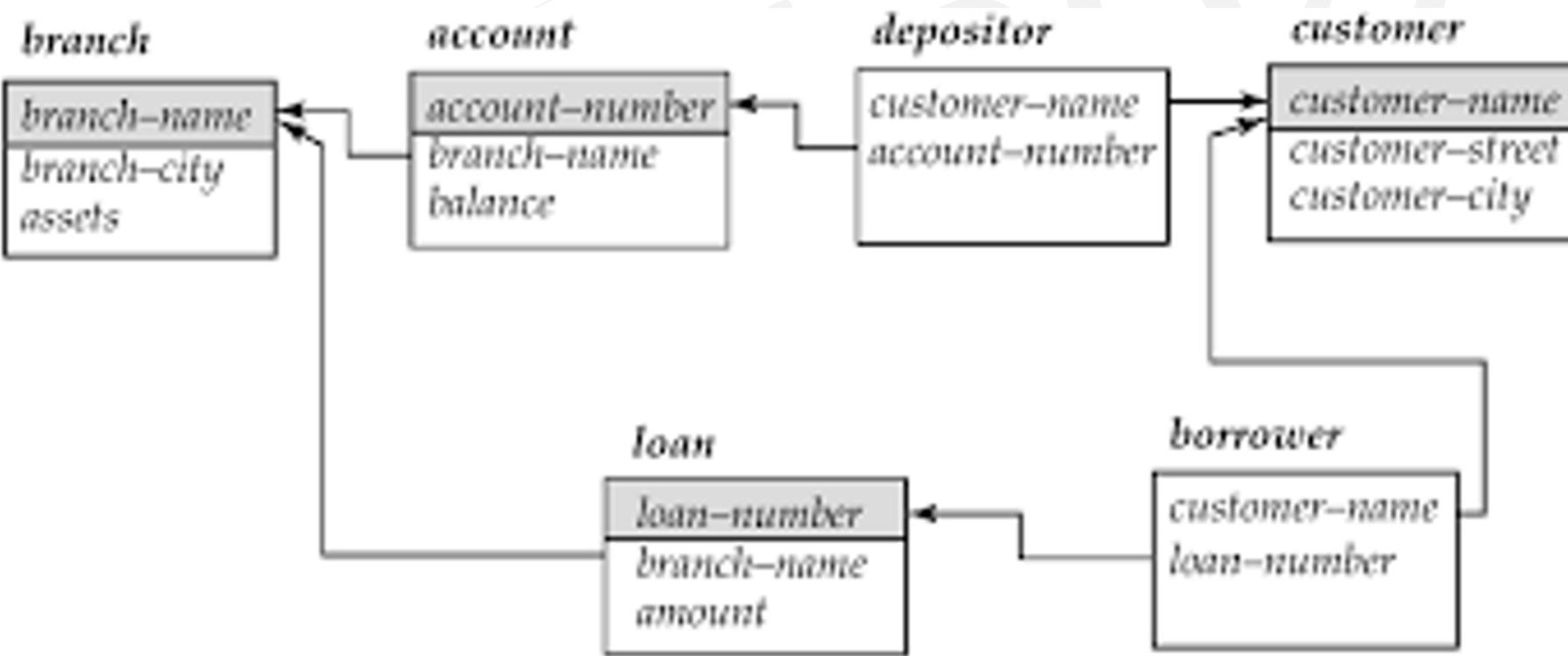
Aggregate Functions

- *Aggregate functions* are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five built-in aggregate functions:
 - Average: **avg**
 - Minimum: **min**
 - Maximum: **max**
 - Total: **sum**
 - Count: **count**
- The input to **sum** and **avg** must be a collection of numbers, but the other operators can operate on collections of nonnumeric data types, such as strings, as well.
- We use the aggregate function **count** frequently to count the number of tuples in a relation. The notation for this function in SQL is **count (*)**.
- Count is the only aggregate function which can work with null, all other aggregate functions simply ignore null.

Q Write a SQL query to find the number of accounts in the bank?



Q Write a SQL query to find the average balance of every account in the banks from south_delhi branch?



Q Consider a table along with two query?

**Select avg (balance)
from account**

Account_no	balance	Branch_name
Abc123	100	N_delhi
Pqr123	500	S_mumbai
Wyz123	null	S_delhi

**Select sum(balance)/count(balance)
from account**

Q In SQL, _____ is an Aggregate function.

- (a) SELECT
- (b) CREATE
- (c) AVG
- (d) MODIFY

Q) Which of the following is not a function of aggregate function? [Hexaware]

- (a) Avg
- (b) Count
- (c) Create
- (d) Max

Q Which of the following is aggregate function in SQL? [Asked in Wipro NLTH]

- A) Avg
- B) Select
- C) Ordered by
- D) distinct

Q Count function in SQL returns the number of [Asked in TCS NQT]

- A) values.
- B) distinct values.
- C) groups.
- D) columns.

Q Given the following Employee relational database. What does the following expression result ? [Asked in TCS NQT]

Select MAX(Salary)
from Employee
where Salary < (Select MAX(Salary) from Employee)

- a) 3000
- b) 4000
- c) 7000
- d) 8000

ID	First Name	Salary
1	Ben	8000
2	Joe	4000
3	Mark	7000
4	Steve	3000
5	John	7000

Q Table employees has 10 records. It has a non-NULL SALARY column which is also UNIQUE. The SQL statement [Asked in Wipro NLTH]

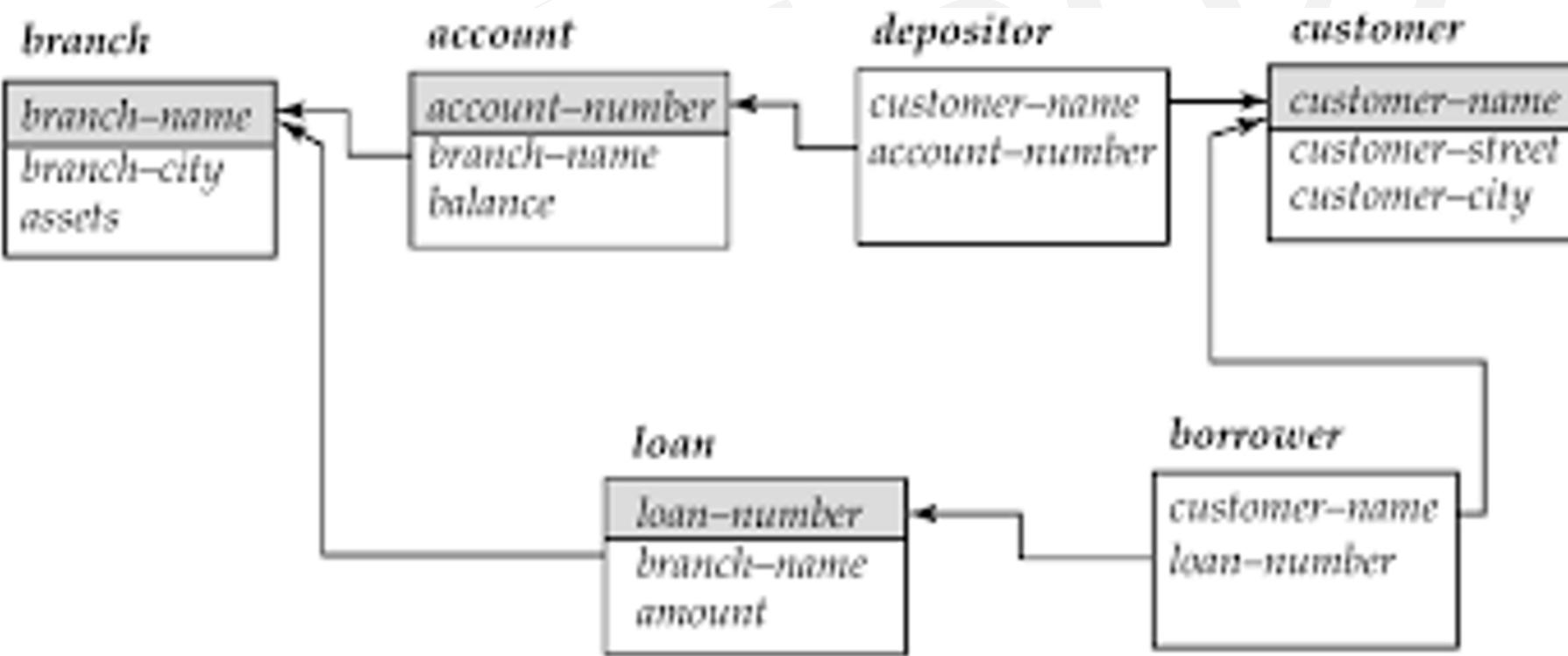
```
SELECT COUNT(*)  
FROM EMPLOYEE  
WHERE SALARY > ALL (SELECT SALARY FROM EMPLOYEE);
```

- a) 10
- b) 9
- c) 5
- d) 0

Ordering the Display of Tuples

- SQL offers the user some control over the order in which tuples in a relation are displayed. The **order by** clause causes the tuples in the result of a query to appear in sorted order.

Q Write a SQL query to find all the branch_name which are situated in Delhi in alphabetic order?



