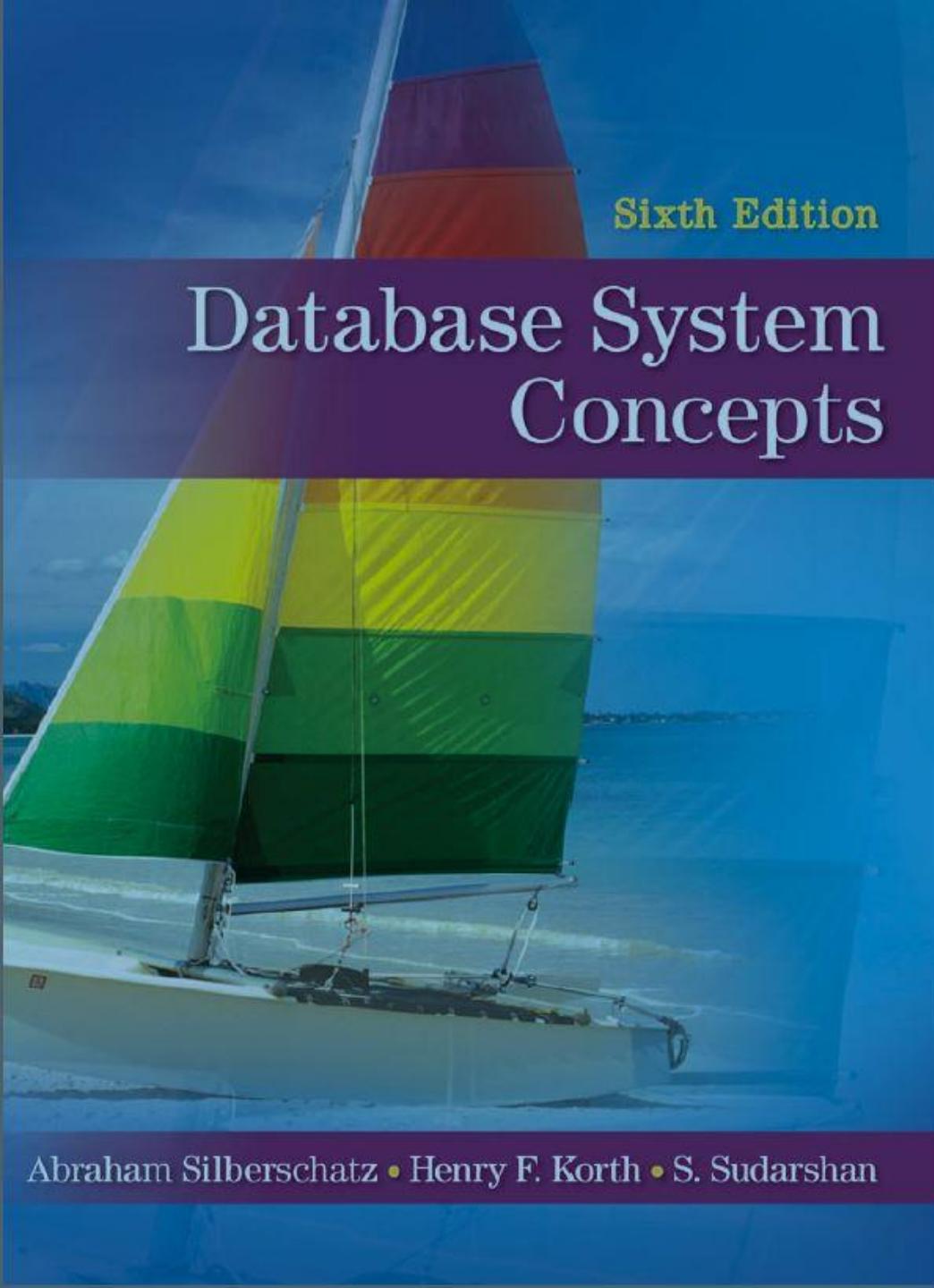
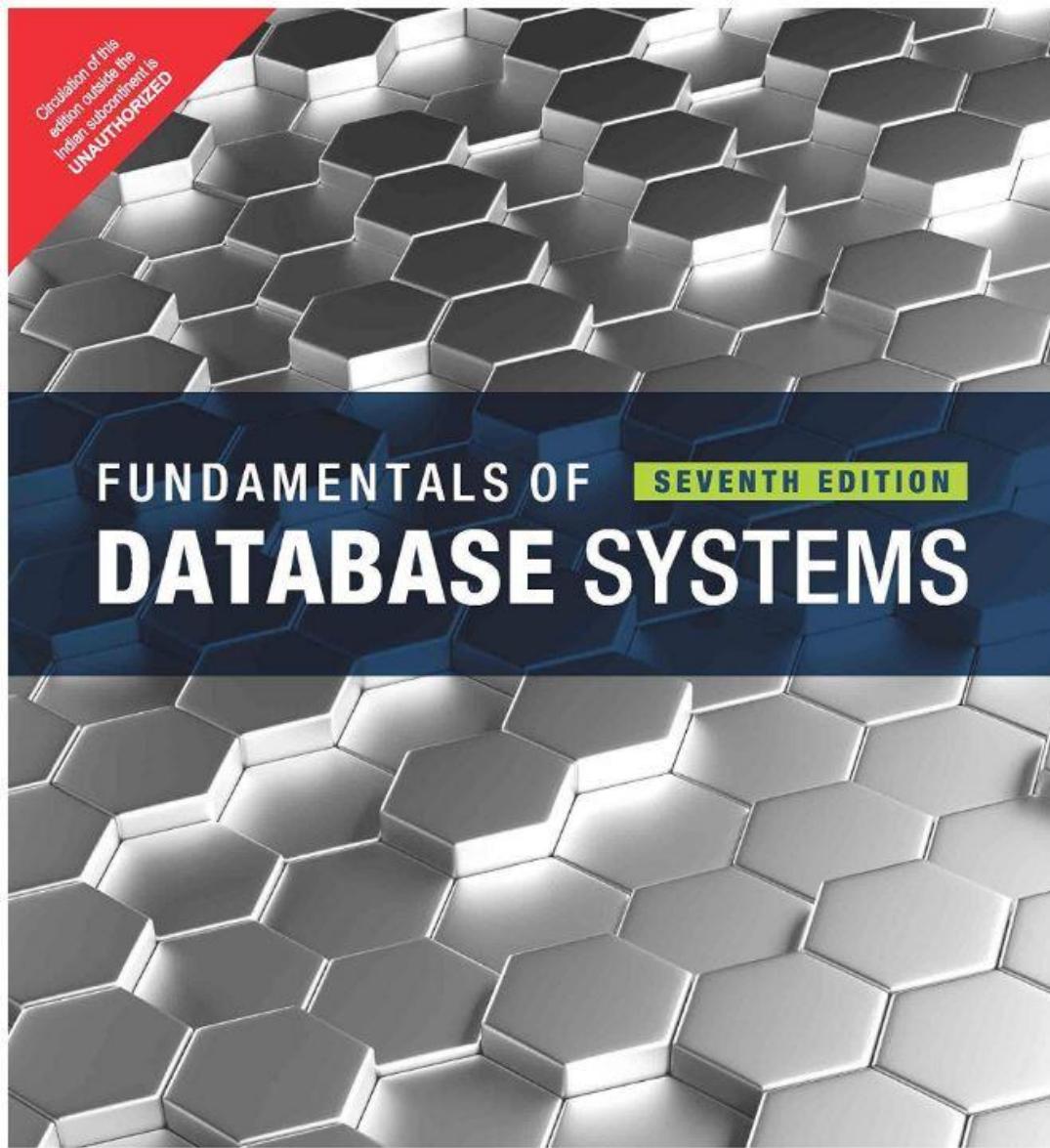


Database

- Core subjects for CS/IT Students
- In GATE 7-8 Marks out of 100 Marks, and 5-6 questions on an average
- Needs less time, very scoring
- Mostly numerical questions
- Indirect Application in Industry, good future



- **Book Name**
 - DATABASE SYSTEM CONCEPTS
- **Writers**
 - Abraham Silberschatz
 - Henry F. Korth
 - S. Sudarshan
- **Publisher – McGraw-Hill**
- **Edition – 6th**



- Book Name
 - Fundamentals of Database Systems
- Writers
 - Ramez Elmasri
 - Shamkant B. Navathe
- Publisher – Pearson
- Edition – 7th