Q) In the following Query, which of the following can be placed in the Query's blank portion to display the salary from highest to lowest amount, and sorting the employ's name alphabetically? [Tcs Ninja].

```
SELECT *  
FROM instructor  
ORDER BY salary ___, name ___;
```

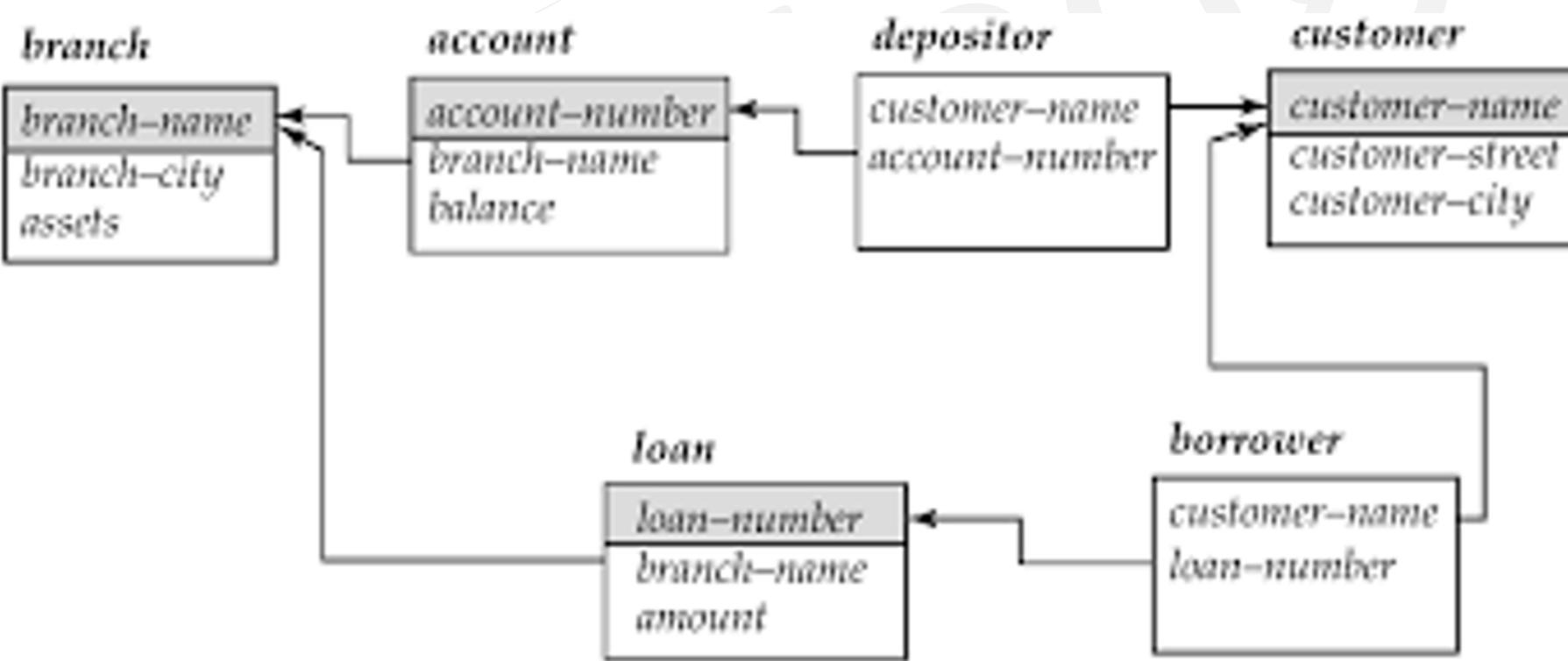
- (a) Ascending, Descending
- (b) Asc , Desc
- (c) Desc , Asc
- (d) All of the above

String Operations

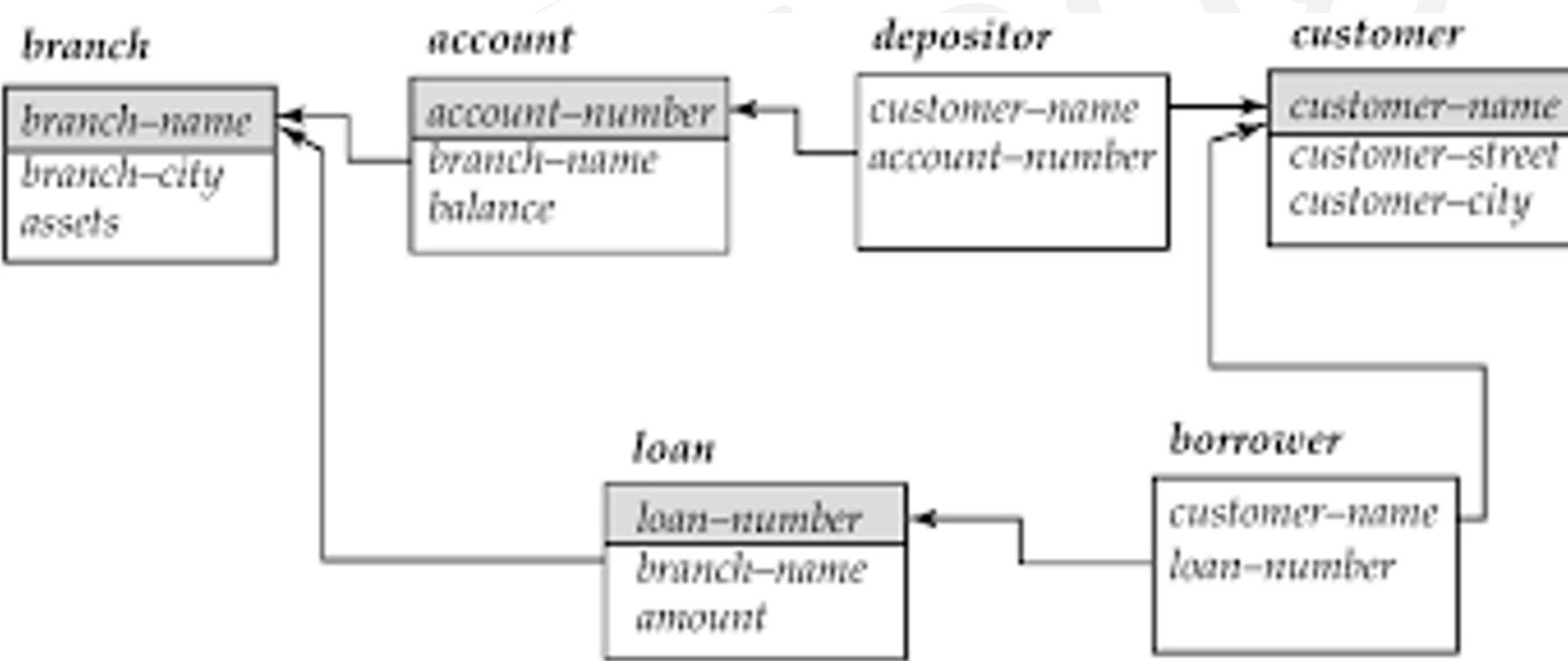
- SQL specifies strings by enclosing them in single quotes, for example, 'Computer'. The SQL standard specifies that the equality operation on strings is case sensitive; as a result the expression 'Computer' = 'computer' evaluates to false.
- However, some database systems, such as MySQL and SQL Server, do not distinguish uppercase from lowercase when matching strings; as a result, would evaluate to true on these databases. This default behavior can, however, be changed, either at the database level or at the level of specific attributes.
- SQL also permits a variety of functions on character strings, such as concatenating, extracting substrings, finding the length of strings, converting strings to uppercase and lowercase, removing spaces at the end of the string and so on. There are variations on the exact set of string functions supported by different database systems.
- A single quote character that is part of a string can be specified by using two single quote characters; for example, the string It's right can be specified by It"’s right.

- Pattern matching can be performed on strings, using the operator `like`. We describe patterns by using two special characters:
 - Percent (%): The % character matches any substring.
 - Underscore (_): The _ character matches any character.
- '%Comp%' matches any string containing “Comp” as a substring, for example, ‘Intro to Computer Science’, and ‘Computational Biology’.
- ‘___’ matches any string of exactly three characters.
- ‘__%’ matches any string of at least three characters.

Q Write a SQL query to find all the branch name who have exactly 5 character in their name ?



Q Write a SQL query to find all the customer name who have 'kumar' in their name ?



- For patterns to include the special pattern characters (that is, % and _), SQL allows the specification of an escape character.
- The escape character is used immediately before a special pattern character to indicate that the special pattern character is to be treated like a normal character.
- We define the escape character for a **like** comparison using the **escape** keyword. To illustrate, consider the following patterns, which use a backslash (\) as the escape character:
 - **like 'ab\%cd%' escape '\'** matches all strings beginning with "ab%cd".
 - **like 'ab\\cd%' escape '\'** matches all strings beginning with "ab\cd".
- SQL allows us to search for mismatches instead of matches by using the **not like** comparison operator. Some databases provide variants of the **like** operation which do not distinguish lower and upper case.
- SQL:1999 also offers a **similar to** operation, which provides more powerful pattern matching than the **like** operation; the syntax for specifying patterns is similar to that used in Unix regular expressions.

Q) Ready the Query carefully:

```
SELECT emp name  
FROM department  
WHERE dept name LIKE ' _____ Computer Science'; [Tcs Ninja].
```

In the above-given Query, which of the following can be placed in the Query's blank portion to select the "dept_name" that also contains Computer Science as its ending string?

- (a) &
- (b) _
- (c) %
- (d) \$

Ans=C.

Q Consider a “CUSTOMERS” database table having a column “CITY” filled with all the names of Indian cities (in capital letters). The SQL statement that finds all cities that have “GAR” somewhere in its name, is:

- (a)** Select * from customers where city = '%GAR%'
- (b)** Select * from customers where city = '\$GAR\$'
- (c)** Select * from customers where city like '%GAR%'
- (d)** Select * from customers where city as '%GAR'

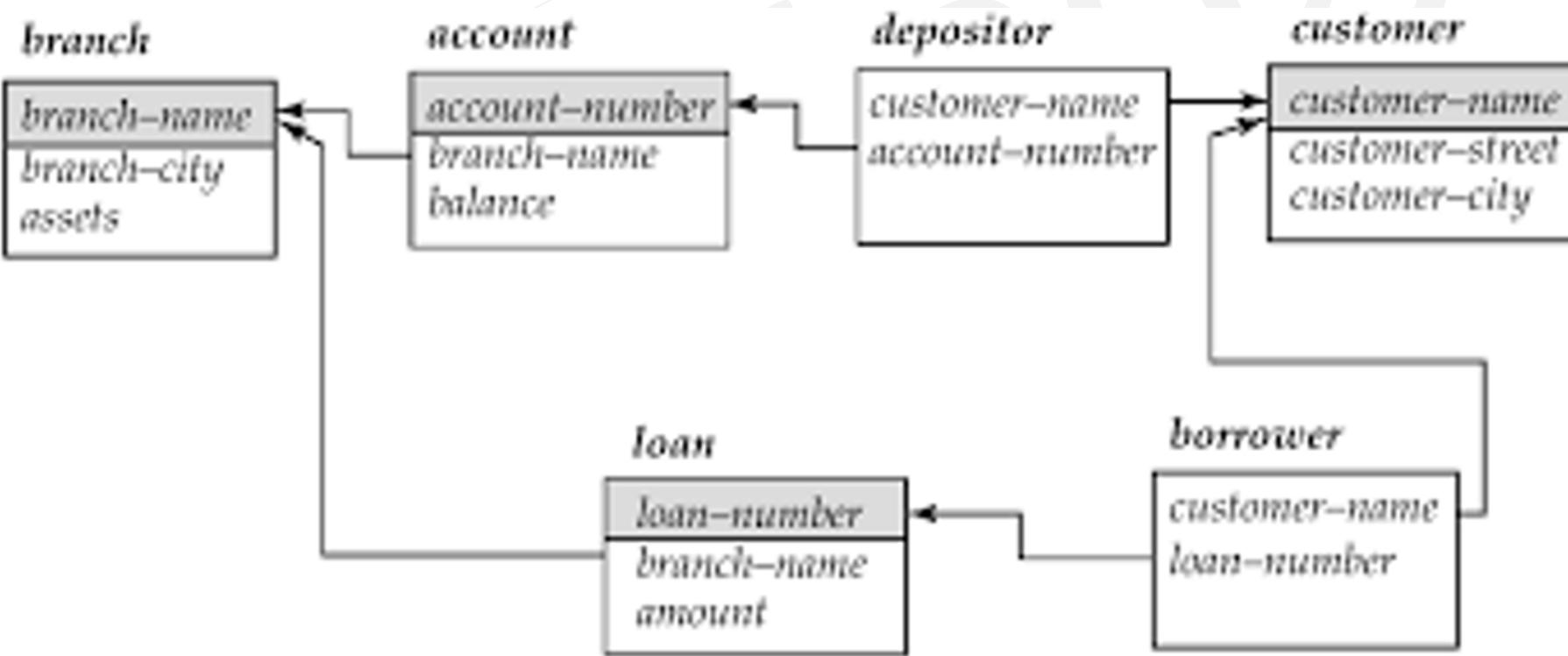
Q Write the SQL statement for the query : “ Find the name of the students whose name contains letter ‘D’.

- a)** Select name from student where name like ‘%D%’ ;
- b)** Select name from student where name like ‘%D’ ;
- c)** Select name from student where name like ‘D%’ ;
- d)** none of the above

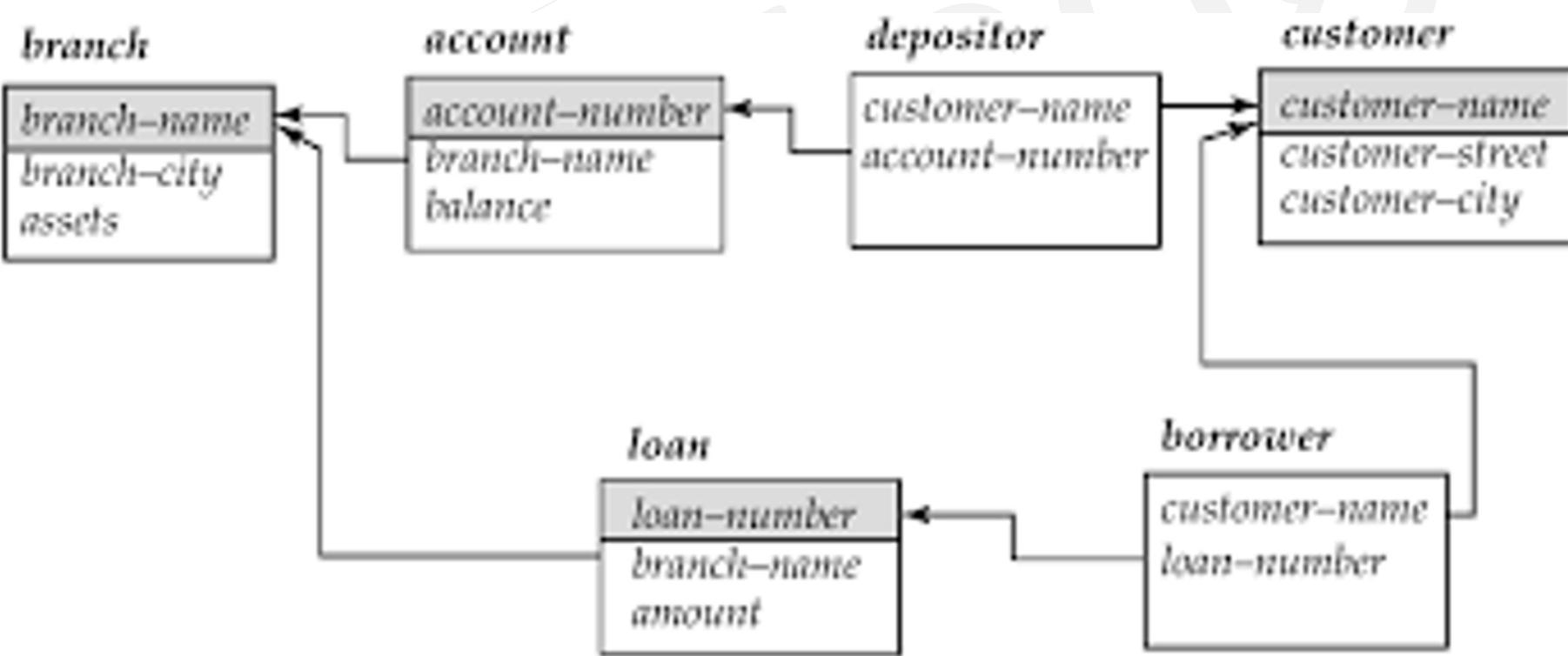
Group by clause

1. There are circumstances where we would like to work on a set of tuples in a relation rather than working on the whole table as one unit.
2. The attribute or attributes given in the **group by** clause are used to form groups. Tuples with the same value on all attributes in the **group by** clause are placed in one group.

Q Write a SQL query to find the average account balance of each branch?



Q Write a SQL query to find the branch name of Gwalior city with average balance more than 1500?



Q

Consider the following tables (relations) :

Students

<u>Roll-No</u>	Name
18CS101	Ramesh
18CS102	Mukesh
18CS103	Ramesh

Performance

<u>Roll-No</u>	<u>Course</u>	Marks
18CS101	DBMS	60
18CS101	Compiler Design	65
18CS102	DBMS	80
18CS103	DBMS	85
18CS102	Compiler Design	75
18CS103	Operating System	70

Primary keys in the tables are shown using underline. Now, consider the following query :

```
SELECT S.Name, Sum (P.Marks)
FROM Students S, Performance P
WHERE S.Roll-No = P.Roll-No
GROUP BY S.Name
```

The number of rows returned by above query is

- a) 0
- b) 1
- c) 2
- d) 3

Q Consider the following ORACLE relations:

R (A, B, C) = {<1, 2, 3>, <1, 2, 0>, <1, 3, 1>, <6, 2, 3>, <1, 4, 2>, <3, 1, 4>}

S (B, C, D) = {<2, 3, 7>, <1, 4, 5>, <1, 2, 3>, <2, 3, 4>, <3, 1, 4>}.

Consider the following two SQL queries SQ_1 and SQ_2 :

SQ_1 : SELECT R·B, AVG (S·B)

FROM R, S

WHERE R·A = S·C AND S·D < 7

GROUP BY R·B;

SQ_2 : SELECT DISTINCT S·B, MIN (S·C)

FROM S

GROUP BY S·B

HAVING COUNT (DISTINCT S·D) > 1;

If M is the number of tuples returned by SQ_1 and N is the number of tuples returned by SQ_2 then

- (a) M = 4, N = 2
- (b) M = 5, N = 3
- (c) M = 2, N = 2
- (d) M = 3, N = 3

A	B	C
1	2	3
1	2	0
1	3	1
6	2	3
1	4	2
3	1	4

B	C	D
2	3	7
1	4	5
1	2	3
2	3	4
3	1	4

Q The relation scheme given below is used to store information about the employees of a company, where empld is the key and deptId indicates the department to which the employee is assigned. Each employee is assigned to exactly one department.

emp(empld, name, gender, salary, deptId)

Consider the following SQL query:

```
select deptId, count(*)  
from emp  
where gender = "female" and salary > (select avg(salary)from emp)  
group by deptId;
```

The above query gives, for each department in the company, the number of female employees whose salary is greater than the average salary of [Asked in E-Litmus]

- A) employees in the department
- B) employees in the company
- C) female employees in the department
- D) female employees in the company

Q The STUDENT information in a university stored in the relation STUDENT (Name, SEX, Marks, DEPT_Name)

Consider the following SQL Query

```
SELECT DEPT_Name  
from STUDENT  
where SEX = 'M'  
group by DEPT_Name  
having avg (Marks)>SELECT avg (Marks) from STUDENT.
```

It Returns the Name of the Department for which: [Asked in Adobe]

A) The Average marks of Male students is more than the average marks of students in the same Department

B) The average marks of male students is more than the average marks of students in the University

C) The average marks of male students is more than the average marks of male students in the University

D) The average marks of students is more than the average marks of male students in the University

Nested Subqueries

A subquery is a **select-from- where** expression that is nested within another query. A common use of sub-queries is to perform

- Tests for set membership
- make set comparisons
- determine set cardinality, by nesting subqueries in the **where** clause.

1. SQL allows testing tuples for membership in a relation.
2. The **in** connective tests for set membership, where the set is a collection of values produced by a **select** clause.
3. The **not in** connective tests for the absence of set membership.

Q Consider the following relation:

Works (emp_name, company_name, salary) Here, emp_name is primary key. Consider the following SQL query

Select emp_name

From works T

where salary > (select avg (salary)

from works S

where T.company_name = S.company_name)

The above query is for following :

- (a) Find the highest paid employee who earns more than the average salary of all employees of his company.
- (b) Find the highest paid employee who earns more than the average salary of all the employees of all the companies.
- (c) Find all employees who earn more than the average salary of all employees of all the companies.
- (d) Find all employees who earn more than the average salary of all employees of their company.

Q Consider the query :

```
SELECT student_name  
FROM student_data  
WHERE rollno (SELECT rollno  
              FROM student_marks  
              WHERE SEM1_MARK=SEM2_MARK);
```

Which of the following is true?

- (A) It gives the name of the student whose marks in semester 1 and semester 2 are same.
- (B) It gives all the names and roll nos of those students whose marks in semester 1 and semester 2 are same.
- (C) It gives the names of all the students whose marks in semester 1 and semester 2 are same.
- (D) It gives roll numbers of all students whose marks in semester 1 and semester 2 are same.

Q Consider the query :

```
SELECT student_name  
FROM students  
WHERE class_name = (SELECT class_name  
                     FROM students  
                     WHERE math_marks=100);
```

what will be the output ?

- (A) the list of names of students with 100 marks in mathematics
- (B) the names of all students of all classes in which at least one student has 100 marks in mathematics
- (C) the names of all students in all classes having 100 marks in mathematics
- (D) the names and class of all students whose marks in mathematics is 100

Consider the following Relation COFFEE with two columns NAME and PRICE, COFFEE

[Asked in Cognizant]

NAME	PRICE
Espresso	1000
Cappuccino	900
Espresso Macchiato	1500
Americano	500
Cafe Breve	600
Tea	NULL

a) 6

b) 4

c) 5

d) 3

SELECT Count (*) FROM COFFEE

WHERE PRICE > ANY (SELECT PRICE FROM COFFEE)

output is

Q Which SQL operator tests if any of the sub-query value meets the condition ? [Asked in E-Litmus]

- a) LIKE operator
- b) SOME operator
- c) OR operator
- d) IN operator

Q Consider the set of relations shown below and the SQL query that follows

Students: (Roll_number, Name, Date_of_birth)

Courses: (Course number, Course_name, Instructor)

Grades: (Roll_number, Course_number, Grade)

```
select distinct Name  
from Students, Courses, Grades  
where Students.Roll_number = Grades.Roll_number  
and Courses.Instructor = Korth  
and Courses.Course_number = Grades.Course_number  
and Grades.grade = A
```

Which of the following sets is computed by the above query?