Syllabus

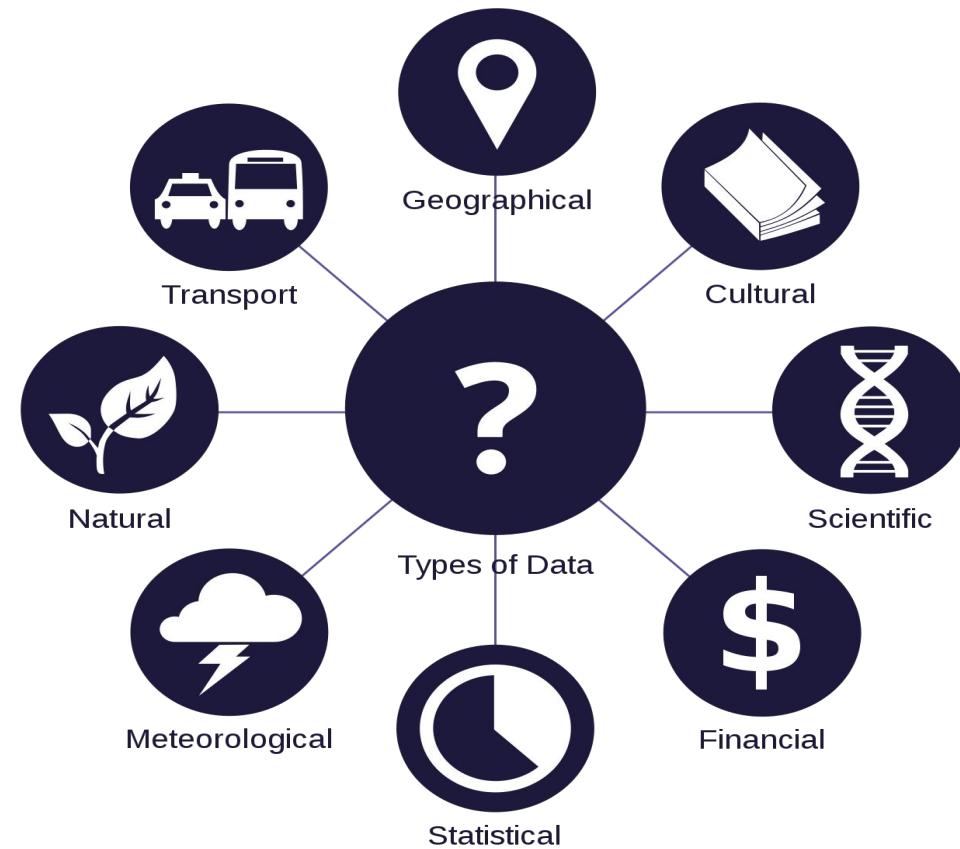
- ER-model
- Relational model
- Integrity constraints
- Normal forms
- File organization, indexing (e.g., B and B+ trees).
- Relational algebra, tuple calculus, SQL.
- Transactions and concurrency control.

What is Data ?



Data

- Data are characteristics, usually numerical, that are collected through observation. In a more technical sense, data are a set of values of **qualitative or quantitative variables** about one or more persons or objects, while a datum (singular of data) is a single value of a single variable.
- Any facts and figures about an entity is called as Data.



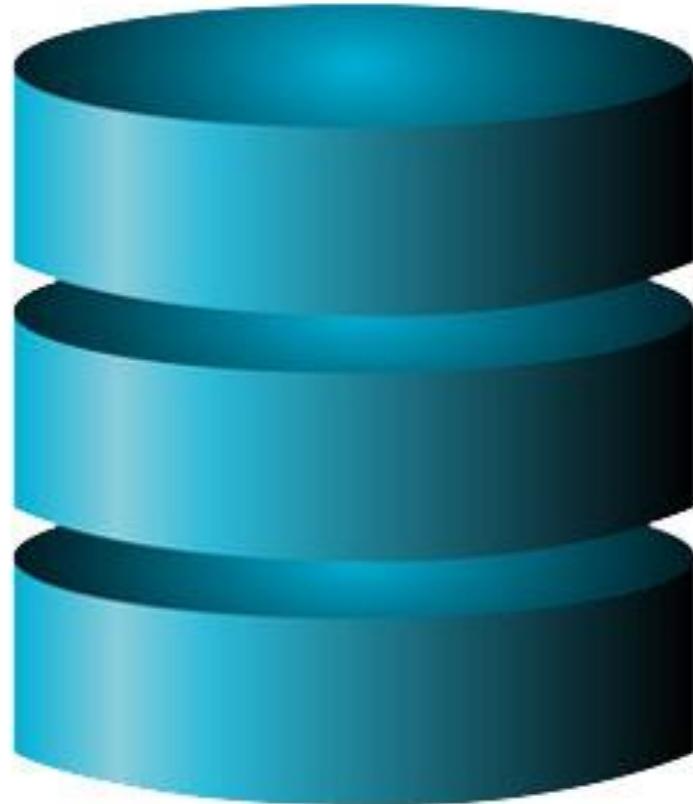
What is Information ?



Information

- Although the terms "data" and "information" are often used interchangeably, these terms have distinct meanings. In some popular publications, data are sometimes said to be transformed into information when they are viewed in context or in post-analysis.
- Processed Data is called information.

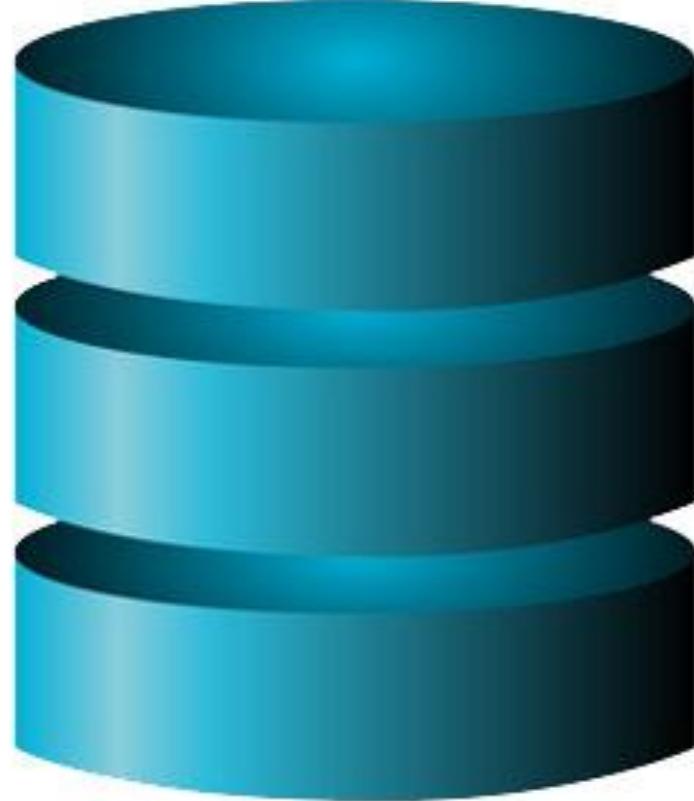
What is Data Base?



Data Base

- A **database** is an organized collection of data, generally stored and accessed electronically from a computer system.

What is Data Base Management System?



Data Base Management System

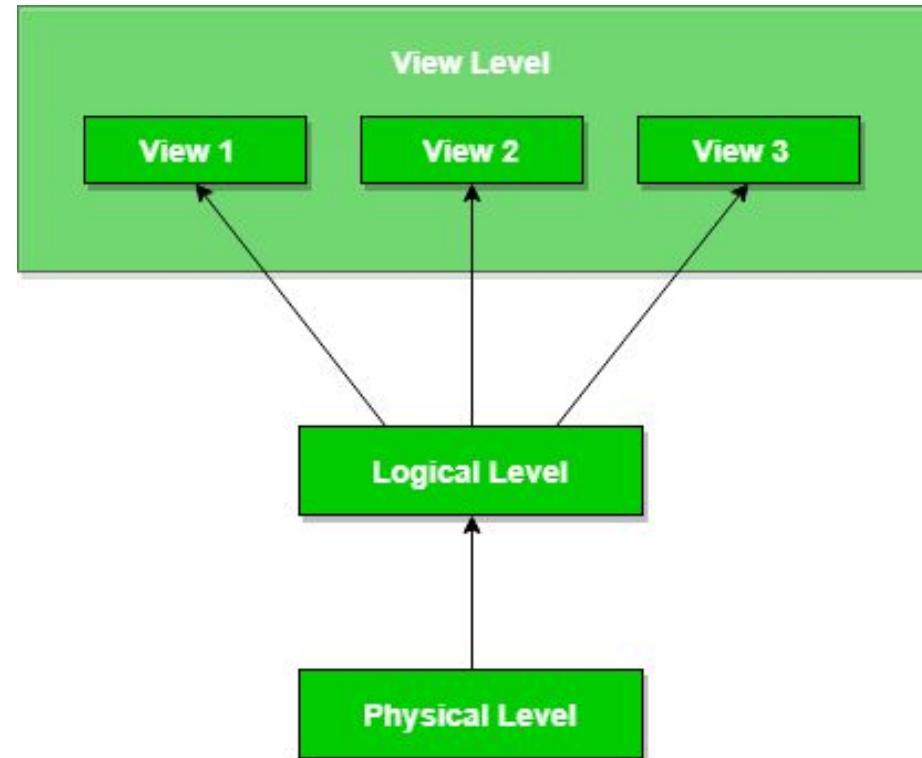
- The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data.
- The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system".

Problem With File System

- Data redundancy and inconsistency
- Difficulty in accessing Data
- Data Isolation
- Integrity Problem
- Atomicity Problem
- Concurrent access Anomalies

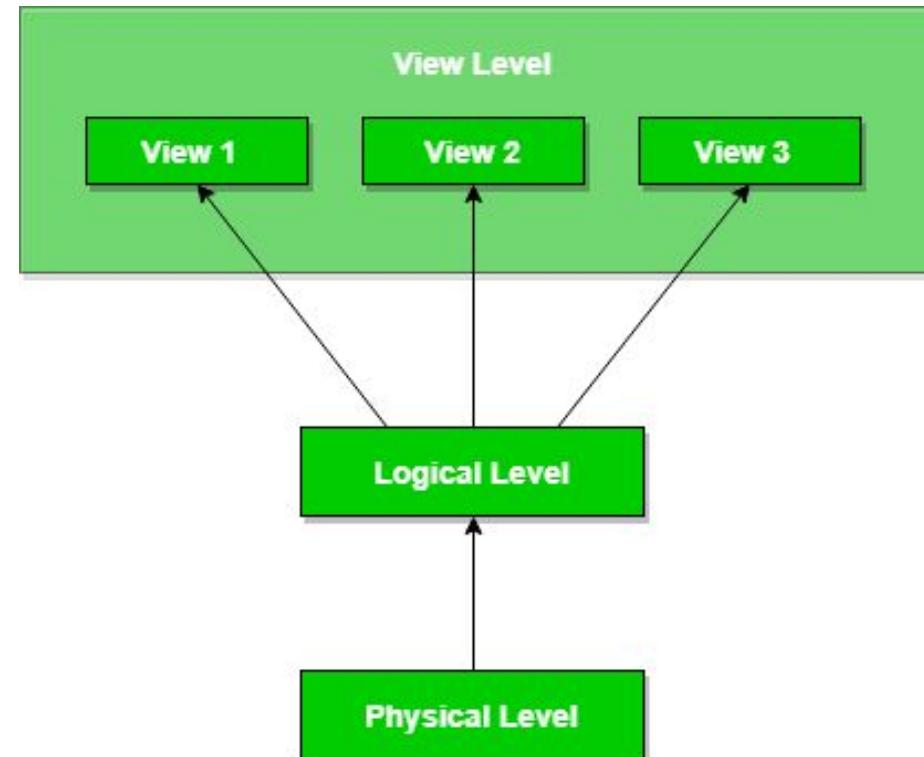
View of Data Base (Physical Level)

- The physical or the internal level schema describes how the data is stored in the hardware.
- It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has complex data structures. Only the database administrator operates at this level.



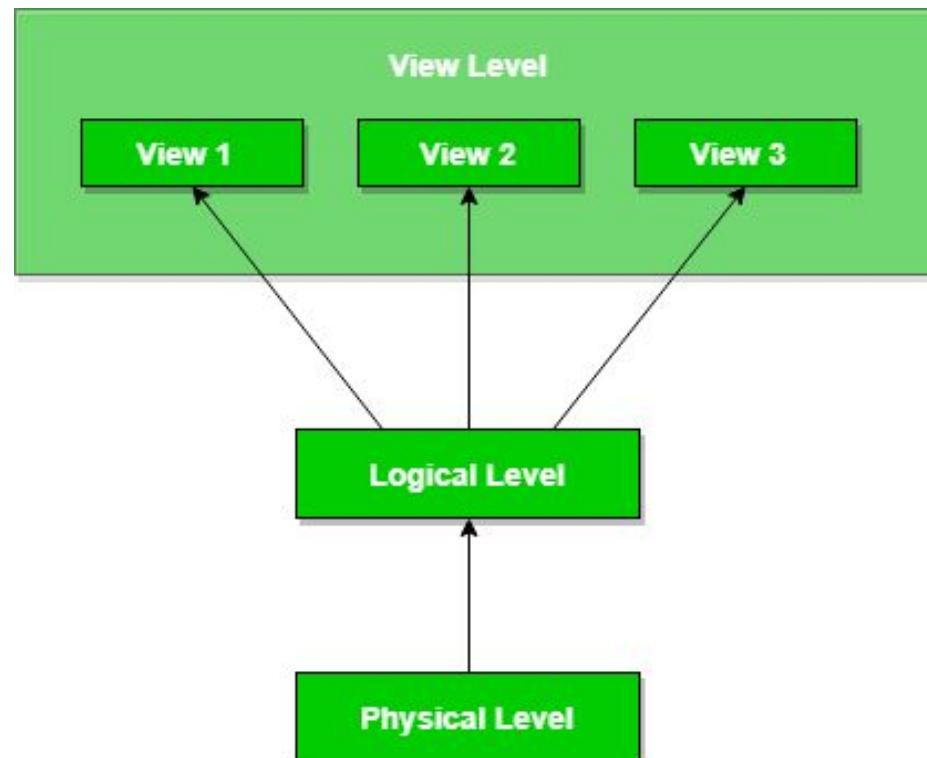
Logical Level/ Conceptual Level

- It is a level above the physical level. Here, the data is stored in the form of the entity set, entities, their data types, the relationship among the entity sets.
- The next-higher level of abstraction describes what data are stored in the database, what relationships exist among those data.



View Level

- It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested.
- The view level can describe many views of the same data. Here, the user retrieves the information using different application from the database.



Instance and Schemas

- The collection of information stored in the database at a particular moment is called an instance of the database.
- The overall design of the database is called the database schema.

OLAP Vs OLTP

OLAP	OLTP
✓ Gives a multi-dimensional view of business activities.	✓ Enables a snapshot of ongoing business processes.
✓ Helps with planning, problem solving, and decision support.	✓ Useful for controlling and running fundamental business tasks.
✓ Data source is consolidated data	✓ Data source is the operational data.
✓ Includes Periodic long-running batch jobs that refresh the data.	✓ Has short and fast inserts and updates which are initiated by end users.
✓ OLAP applications are widely used by Data Mining techniques.	✓ Large number of short on-line transactions
✓ Database design is typically de-normalized and contains fewer tables.	✓ Database design in OLTP is highly normalized.
✓ Often involves complex queries along with aggregations, which in turn compels processing speed to be dependent on the amount of data involved; batch data refreshes, etc.	✓ Involves standardized and simple queries that return relatively few records hence is faster.