- (A)** Names of students who have got an A grade in all courses taught by Korth
- (B)** Names of students who have got an A grade in all courses
- (C)** Names of students who have got an A grade in at least one of the courses taught by Korth
- (D)** None of the above

Q The number of rows returned by SQL query on the given EMP table is _____

[Asked in Capgemini]

```
SELECT *  
FROM EMP  
WHERE eno NOT IN (SELECT manager FROM EMP);
```

- a) 0
- b) 1
- c) 3
- d) error

eno	ename	manager
1	a	2
2	b	3
3	c	4
4	d	NULL

Q Consider a relation **book** (title, price) which contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list? Select title from book as B where (select count (*)

from book as T

where T.price > B.price) < 5

(a) Titles of the six most expensive books.

(b) Title of the sixth most expensive books.

(c) Titles of the seven most expensive books.

(d) Title of the seventh most expensive books.

Test for Empty Relations

```
Select branch_name  
from branch as a  
Where assets <= 1,00,000 and  
exists(Select *  
      from Branch as b  
      where branch_city = 'Gwalior'  
        and a.branch_name = b.branch_name)
```

- SQL includes a feature for testing whether a subquery has any tuples in its result. The exists construct returns the value true if the argument subquery is nonempty.
- The above query also illustrates a feature of SQL where a correlation name from an outer query, can be used in a subquery in the where clause. A subquery that uses a correlation name from an outer query is called a correlated subquery.

Q Consider the table **Student**(stuid, name, course, marks).

Which one of the following two queries is correct to find the highest marks student in course 5 ?

Q.1 Select S.stuid

From student S

Where not exists (select *

from student e

where e course = '5' and e marks \geq s marks)

Q.2 Select s.stu.id

From student S

Where s · marks > any (select distinct marks

from student S

where s · course = 5)

(A) Q.1

(B) Q.2

(C) Both Q.1 and Q.2

(D) Neither Q.1 nor Q.2

Q Consider the relational database with the following four schemas and their respective instances. Student(sNo, sName, dNo) Dept(dNo, dName)
Course(cNo, cName, dNo) Register(sNo, cNo)

Student		
<u>sNo</u>	sName	dNo
S01	James	D01
S02	Rocky	D01
S03	Jackson	D02
S04	Jane	D01
S05	Milli	D02

Dept	
<u>dNo</u>	dName
D01	CSE
D02	EEE

	Course	
<u>cNo</u>	cName	dNo
C11	DS	D01
C12	OS	D01
C21	DE	D02
C22	PT	D02
C23	CV	D03

Register	
<u>sNo</u>	<u>cNo</u>
S01	C11
S01	C12
S02	C11
S03	C21
S03	C22
S03	C23
S04	C11
S04	C12
S05	C11
S05	C21

SQL Query:

```
SELECT * FROM Student AS S WHERE NOT EXIST  
(SELECT cNo FROM Course WHERE dNo = "D01")
```

EXCEPT

```
SELECT cNo FROM Register WHERE sNo = S.sNo)
```

The number of rows returned by the above SQL query is _____.

ANS:- (2)

knowledgeGate

www.knowledgegate.ai

Q) Which of the operation are not specifies in triggers.[Amcat].

- (a) ALTER
- (b) UPDATE
- (c) INSERT
- (d) DELETE

Ans=A.

Q) In SQL, which of the following commands is used to add a column to a table in a database? [Tcs Ninja].

- a) ALTER TABLE <table name> ADD <column name>
- b) MODIFY TABLE <table name> ADD <column name>
- c) MODIFY TABLE <table name> INSERT <column name>
- d) ALTER TABLE <table name> INSERT <column name>

Ans=A.

Q) Which of the following is CORRECT about DELETE command?.[Accenture].

- a)** It can delete single or multiple columns from a table
- b)** It can delete single or multiple records from a table
- c)** Records deleted can be roll backed.
- d)** None of these.

Q) Consider the following two commands, C1 and C2, on the relation R from an SQL database:[Tcs Ninja].

C1: drop table R;

C2: delete from R;

Which of the following statements is TRUE?

- I. Both C1 and C2 delete the schema for R.
- II. C2 retains relation R but deletes all tuples in R.
- III. C1 deletes all tuples of R and the schema for R.

(a) I only

(b) I and II only

(c) II and III only

(d) I, II, and III

Ans=C.

Q) What is correct about DELETE command and TRUNCATE command? [Accenture].

- (1) Delete command can be Rollback.
 - (2) Truncate command can be Rollback.
 - (3) Delete command cannot be Rollback.
 - (4) Truncate command cannot be Rollback.
-
- (a) 1 &2
 - (b) 1 &4
 - (c) 2&3
 - (d) All of the above.

Ans=B.

Q) Which one is correct syntax for Insert Statement.[Amcat].

- (a)** Insert table name Columns(Col1, Col2, Col3);
- (b)** Insert into table name (Col1, Col2, Col3) VALUES (Val1, Val2, Val3);
- (c)** Insert Columns(Col1, Col2, Col3) VALUE (Val1, Val2, Val3) Into table name;
- (d)** None of the above

Ans=B.

Q) Which one is correct syntax for Update Statement.[Amcat].

- (a) Update Table table name Columns(Col1, Col2, Col3);
- (b) Update into table name (Col1, Col2, Col3) VALUES (Val1, Val2, Val3);
- (c) Update table name Set Col name=Value;
- (d) None of the above

Ans=C.

knowledgeGate

www.knowledgegate.ai

Q) The CREATE TRIGGER statement is used to create the trigger. THE _____ clause specifies the table name on which the trigger is to be attached. The _____ specifies that this is an AFTER INSERT trigger.[Amcat].

- (a) For insert, on
- (b) On, for insert
- (c) For, insert
- (d) Both a and c

Ans=B.

Q) What is the error in the following code of PL/SQL: [Amcat].

```
declare  
a integer :=5;  
b integer :=10;  
c integer;  
begin  
c:=a+b;  
dbms _output .put _line ('sum='||c);  
end;
```

- (a) Dbms _output. put_line('sum='||c);
- (b) Integer :=5;
- (c) C:=A+B;
- (d) Output. put_line('sum='||c);

Q) How can we specifies a row-level trigger.[Amcat].

- (a) Using ON ROW
- (b) Using FOR EACH COL
- (c) Using FOR EACH ROW
- (d) Using OR ROW

Ans=C.

Q) Anil is designing a database of motor vehicles. It has one base entity ‘Vehicles’ which is classified into two sub-entities, 2 -wheeler and 4-wheeler. He further breaks them down into more entities. What is the process being used here? [Amazon].

- (a) Generalization
- (b) Specialization
- (c) Aggregation
- (d) Segregation

Q) Metadata about relations are stored in a structure known as.[Amcat].

- (a)** Sequence dictionary
- (b)** Storage dictionary
- (c)** Data Dictionary
- (d)** Functional Dictionary

Ans=C.

Q) Data dictionary tell DBMS.[Amcat].

- (a) What files are in the database**
- (b) What attributes are processed by data**
- (c) What these files contain**
- (d) All of above**

Ans=D.

TRANSACTION

- According to general computation principle (operating system) we may have partially executed program, as the level of atomicity is instruction i.e. either an instruction is executed completely or not. But in DBMS view, user perform a logical work(operation) which is always atomic in nature i.e. either operation is execute or not executed, there is no concept like partial execution. For example, Transaction T_1 which transfer 100 units from account A to B.
- In this transaction if a failure occurs after Read(B) then the final statue of the system will be inconsistent as 100 units are debited from account A but not credited in account B, this will generate inconsistency. Here for ‘consistency’ before $(A + B) ==$ after $(A + B)$ ”.
- To remove this partial execution problem, we increase the level of atomicity and bundle all the instruction of a logical operation into a unit called transaction. So formally ‘A transaction is a Set of logically related instructions to perform a logical unit of work’.

T_1
Read(A)
$A = A - 100$
Write(A)
Read(B)
$B = B + 100$
Write(B)



- As here we are only concerned with DBMS so we well only two basic operation on database
- **READ (X)** - Accessing the database item x from disk (where database stored data) to memory variable also name as X.
- **WRITE (X)** - Writing the data item from memory variable X to disk.

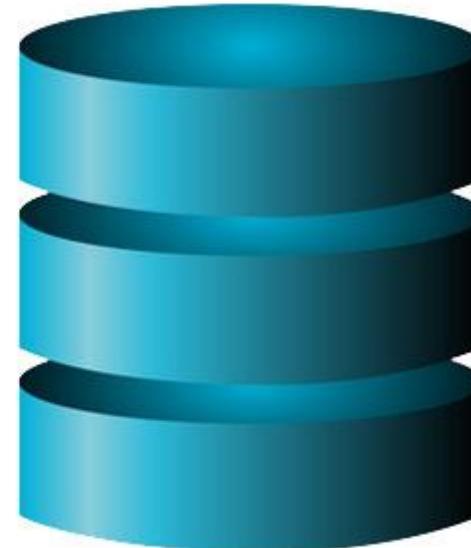
Q A transaction can include following basic database access operations:

- (A)** Read_item(X)
- (B)** Write_item(X)
- (C)** Both (A) and (B)
- (D)** None of these

Desirable properties of transaction

- Now as the smallest unit which have atomicity in DBMS view is transaction, so if want that our data should be consistent then instead of concentrating on data base, we must concentrate on the transaction for our data to be consistent.
- Transactions should possess several properties, often called the **ACID** properties; to provide integrity and consistency of the data in the database. The following are the ACID properties:

T_1
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)



- **Atomicity** - A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all. It is the responsibility of *recovery control manager / transaction control manager of DBMS* to ensure atomicity.
- **Consistency** - A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another. The definition of consistency may change from one system to another. The preservation of consistency of database is the responsibility of programmers(users) or the DBMS modules that enforces integrity constraints.
- **Isolation** - A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently. The isolation property of database is the responsibility of *concurrency control manager of database*.
- **Durability** - The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure. It is the responsibility of *recovery control manager of DBMS*.

Q NOT a part of the **ACID** properties of database transactions?