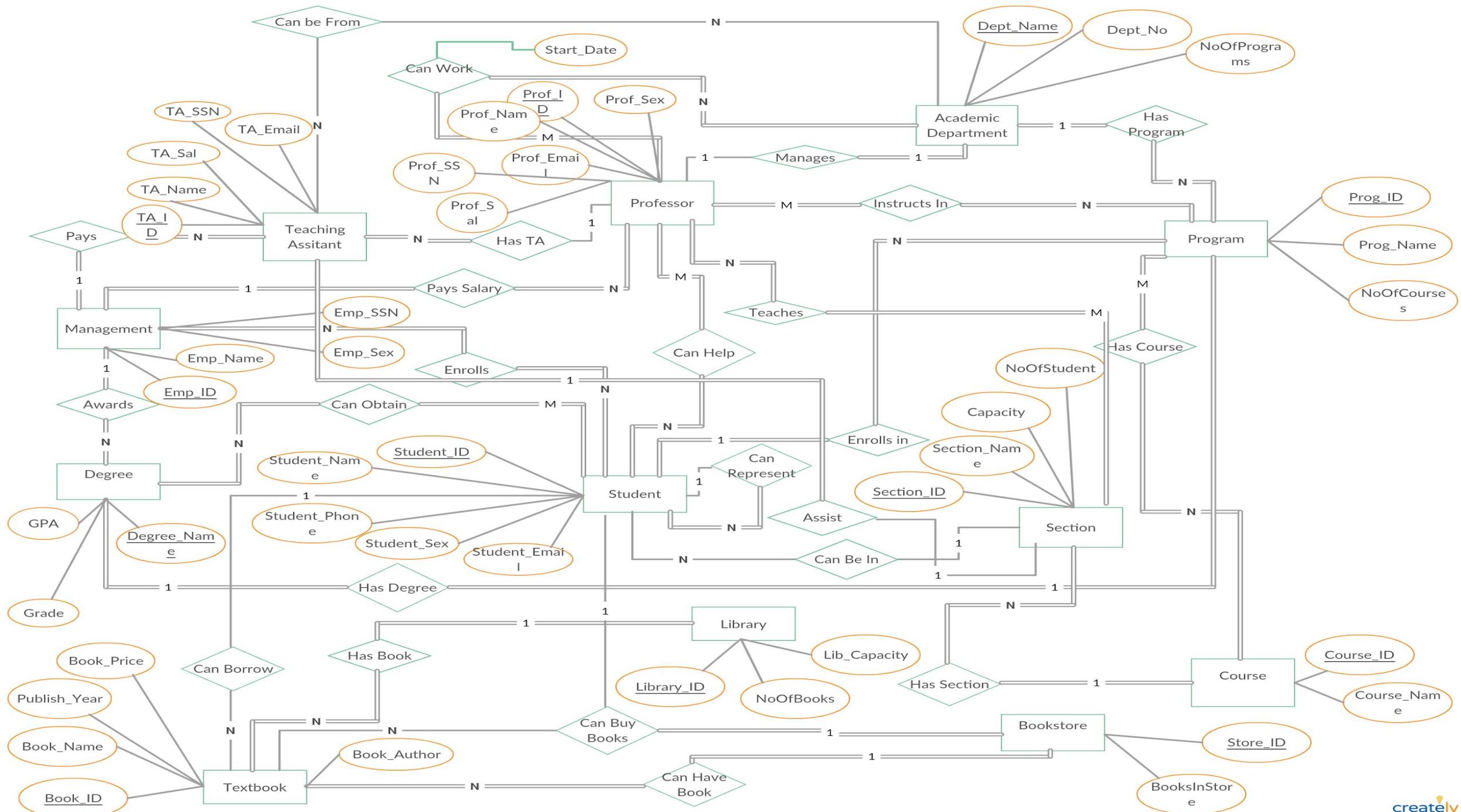
DBA



E-R DIAGRAM/MODEL

- Introduced in 1976 by Dr Peter Chen, a non-technical design method works on conceptual level based on the perception of the real world.
- E-R data model was developed to facilitate database design by allowing specification of an enterprise schema that represents ***the overall logical structure of a database.***

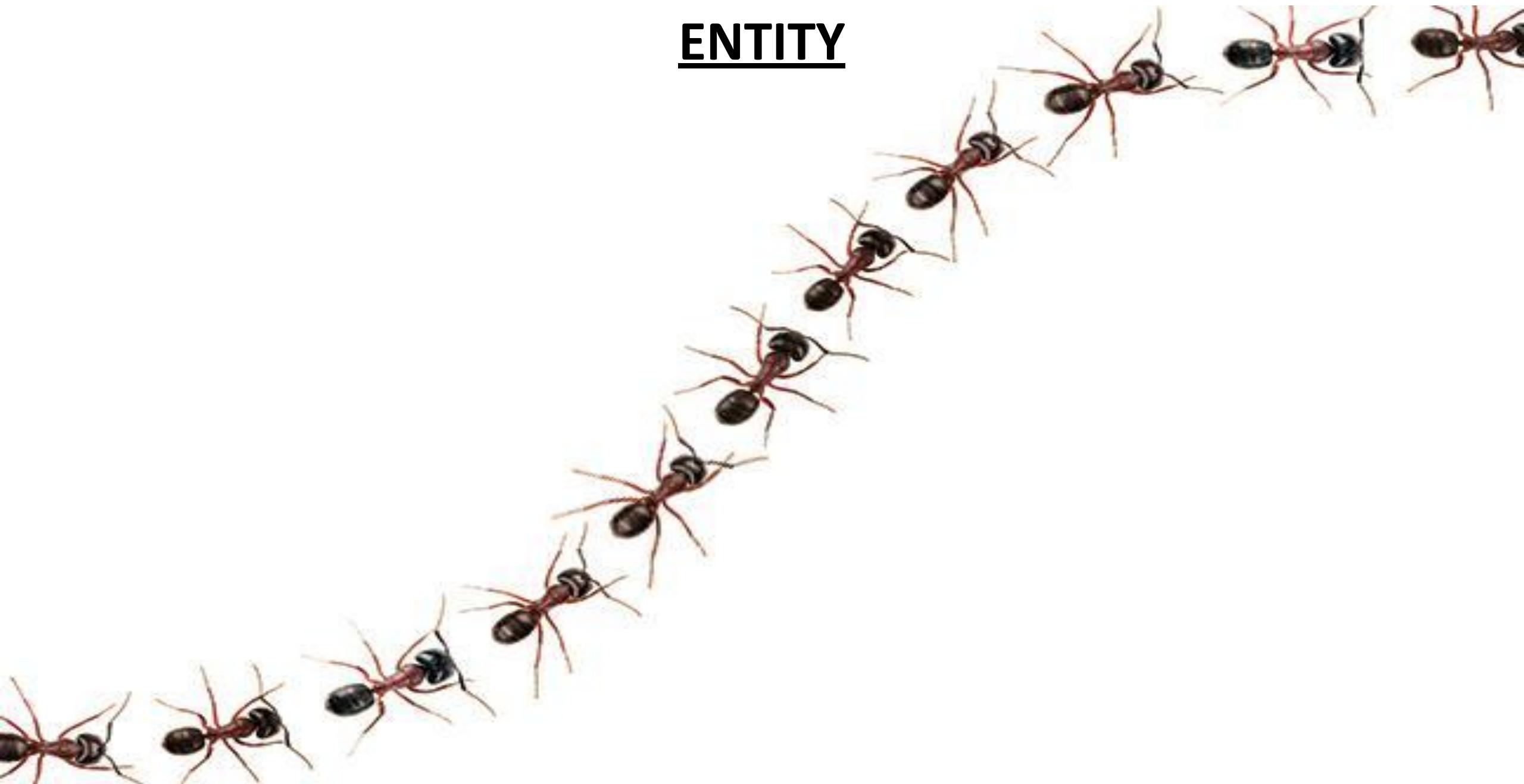




Conclusion

- It consists of collections of basic objects, called entities and of relationships among these entities and attributes which defines their properties.
- E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.
- It is free from ambiguities and provides a standard and logical way of visualizing the data.
- As basically it is a diagrammatical representation easy to understand even by a non-technical user.

ENTITY



ENTITY

- An entity is a thing or an object in the real world that is distinguishable from other objects 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.

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

- *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.*



- **ENTIY 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



- 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.

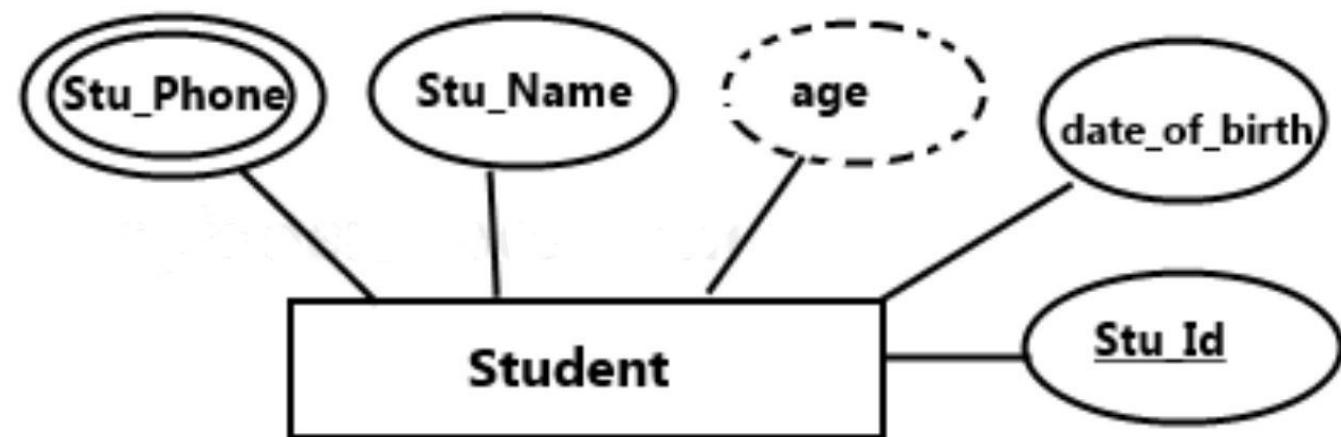
ATTRIBUTES

- 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

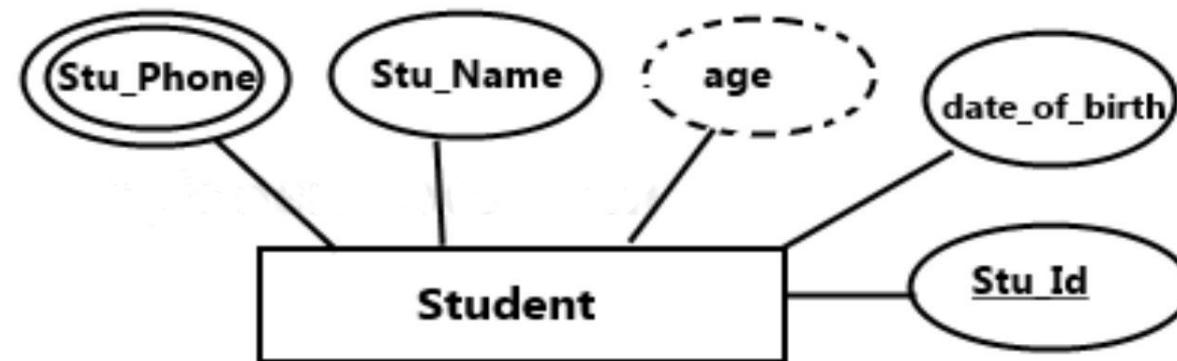
- In an ER diagram attributes are represented by ellipse or oval connected to rectangle.
- While in a relational model they are represented by independent column. e.g. Instructor (ID, name, salary, dept_name)

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



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			
Customer ID	First Name	Surname	Telephone Number
123	Rabri Devi	Singh	555-861-2025, 192-122-1111
456	Imarti Devi	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789	Barfi Devi	Doe	555-808-9633

Customer

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

સુગાડ્ ટેકનોલોજી

**Customer**

Customer ID	First Name	Surname	Telephone Number1	Telephone Number2
123	Pooja	Singh	555-861-2025	192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53	182-929-2929
789	John	Doe	555-808-9633	

Customer

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

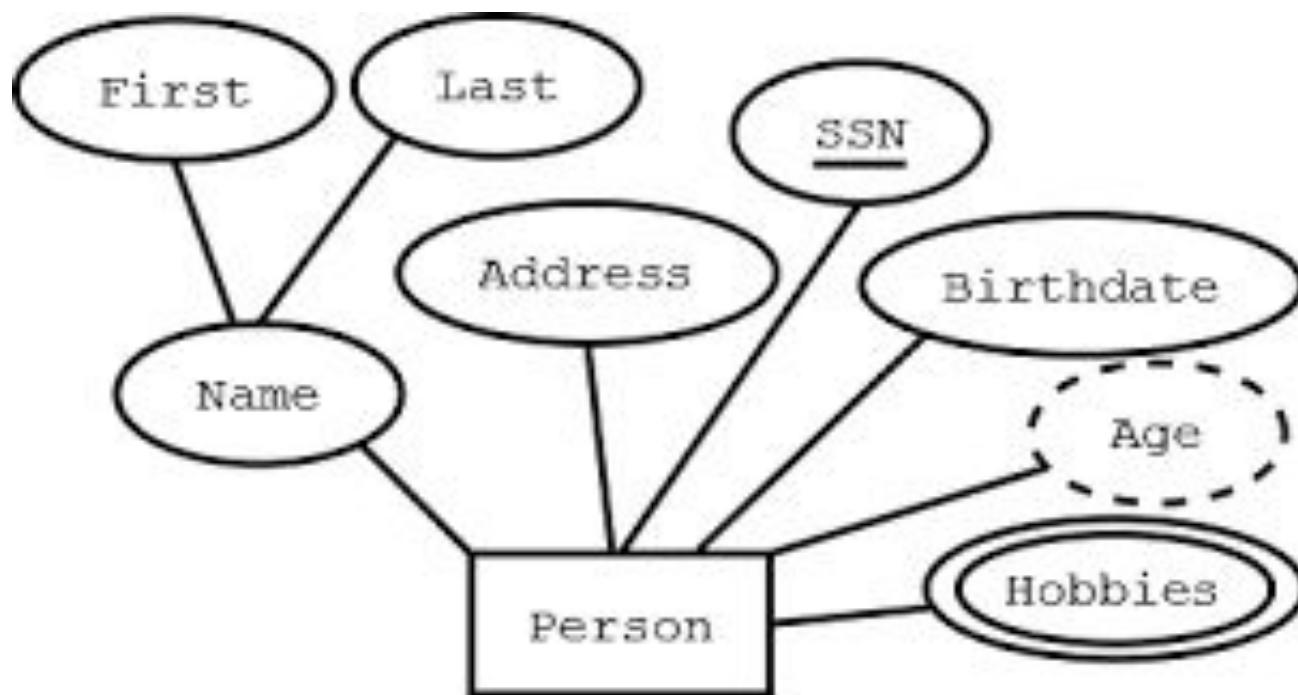


Customer Name		
<u>Customer ID</u>	First Name	Surname
123	Pooja	Singh
456	San	Zhang
789	John	Doe

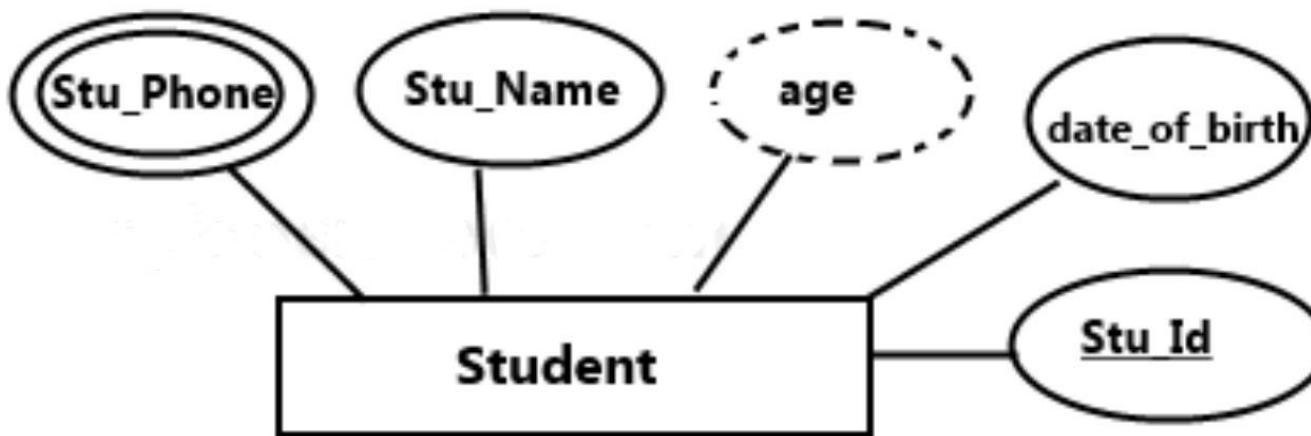
Customer Phone Number	
<u>Customer ID</u>	Telephone Number
123	555-861-2025
123	192-122-1111
456	(555) 403-1659 Ext. 53
456	182-929-2929
789	555-808-9633



- **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.

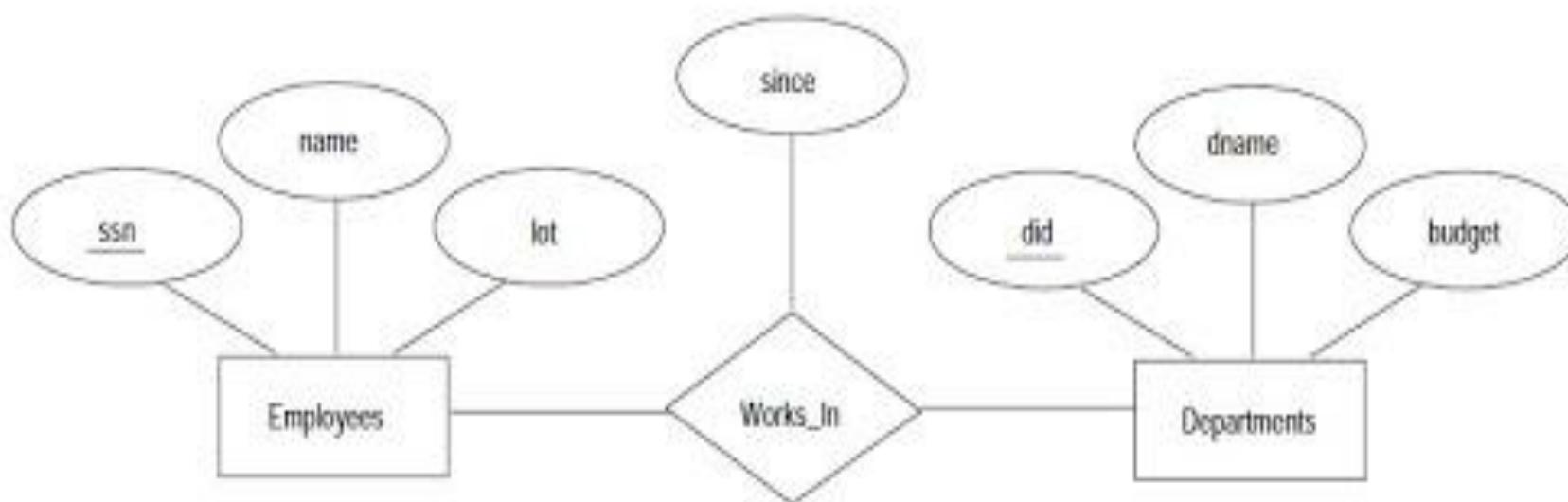


- **Stored** - Main attributes whose value is permanently stored in database. E.g. date_of_birth
- **Derived** -The value of these types of attributes can be derived from values of other Attributes. E.g. - Age attribute can be derived from date_of_birth and Date attribute.



Descriptive attribute - Attribute of relationship is called descriptive attribute.

- An attribute takes a null value when an entity does not have a value for it. The null value may indicate “not applicable”— that is, that the value does not exist for the entity.