(a) Atomicity

(b) Consistency

(c) Isolation

(d) Deadlock-freedom

Q All Oracle transactions obey the basic properties of a database transaction. What is the name of the following property? ‘All tasks of a transaction are performed or none of them are. There are no partial transactions.’ [Asked in Wipro NLTH]

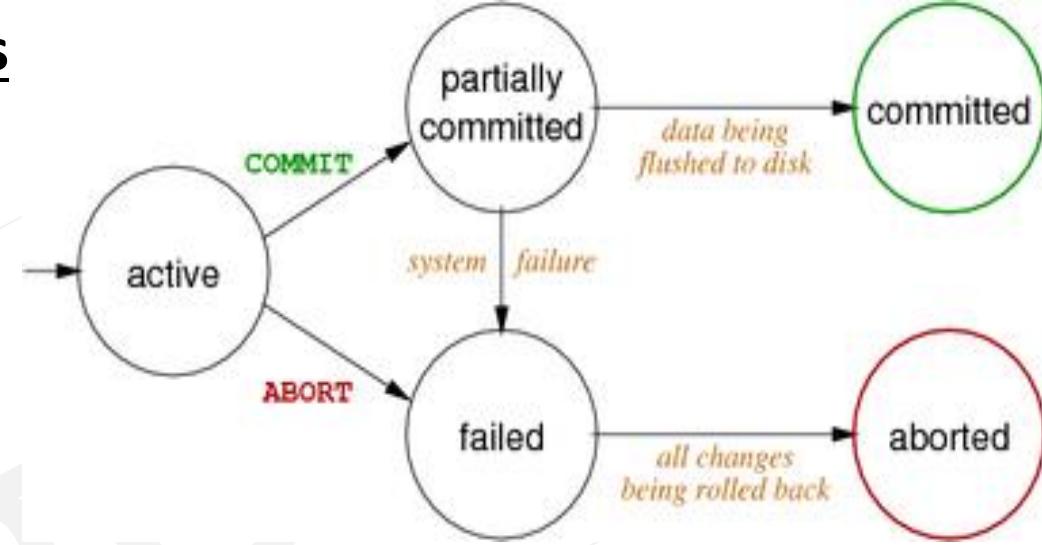
- a) Atomicity
- b) Durability
- c) Consistency
- d) Isolation

Q) Sunil is working on a database management system. He wants to program transactions such that a transaction is completed only if the entire database operations are completed and committed, otherwise the transaction is aborted and rolled back. Which of the following database characteristics is he trying to implement? [Amazon].

- (a) Atomicity
- (b) Consistency
- (c) Isolation
- (d) Durability

Transaction states

- **ACTIVE** - It is the initial state. Transaction remains in this state while it is executing operations.
- **PARTIALLY COMMITTED** - After the final statement of a transaction has been executed, the state of transaction is partially committed as it is still possible that it may have to be aborted (due to any failure) since the actual output may still be temporarily residing in main memory and not to disk.
- **FAILED** - After the discovery that the transaction can no longer proceed (because of hardware /logical errors). Such a transaction must be rolled back.
- **ABORTED** - A transaction is said to be in aborted state when the when the transaction has been rolled back and the database has been restored to its state prior to the start of execution.
- **COMMITTED** - A transaction enters committed state after successful completion of a transaction and final updation in the database.



Q) Rollback is used to restore the data _____ [Amcat].

- (a) Until the location the first commit was performed.**
- (b) Until the location the last commit was performed.**
- (c) Until the location the second commit was performed.**
- (d) All of the above.**

Q) A transaction completes its execution is said to be.[Amcat].

(a) Saved

(b) Loaded

(c) Rolled

(d) Committed

Ans: d

Q if the transaction is in which of the state that we can guarantee that data base is in consistent state

- a) aborted
- b) committed
- c) both aborted & committed
- d) none

Why we need concurrent execution

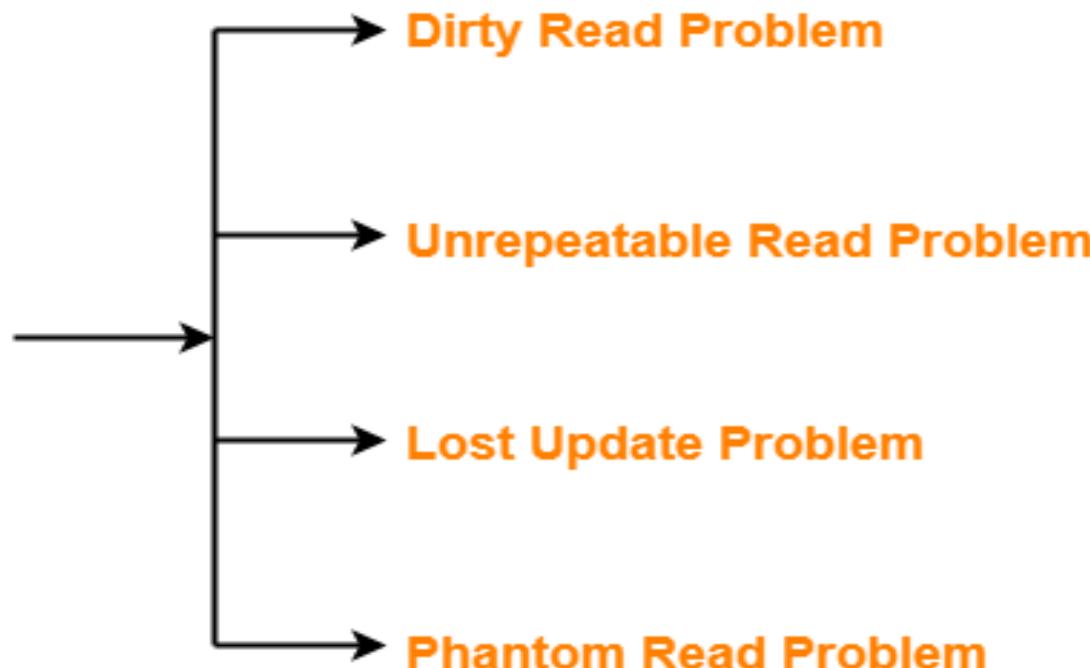
- Concurrent execution is necessary because-
 - It leads to good database performance , less weighting time.
 - Overlapping I/O activity with CPU increases throughput and response time.



PROBLEMS DUE TO CONCURRENT EXECUTION OF TRANSACTION

- But interleaving of instructions between transactions may also lead to many problems that can lead to inconsistent database.
- Sometimes it is possible that even though individual transaction are satisfying the acid properties even though the final statuses of the system will be inconsistent.

**Concurrency Problems
in
Transactions**



Lost update problem / Write - Write problem-

- If there is any two write operation of different transaction on same data value, and between them there is no read operations, then the second write over writes the first write.

T_1	T_2
Read(A)	
Write(A)	
	Write(A)
	Commit
Commit	

Dirty read problem/ Read -Write problem

- In this problem, the transaction reads a data item updated by another uncommitted transaction, this transaction may in future be aborted or failed.
- The reading transactions end with incorrect results.

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Commit
Abort	

Unrepeatable read problem

- When a transaction tries to read a value of a data item twice, and another transaction updates the data item in between, then the result of the two read operation of the first transaction will differ, this problem is called, Non-repeatable read problem

T_1	T_2
Read(A)	
	Read(A)
	Write(A)
Read(A)	

Phantom read problem

T_1	T_2
Read(A)	
	Delete(A)
Read(A)	

Q Which of the following scenarios may lead to an irrecoverable error in a database system?

- (A)** A transaction writes a data item after it is read by an uncommitted transaction
- (B)** A transaction reads a data item after it is read by an uncommitted transaction
- (C)** A transaction reads a data item after it is written by a committed transaction
- (D)** A transaction reads a data item after it is written by an uncommitted transaction

Solution is Schedule

- When two or more transaction executed together or one after another then they can be bundled up into a higher unit of execution called schedule.
- A **schedule** of n transactions T_1, T_2, \dots, T_n is an ordering of the operations of the transactions. Operations from different transactions can be interleaved in the schedule S .
- However, schedule for a set of transaction must contain all the instruction of those transaction, and for each transaction T_i that participates in the schedule S , the operations of T_i in S must appear in the same order in which they occur in T_i .

- **Serial schedule** - A serial schedule consists of sequence of instruction belonging to different transactions, where instructions belonging to one single transaction appear together. Before complete execution of one transaction another transaction cannot be started.

T_0	T_1
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

- For a set of n transactions, there exist $n!$ different valid serial schedules. Every serial schedule lead database into consistent state. Throughput of system is less.

- **Non-serial schedule** - A schedule in which sequence of instructions of a transaction appear in the same order as they appear in individual transaction but the instructions may be interleaved with the instructions of different transactions i.e. concurrent execution of transactions takes place.

T_2	T_3
read(B)	
read(A)	read(B) write(B)

- So the number of schedules for n different transaction $T_1, T_2, T_3, \dots, T_N$ where each transaction contains $n_1, n_2, n_3, \dots, n_n$ respectively will be.
- $\{(n_1 + n_2 + n_3 + \dots + n_n)!\} / (n_1! n_2! n_3! \dots n_n!)\}$
- $\{(n_1 + n_2 + n_3 + \dots + n_n)!\} / (n_1! n_2! n_3! \dots n_n!) - n!$

- **Conclusion of schedules**
 - We do not have any method to proof that a schedule is consistent, but from the above discussion we understand that a serial schedule will always be consistent, so if somehow we proof that a non-serial schedule will also have same effects as of a serial schedule that we get a proof that, this particular non-serial schedule will also be consistent “find those schedules that are logically equal to serial schedules”.
 - For a concurrent schedule to result in consistent state, it should be equivalent to a serial schedule. i.e. it must be serializable.

On the basis of
SERIALIZABILITY

Conflict
serializable

View
serializable

Result
Equivalent

On the basis of
RECOVERABILITY

Recoverable

Cascadeless

Strict

T ₁	T ₂
R(A)	
	R(B)

T ₁	T ₂
	R(B)
R(A)	

T ₁	T ₂
R(A)	
	W(A)

T ₁	T ₂
	W(A)
R(A)	

T ₁	T ₂
R(A)	
	W(B)

T ₁	T ₂
	W(B)
R(A)	

T ₁	T ₂
	R(A)
W(A)	

T ₁	T ₂
W(A)	
	R(A)

T ₁	T ₂
R(A)	
	R(A)

T ₁	T ₂
	R(A)
R(A)	

T ₁	T ₂
	W(A)
W(A)	

T ₁	T ₂
W(A)	
	W(A)

SERIALIZABILITY

- **Conflicting instructions** - Let I and J be two consecutive instructions belonging to two different transactions T_i and T_j in a schedule S, the possible I and J instruction can be as-
 - $I = \text{READ}(Q), J = \text{READ}(Q) \rightarrow \text{Non-conflicting}$
 - $I = \text{READ}(Q), J = \text{WRITE}(Q) \rightarrow \text{Conflicting}$
 - $I = \text{WRITE}(Q), J = \text{READ}(Q) \rightarrow \text{Conflicting}$
 - $I = \text{WRITE}(Q), J = \text{WRITE}(Q) \rightarrow \text{Conflicting}$
- So, the instructions I and J are said to be conflicting, if they are operations by different transactions on the same data item, and at least one of these instructions is a write operation.

Q In the context of concurrency control, a given pair of operations in a schedule is called conflict schedule if

- a. At least one of the operations is write operation
- b. Both the operations are performed on the same data item
- c. Both the operations are performed by different transactions
- d. Both the operations are performed on different data items

Choose the correct answer from the options given below:

(A) (a) and (b) only

(B) (a), (b) and (c) only

(C) (a), (c) and (d) only

(D) (c) and (d) only

Ans:B

CONFLICT SERIALIZABLE

- The schedules which are conflict equivalent to a serial schedule are called conflict serializable schedule.
- If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are **conflict equivalent**.
- A schedule S is ***conflict serializable***, if it is conflict equivalent to a serial schedule.

T_1	T_2
read(A) write(A)	read(A) write(A)
read(B) write(B)	read(B) write(B)

T_1	T_2
read(A) write(A) read(B) write(B)	read(A) write(A) read(B) write(B)
	read(A) write(A) read(B) write(B)

Q Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item z by a transaction T_i . Consider the following two schedules.

- $S_1: r_1(x)r_1(y)r_2(x)r_2(y)w_2(y)w_1(x)$
- $S_2: r_1(x)r_2(x)r_2(y)w_2(y)r_1(y)w_1(x)$

Which one of the following options is correct?

- a) S_1 is conflict serializable, and S_2 is not conflict serializable
- b) S_1 is not conflict serializable, and S_2 is conflict serializable
- c) Both S_1 and S_2 are conflict serializable
- d) Neither S_1 nor S_2 is conflict serializable

S_1	
T_1	T_2
R(X)	
R(Y)	
	R(X)
	R(Y)
	W(Y)
W(X)	

S_2	
T_1	T_2
R(X)	
	R(X)
	R(Y)
	W(Y)
R(Y)	
	W(X)

Q Consider the following schedule for transactions T_1 , T_2 and T_3 :

Which one of the schedules below is the correct serialization of the above?

(A) $T_1 \rightarrow T_3 \rightarrow T_2$

(B) $T_2 \rightarrow T_1 \rightarrow T_3$

(C) $T_2 \rightarrow T_3 \rightarrow T_1$

(D) $T_3 \rightarrow T_1 \rightarrow T_2$

	<u>T1</u>	<u>T2</u>	<u>T3</u>
Read (X)			
Read (Y)			
Write (Y)			
Write (X)			
		Read (X)	
		Write (X)	

Q Consider the transactions T_1 , T_2 , and T_3 and the schedules S_1 and S_2 given below.

$T_1: r_1(X); r_1(Z); w_1(X); w_1(Z)$

$T_2: r_2(Y); r_2(Z); w_2(Z)$

$T_3: r_3(Y); r_3(X); w_3(Y)$

$S_1: r_1(X); r_3(Y); r_3(X); r_2(Y); r_2(Z); w_3(Y); w_2(Z); r_1(Z); w_1(X); w_1(Z)$

$S_2: r_1(X); r_3(Y); r_2(Y); r_3(X); r_1(Z); r_2(Z); w_3(Y); w_1(X); w_2(Z); w_1(Z)$

Which one of the following statements about the schedules is TRUE?

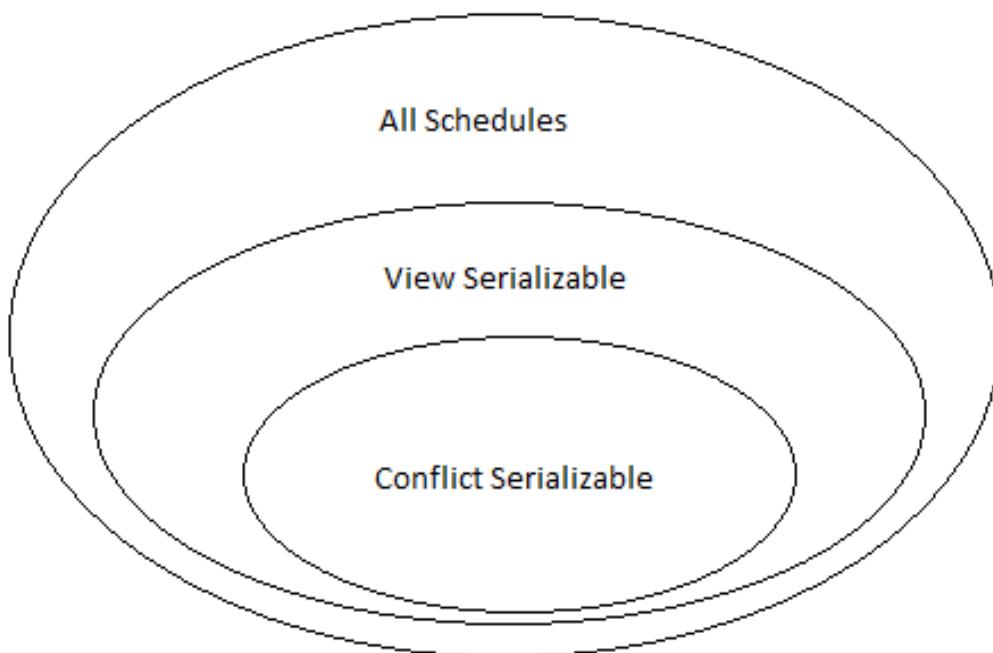
- (A) Only S_1 is conflict-serializable.
- (B) Only S_2 is conflict-serializable.
- (C) Both S_1 and S_2 are conflict-serializable.
- (D) Neither S_1 nor S_2 is conflict-serializable.

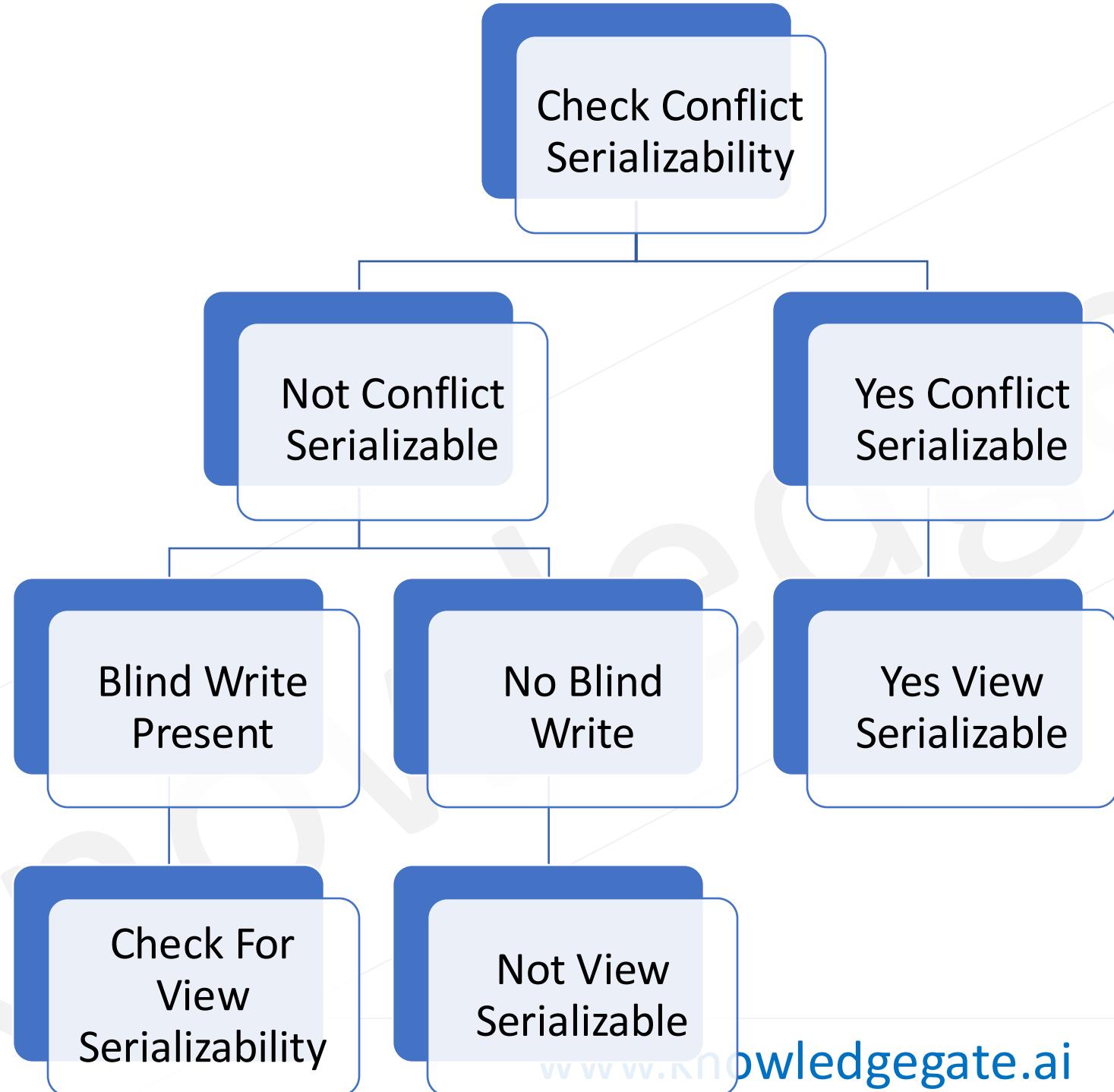
	S_1	
T_1	T_2	T_3
$R(X)$		
		$R(Y)$
		$R(X)$
	$R(Y)$	
	$R(Z)$	
		$W(Y)$
	$W(Z)$	
$R(Z)$		
$W(X)$		
$W(Z)$		

	S_2	
T_1	T_2	T_3
$R(X)$		
		$R(Y)$
	$R(Y)$	
		$R(X)$
$R(Z)$		
	$R(Z)$	
		$W(Y)$
$W(X)$		
	$W(Z)$	
$W(Z)$		

VIEW SERIALIZABLE

- **Why we need it** – View serializability captures schedules that are *not* conflict-serializable yet still behave correctly.
- **Relation to conflict serializability** – Every conflict-serializable schedule is view-serializable, so we test view serializability only if the conflict test fails.
- **Role of a blind write** – A non-conflict-serializable schedule without any blind write can never be view-serializable; having at least one blind write merely makes it possible.
- **How to test** – List all serial orders and see if the schedule matches one on first read, read-from, and final write; a match means the schedule is view-serializable.
- Complexity wise finding the schedule is view serializable or not is a **NP- complete** problem.