- Null can also designate that an attribute value is **unknown**. An unknown value may be either missing (the value does exist, but we do not have that information) or not known (we do not know whether or not the value actually exists).

Relationship / Association

Relationship Status:

- Single
- In a relationship
- Engaged
- Married
- It's complicated
- In an open relationship
- Widowed
- Separated
- Divorced
- In a civil union
- In a domestic partnership

Family: Select Relation:

Featured Friends: 

Create new list - Add an existing list or group

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.

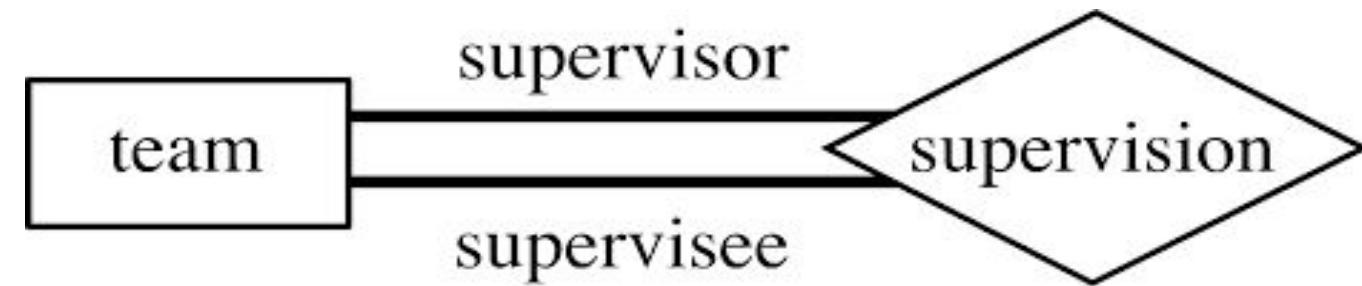
- Every relationship type has three components.
 - Name- Every relation must have a unique name.
 - Degree-
 - Structural constraints (cardinalities ratios, participation)

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.



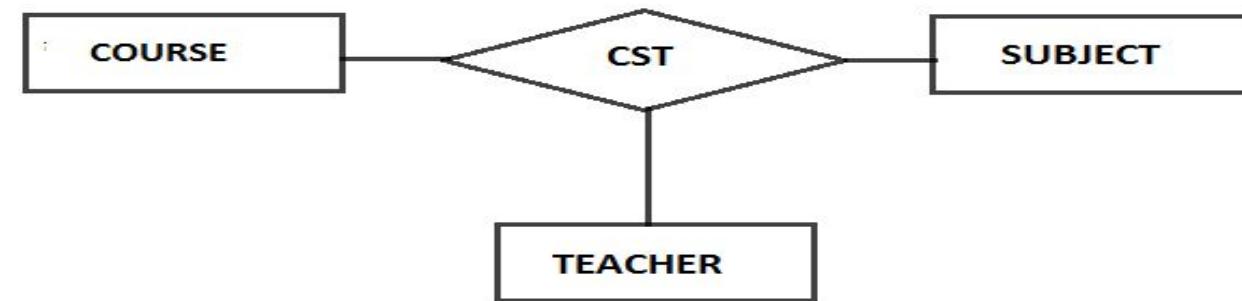
- **Unary Relationship** - One single entity set participate in a Relationship, means two entities of the same entity set are related to each other.
- These are also called as self-referential Relationship set.
- E.g.- A member in a team maybe supervisor of another member in team.



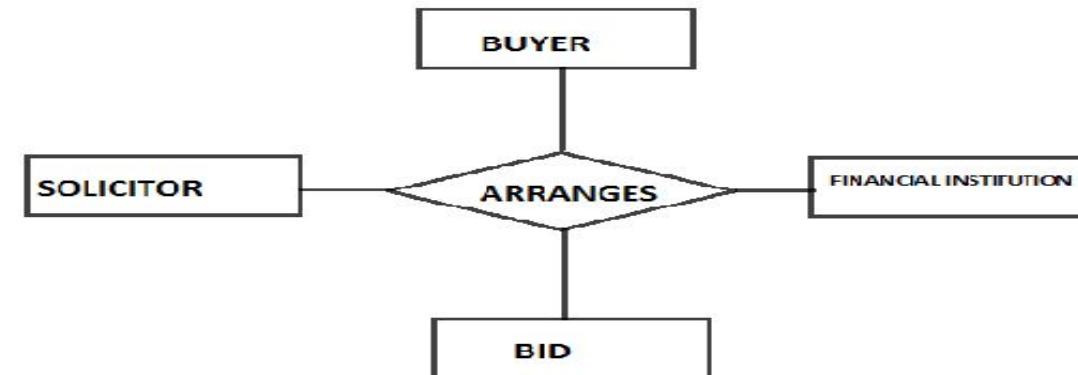
- **Binary Relationship** - Two entity sets participate in a Relationship. It is most common Relationship.



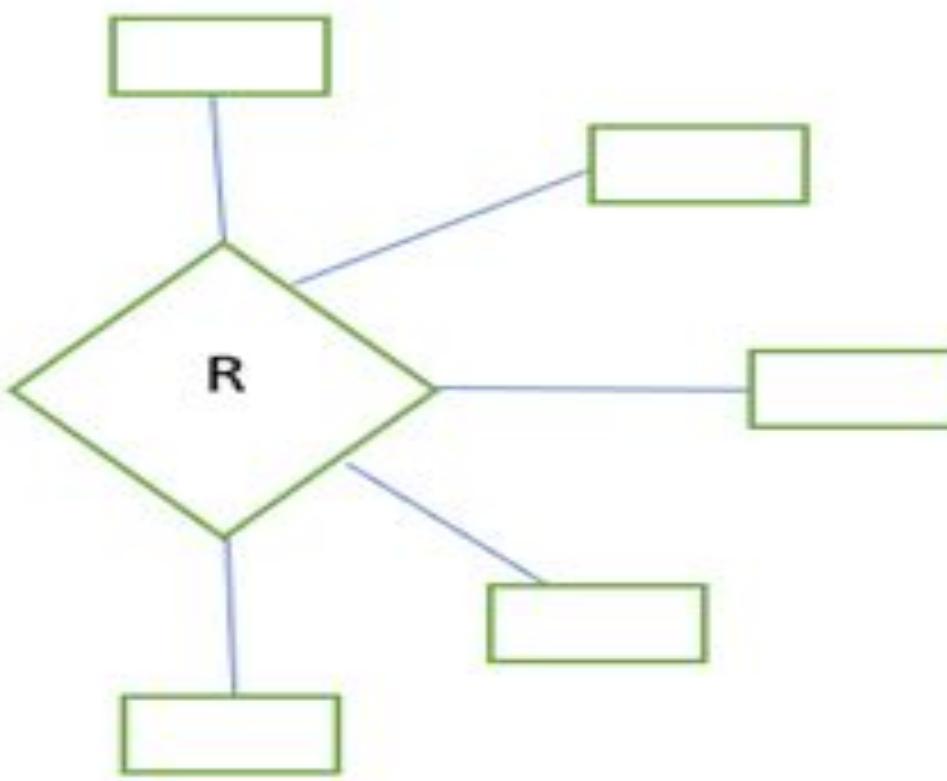
- **Ternary Relationship** - When three entities participate in a Relationship. E.g. The University might need to record which teachers taught which subjects in which courses.



- **Quaternary Relationship** - When four entities participate in a Relationship.



- **N-ary relationship** – where n number of entity set are associated

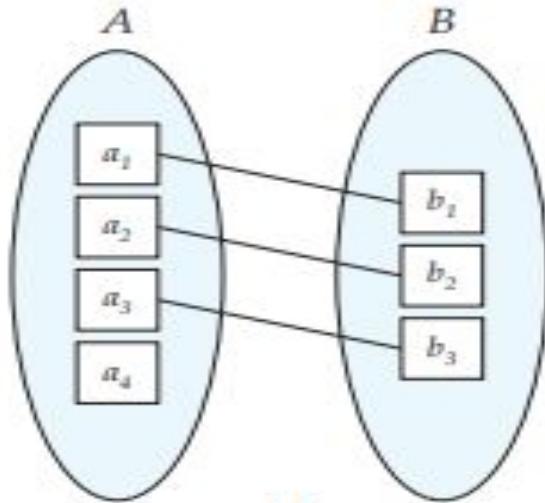


- But the most common relationships in ER models are **Binary**.