- Two schedules S and S' are view equivalent, if they satisfy following conditions –
 - For each data item Q, if the transaction T_i reads the initial value of Q in schedule S , then the transaction T_i must, in schedule S' ,also read the initial value of Q.
 - If a transaction T_i in schedule S reads any data item Q, which is updated by transaction T_j , then a transaction T_i must in schedule S' also read data item Q updated by transaction T_j in schedule S'.
 - For each data item Q, the transaction (if any) that performs the final write(Q) operation in schedule S, then the same transaction must also perform final write(Q) in schedule S'.

T_1	T_2
read(A) write(A)	read(A) write(A)
read(B) write(B)	read(B) write(B)

T_1	T_2
read(A) write(A) read(B) write(B)	
	read(A) write(A) read(B) write(B)

View Serializable

- A schedule S is view serializable, if it is view equivalent to a serial schedule.

Schedule A		
T3	T4	T6
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

View Serializable

A schedule S is view serializable, if it is view equivalent to a serial schedule.

Schedule A			Serial schedule <T3,T4,T6>		
T3	T4	T6	T3	T4	T6
read(Q)			read(Q)		
	write(Q)		write(Q)		
write(Q)				write(Q)	
		write(Q)			write(Q)

- **BLIND WRITES**- In the above example, transaction T_4 and T_6 perform write operation on data item Q without accessing (reading the data item), such updation without knowing/accessing previous value of data item, are called Blind updation or **BLIND WRITE**.

Q A Schedule that is not conflict serializable and contains at least one blind write then the schedule is

- a)** Always view serializable
- b)** Always non-serializable
- c)** May be View serializable
- d)** None of the above

Q Which of the following statements (s) is/are TRUE?

- S₁:** All view serializable schedules are also conflict serializable.
- S₂:** All conflict serializable schedules are also view serializable.
- S₃:** If a schedule is not conflict serializable then it is not view serializable
- S₄:** If a schedule is not view serializable then it is not conflict serializable.

- a)** S₁ and S₂ only
- b)** S₂ and S₃ only
- c)** S₂ and S₄ only
- d)** S₁ and S₃ only

- **NON- RECOVERABLE SCHEDULE**: A schedule in which for each pair of transaction T_i and T_j , such that if T_j reads a data item previously written by T_i , then the commit or abort operation of T_i appears before T_j . Such a schedule is called Non- Recoverable schedule.
- **RECOVERABLE SCHEDULE**: A schedule in which for each pair of transaction T_i and T_j , such that if T_j reads a data item previously written by T_i , then the commit or abort of T_i must appear before T_j . Such a schedule is called Recoverable schedule.

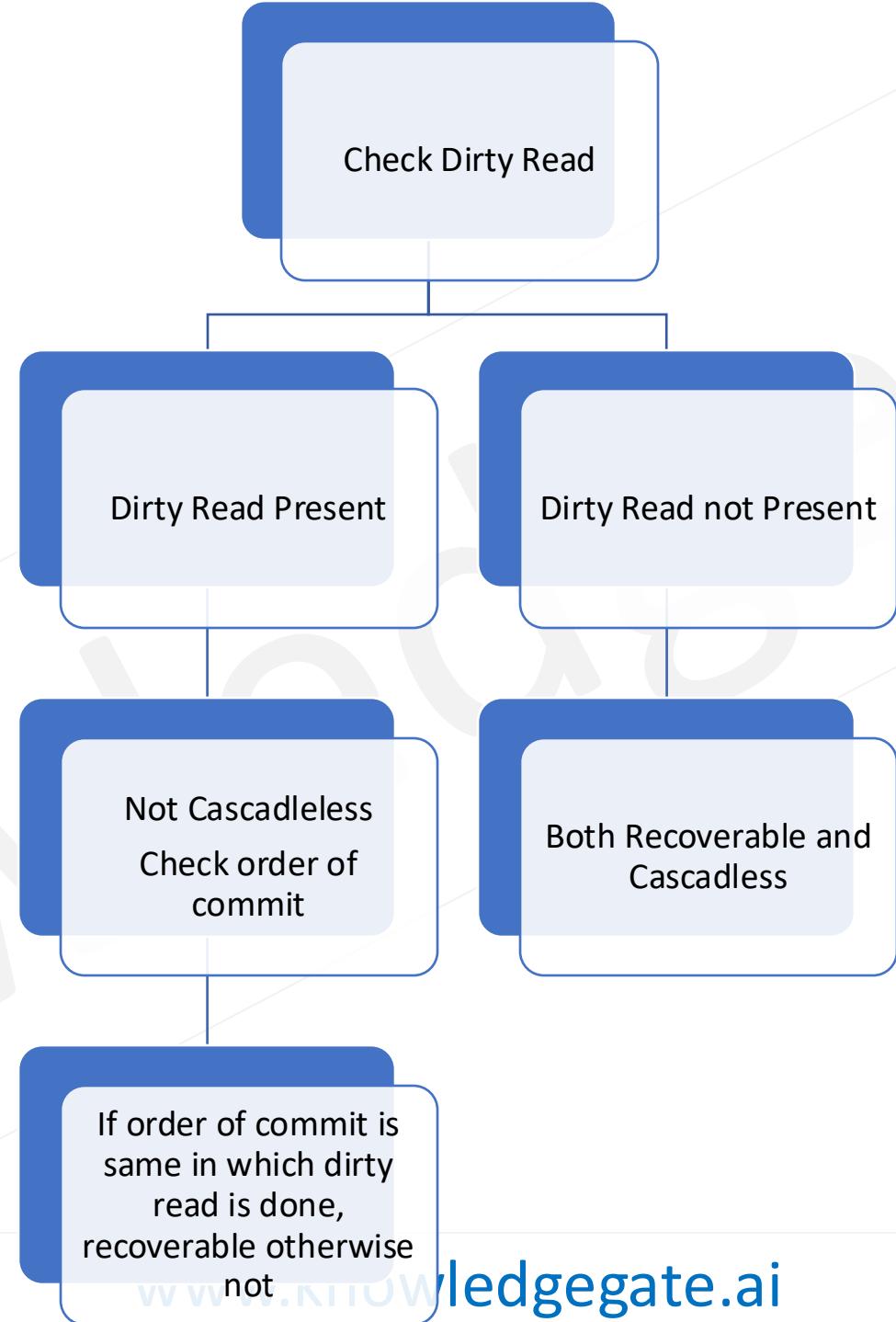
S	
T_1	T_2
R(X)	
W(X)	
	R(X)
	C
C	

S	
T_1	T_2
R(X)	
W(X)	
	R(X)
	C
C	

- **CASCADING ROLLBACK:** It is a phenomenon, in which a single transaction failure leads to a series of transaction rollbacks, is called cascading rollback. Even if the schedule is recoverable the, the commit of transaction may lead lot of transaction to rollback. Cascading rollback is undesirable, since it leads to undoing of a significant amount of work. Uncommitted reads are not allowed in cascade less schedule.
- **CASCADELESS SCHEDULE:** To avoid cascading rollback, cascade less schedule are used. A schedule in which for each pair of transactions T_i and T_j , such that if T_j reads a data item previously written by T_i then the commit or abort of T_i must appear before read operation of T_j . Such a schedule is called cascade less schedule.

S		
T_1	T_2	T_3
R(X)		
W(X)		
	R(X)	
	W(X)	
		R(X)
C		
	C	
		C

S		
T_1	T_2	T_3
R(X)		
W(X)		
C		
	R(X)	
	W(X)	
	C	
		R(X)
		C



**Q) A single transaction failure may result into a set of transaction rollbacks is known to be.
[Amcat].**

- (a) Iterated rollback**
- (b) Cascadeless rollback**
- (c) Serial rollback**
- (d) Cascading rollback**

Ans: d

CONCURRENCY CONTROL

- Now we understood that if there is a schedule how to check whether it will work correctly or not i.e. weather it will maintain the consistency of the data base or not. (conflict serializability, view serializability, recoverability and cascade less). Now we will understand those protocol which guarantee how to design those schedules which ensure conflict serializability or other properties.
- There are different approach or idea to ensure conflict serializability which is the most important property. So first we must understand what is the possibility of conflict between two instruction and if somehow, we manage than the generated schedule will always be conflict serializable.
- If we remember, two instructions are conflicting if and only if three things happen simultaneously
 - Belong to different transactions
 - Must operate on same data value
 - At least one of them should be a write instruction
- if we think sufficiently, there will be no way to change any of these three conditions.
But actual problem is not that two instructions are trying to access same data base but, they are trying to do that at same time. So point if we somehow by any technique manage that two transaction do not access same data at same time then ensuring conflict serializability will be easy. Now question is how to approach conflict serializability, there are Three popular approaches to go forwards.

- **Time stamping based method:** - where before entering the system, a specific order is decided among the transaction, so in case of a clash we can decide which one to allow and which to stop.
- **Lock based method:** - where we ask a transaction to first lock a data item before using it. So that no different transaction can use a data at the same time, removing any possibility of conflict.
 - 2 phase locking
 - Basic 2pl
 - Conservative 2pl
 - Rigorous 2pl
 - Strict 2pl
 - Graph based protocol
- **Validation based protocol** – Majority of transactions are read only transactions, the rate of conflicts among the transaction may be low, thus many of transaction, if executed without the supervision of a concurrency control scheme, would nevertheless leave the system in a consistent state.

Goals of a Protocol: - We desire the following properties from schedule generating protocols

- Concurrency should be as high as possible, as this is our ultimate goal because of which we are making all the effort.
- The time taken by a transaction should also be less.
- Desirable Properties satisfied by the protocol
- Easy to understand and implement

Q) Serializability of schedules can be ensured through a mechanism called.[Amcat].

- (a) Concurrency control policy**
- (b) Evaluation control policy**
- (c) Execution control policy**
- (d) Cascading control policy**

Q) Characteristics of Good Concurrency Protocol. [Amcat]

- (a)** It allows the parallel execution of transactions to achieve maximum concurrency.
- (b)** Its storage mechanisms and computational methods should be modest to minimize overhead.
- (c)** It must enforce some constraints on the structure of atomic actions of transactions.
- (d)** All of the Above

TIME STAMP ORDERING PROTOCOL

- Basic idea of time stamping is to decide the order between the transaction before they enter in the system using a stamp (time stamp), in case of any conflict during the execution order can be decided using the time stamp.
- **Each transaction T gets a unique, fixed stamp $TS(T)$** when it starts (system-clock value \Rightarrow always increasing & unique). Serial-order rule: if $TS(T_1) < TS(T_2)$, the executed schedule must be equivalent to T_1 before T_2 .
- For **every data item Q** the protocol keeps two stamps:
 - **W-TS(Q)** – largest stamp of any transaction that *successfully wrote* Q
 - **R-TS(Q)** – largest stamp of any transaction that *successfully read* Q

- Suppose a transaction T_i request a ***read(Q)***
 - If $TS(T_i) < W\text{-timestamp}(Q)$, then T_i needs to read a value of Q that was already overwritten. Hence, the read operation is rejected, and T_i is rolled back.
 - If $TS(T_i) \geq W\text{-timestamp}(Q)$, then the read operation is executed, and $R\text{-timestamp}(Q)$ is set to the maximum of $R\text{-timestamp}(Q)$ and $TS(T_i)$.
- Suppose that transaction T_i issues ***write(Q)***.
 - If $TS(T_i) < R\text{-timestamp}(Q)$, then the value of Q that T_i is producing was needed previously, and the system assumed that that value would never be produced. Hence, the write operation is rejected, and T_i is rolled back.
 - If $TS(T_i) < W\text{-timestamp}(Q)$, then T_i is attempting to write an obsolete value of Q . Hence, this write operation is rejected, and T_i is rolled back.
 - If $TS(T_i) \geq R\text{-timestamp}(Q)$, then the write operation is executed, and $W\text{-timestamp}(Q)$ is set to $\max(W\text{-timestamp}(Q), TS(T_i))$.
 - If $TS(T_i) \geq W\text{-timestamp}(Q)$, then the write operation is executed, and $W\text{-timestamp}(Q)$ is set to $\max(W\text{-timestamp}(Q), TS(T_i))$.

Properties

- Time stamp ordering protocol ensures conflict serializability. Because conflicting operations are processed in timestamp order, since all the arcs in the precedence graph are of the form thus, there will be no cycles in the precedence graph.
- As we know that view is liberal form conflict so view serializability also holds good
- As there is a possibility of dirty read, and no restriction on when to commit, so can be irrecoverable and may suffer from cascading rollback.
- At the time of request, here either we allow or we reject, so there is no idea of deadlock.
- If a schedule is not conflict serializable then it is not allowed by time stamp ordering scheme.
- But it is not necessary that all conflict serializable schedule generated by time stamping.

	Conflict Serializability	View Serializability	Recoverability	Cascadelessness	Deadlock Freedom
Time Stamp Ordering	YES	YES	NO	NO	YES
Thomas Write Rule	NO	YES	NO	NO	YES
Basic 2PL					
Conservative 2PL					
Rigorous 2PL					
Strict 2PL					

Lock Based Protocols

- To ensure isolation is to require that data items be accessed in a mutually exclusive manner i.e. while one transaction is accessing a data item, no other transaction can modify that data item. Locking is the most fundamental approach to ensure this. Lock based protocols ensure this requirement. Idea is first obtain a lock on the desired data item then if lock is granted then perform the operation and then unlock it. In general, we support two modes of lock because, to provide better concurrency.
- **Shared mode:** If transaction T_i has obtained a shared-mode lock (denoted by S) on any data item Q, then T_i can read, but cannot write Q, any other transaction can also acquire a shared mode lock on the same data item(this is the reason we called this shared mode).
- **Exclusive mode:** If transaction T_i has obtained an exclusive-mode lock (denoted by X) on any data item Q, then T_i can both read and write Q, any other transaction cannot acquire either a shared or exclusive mode lock on the same data item. (this is the reason we called this exclusive mode)

Lock –Compatibility Matrix

- Conclusion shared is compatible only with shared while exclusive is not compatible either with shared or exclusive.
- To access a data item, transaction T_i must first lock that item, if the data item is already locked by another transaction in an incompatible mode, or some other transaction is already waiting in non-compatible mode, then concurrency control manager will not grant the lock until all incompatible locks held by other transactions have been released. The lock is then granted.

		Current State of lock of data items		
		Exclusive	Shared	Unlocked
Requested Lock	Exclusive	N	N	Y
	Shared	N	Y	Y
	Unlock	Y	Y	-

Q) If a transaction obtains an exclusive lock on a row, it means that the transaction wants to that row.[Amcat].

(a) Select

(b) Update

(c) View

(d) Read

Q) A _____ lock is also called a Read-only lock. [Amcat].

(a) Shared Lock

(b) Exclusive Lock

(c) Simplistic Lock Protocol

(d) Pre-claiming Locking

- Lock based protocol do not ensure serializability as granting and releasing of lock do not follow any order and any transaction any time may go for lock and unlock. Here in the example below we can see, that even this transaction in using locking but neither it is conflict serializable nor independent from deadlock.
- If we do not use locking, or if we unlock data items too soon after reading or writing them, we may get inconsistent states, as there exists a possibility of dirty read. On the other hand, if we do not unlock a data item before requesting a lock on another data item, concurrency will be poor.
- We shall require that each transaction in the system follow a set of rules, called a **locking protocol**, indicating when a transaction may lock and unlock each of the data items for e.g. 2pl or graph based locking.
- Locking protocols restrict the number of possible schedules.

T_1	T_2
LOCK-X(A)	
READ(A)	
WRITE(A)	
UNLOCK(A)	
	LOCK-S(B)
	READ(B)
	UNLOCK(B)
LOCK-X(B)	
READ(B)	
WRITE(B)	
UNLOCK(B)	
	LOCK-S(A)
	READ(A)
	UNLOCK(A)

Two phase locking protocol(2PL)

- The protocol ensures that each transaction issue lock and unlock requests in two phases, note that each transaction will be 2 phased not schedule.
- **Growing phase**- A transaction may obtain locks, but not release any locks.
- **Shrinking phase**- A transaction may release locks, but may not obtain any new locks.
- Initially a transaction is in growing phase and acquires lock as needed and in between can perform operation reach to lock point and once a transaction releases a lock, it can issue no more lock requests i.e. it enters the shrinking phase.

T ₁	T ₂
LOCK-X(A)	
READ(A)	
WRITE(A)	
	LOCK-S(B)
	READ(B)
LOCK-X(B)	
READ(B)	
WRITE(B)	
	LOCK-S(A)
	READ(A)
	UNLOCK(B)
UNLOCK(A)	
UNLOCK(B)	
	UNLOCK(A)

	Conflict Serializability	View Serializability	Recoverability	Cascadelessness	Deadlock Freedom
Time Stamp Ordering	YES	YES	NO	NO	YES
Thomas Write Rule	NO	YES	NO	NO	YES
Basic 2PL	YES	YES	NO	NO	NO
Conservative 2PL					
Rigorous 2PL					
Strict 2PL					

Properties

- 2PL ensures conflict serializability, and the ordering of transaction over lock points is itself a serializability order of a schedule in 2PL.
- If a schedule is allowed in 2PL protocol then definitely it is always conflict serializable. But it is not necessary that if a schedule is conflict serializable then it will be generated by 2PL. Equivalent serial schedule is based on the order of lock points.
- View serializability is also guaranteed.
- Does not ensure freedom from deadlock
- May cause non-recoverability.
- Cascading rollback may occur.

Q Which of the following concurrency protocol ensures both conflict serializability and freedom from deadlock?

- (1)** 2 - phase Locking **(2)** Time stamp - ordering
- (a)** Both (1) and (2)
(c) (2) only **(b)** (1) only
(d) Neither (1) nor (2)

Q) A system is in a _____ state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set. [Amcat].

- (a) Idle
- (b) Waiting
- (c) Deadlock
- (d) Ready

- **Conservative 2PL**: The idea is there is no growing phase transaction start directly from lock point, i.e. transaction must first acquire all the required locks then only it can start execution. If all the locks are not available then transaction must release the acquired locks and must wait. Shrinking phase will work as usual, and transaction can unlock any data item anytime. we must have a knowledge in future to understand what is data required so that we can use it

Q In conservative two phase locking protocol, a transaction?

- a)** Should release exclusive locks only after the commit operation
- b)** Should release all the locks only at beginning of the transaction
- c)** should acquire all the locks at beginning of the transaction
- d)** Should acquire all the exclusive locks at beginning transaction

	Conflict Serializability	View Serializability	Recoverability	Cascadelessness	Deadlock Freedom
Time Stamp Ordering	YES	YES	NO	NO	YES
Thomas Write Rule	NO	YES	NO	NO	YES
Basic 2PL	YES	YES	NO	NO	NO
Conservative 2PL	YES	YES	NO	NO	YES
Rigorous 2PL					
Strict 2PL					

- **RIGOROUS 2PL**: Requires that all locks be held until the transaction commits. This protocol requires that locking be two phase and also all the locks taken be held by transaction until that transaction commit. Hence there is no shrinking phase in the system.
- **STRICT 2PL**: that all exclusive-mode locks taken by a transaction be held until that transaction commits. This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data. This protocol requires that locking be two phase and also that exclusive –mode locks taken by transaction be held until that transaction commits. So it is simplified form of rigorous 2pl

Q In a Rigorous 2 phase protocol

- a)** All shared locks held by the transaction are released after the transaction is committed
- b)** All exclusive locks held by the transaction are released after the transaction is committed
- c)** All locks held by the transaction are released after the transaction is committed
- d)** All locks held by the transaction are released before the transaction is committed

	Conflict Serializability	View Serializability	Recoverability	Cascadelessness	Deadlock Freedom
Time Stamp Ordering	YES	YES	NO	NO	YES
Thomas Write Rule	NO	YES	NO	NO	YES
Basic 2PL	YES	YES	NO	NO	NO
Conservative 2PL	YES	YES	NO	NO	YES
Rigorous 2PL	YES	YES	YES	YES	NO
Strict 2PL	YES	YES	YES	YES	NO

Q Consider the following two statements about database transaction schedules:

- I.** Strict two-phase locking protocol generates conflict serializable schedules that are also recoverable.
- II.** Timestamp-ordering concurrency control protocol with Thomas Write Rule can generate view serializable schedules that are not conflict serializable.

Which of the above statements is/are TRUE?

- (a)** Both I and II
- (b)** Neither I nor II
- (c)** II only
- (d)** I only

Q Which of the following statement is/are correct

- a)** Every conflict serializable schedule allowed under 2PL protocol is allowed by basic time stamping protocol.
- b)** Every schedule allowed under basic time stamping protocol is allowed by Thomas-write rule
- c)** Every schedule allowed under Thomas-write rule is allowed by basic time stamping protocol
- d)** none