Structural constraints (Cardinalities Ratios, Participation)

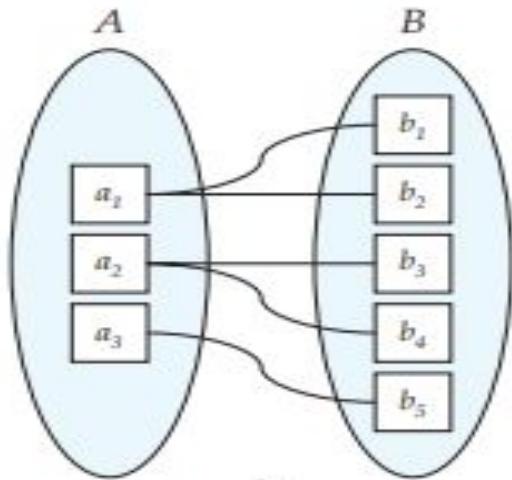
- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.
- Express the number of entities to which another entity can be associated via a relationship set. Four possible categories are-
 - One to One (1:1) Relationship.
 - One to Many (1: M) Relationship.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

One to One (1:1) Relationship - An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



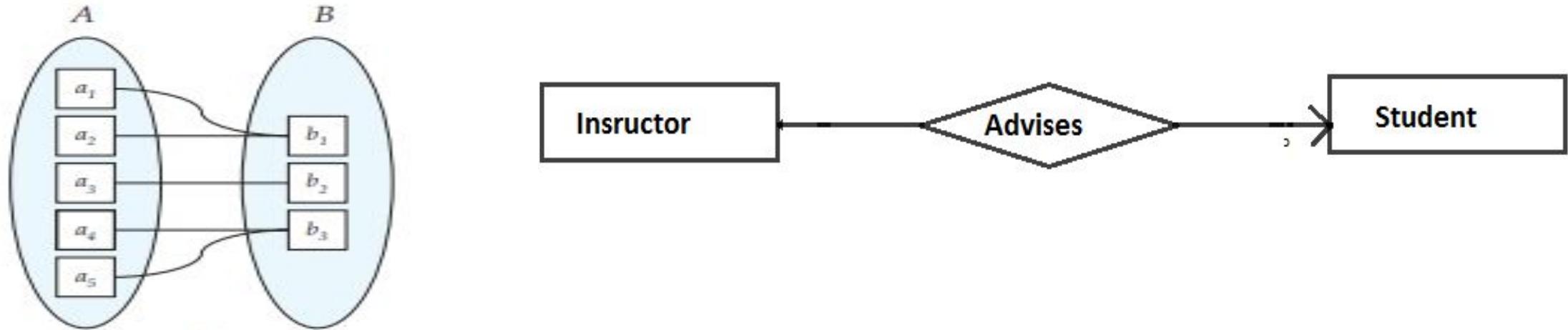
E.g.- The directed line from relationship set advisor to both entities set indicates that 'an instructor may advise at most one student, and a student may have at most one advisor'.

One to Many (1: M) Relationship - An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.



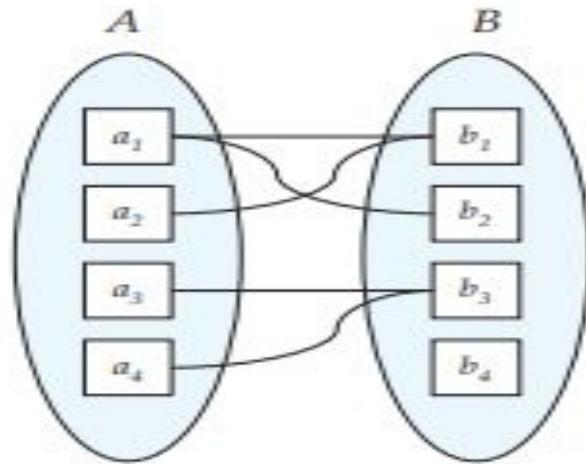
E.g.- This indicates that an instructor may advise many students, but a student may have at most one advisor.

Many to One (M: 1) Relationship - An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A.



E.g.- This indicates that student may have many instructors but an instructor can advise at most one student.

Many to Many(M:N) Relationship - An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



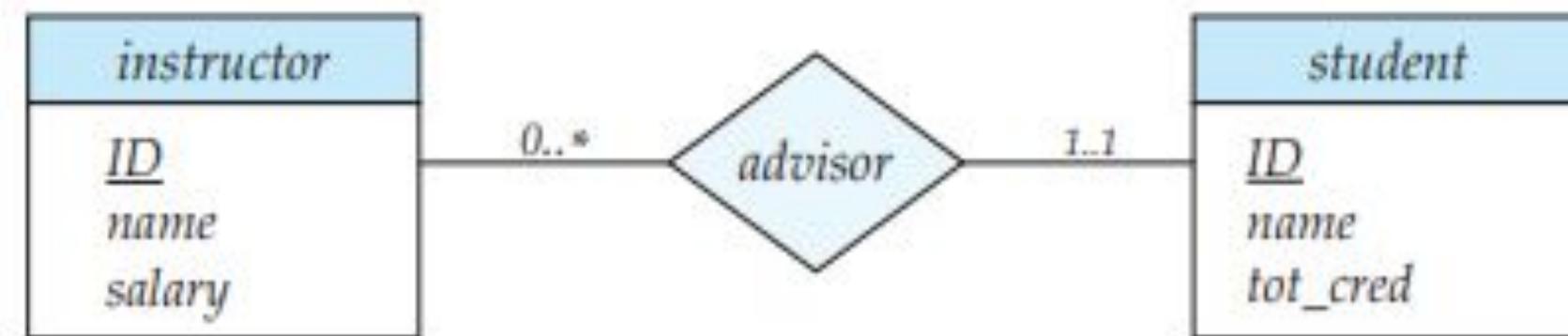
E.g.- This indicates a student may have many advisors and an instructor may advise many students.

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.



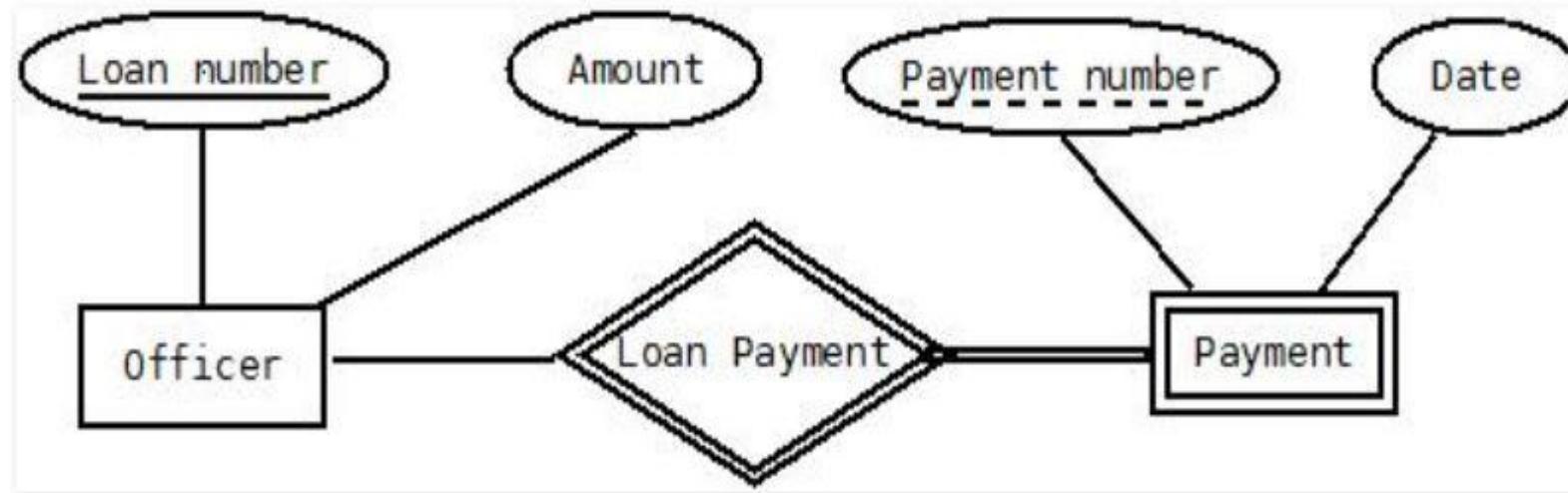
- **PARTICIPATION CONSTRAINTS**- it defines participations of entities of an entity type in a relationship.
- **PARTIAL PARTICIPATION (min cardinality zero)** - In Partial participation only some entities of entity set participate in Relationship set, that is there exists at least one entity which do 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.



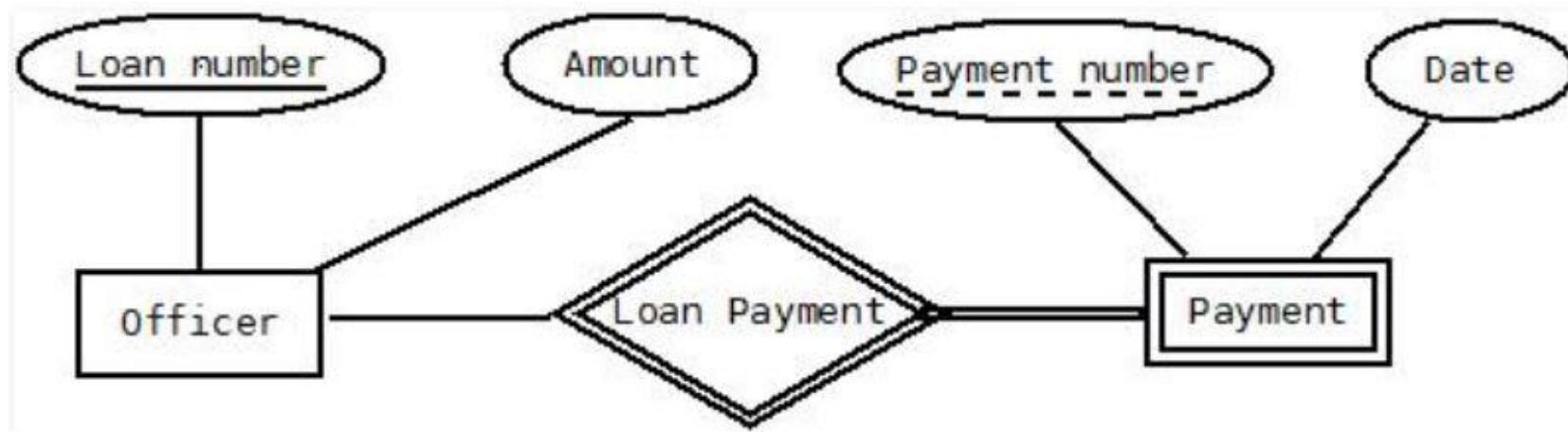
- **Conversion of 1-1 relationship(binary)**
 - No separate table is required, take pk of one side as pk on other side, priority must be given to the side having total participation.
- **Conversion of 1-n or n-1 relationship (binary)**
 - No separate table is required, modify n side by taking pk of 1 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.

STRONG AND WEAK ENTITY SET

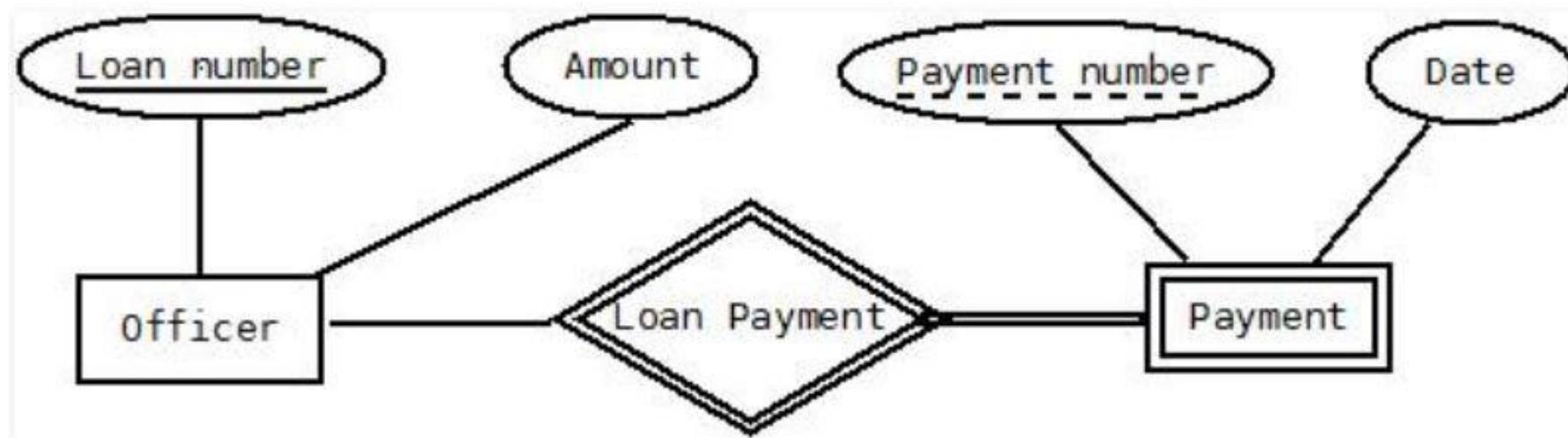
- An entity set is called strong entity set, if it has a primary key, all the tuples in the set are distinguishable by that key.
- An entity set that does not possess sufficient attributes to form a primary key is called a weak entity set. It contains discriminator attributes (partial key) which contain partial information about the entity set, but it is not sufficient enough to identify each tuple uniquely. Represented by double rectangle.



- For a weak entity set to be meaningful and converted into strong entity set, it must be associated with another strong entity set called the **identifying or owner entity set** i.e. weak entity set is said to be **existence dependent** on the identity set.
- The identifying entity set is said to own weak entity set that it identifies.
- A weak entity set may participate as owner in an identifying relationship with another weak entity set.



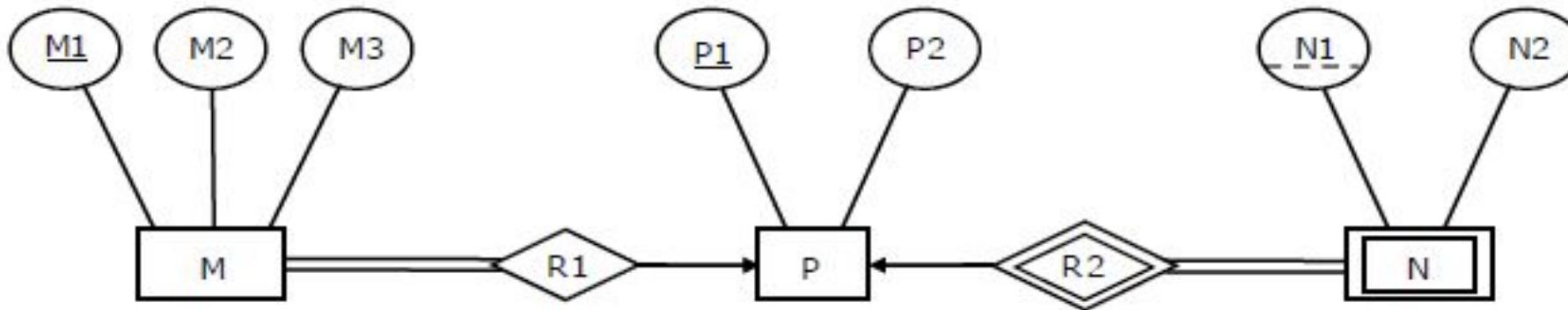
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship (double diamonds)**.
- The identifying relationship is many to one from the weak entity set to identifying entity set, and the participation of the weak entity set in relationship is always total.
- The primary key of weak entity set will be the union of primary key and discriminator attributes.



REASONS TO HAVE WEAK ENTITY SET

- Weak entities reflect the logical structure of an entity being dependent on another.
- Weak entity can be deleted automatically when their strong entity is deleted.
- Without weak entity set it will lead to duplication and consequent possible inconsistencies.

Consider the following ER diagram



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

(GATE-2008) (2 Marks)

(GATE-2008) (2 Marks)

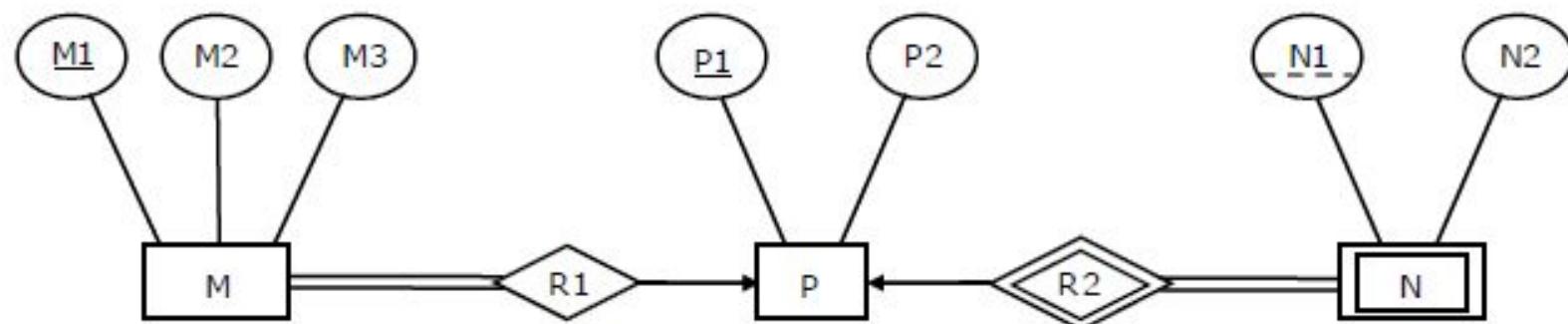
Consider the following ER diagram

a) $\{M_1, M_2, M_3, P_1\}$

b) $\{M_1, P_1, N_1, N_2\}$

c) $\{M_1, P_1, N_1\}$

d) $\{M_1, P_1\}$



Which of the following is a correct attribute set for one of the tables for the correct answer to the above question?

Q Map the following statements with true (T)/false (F)p-

S₁: Participation of the weak entity set in identifying relationship must be total.

S₂: Multi valued attributes in E- R diagram require separate tables when converted into relational model

a) F T

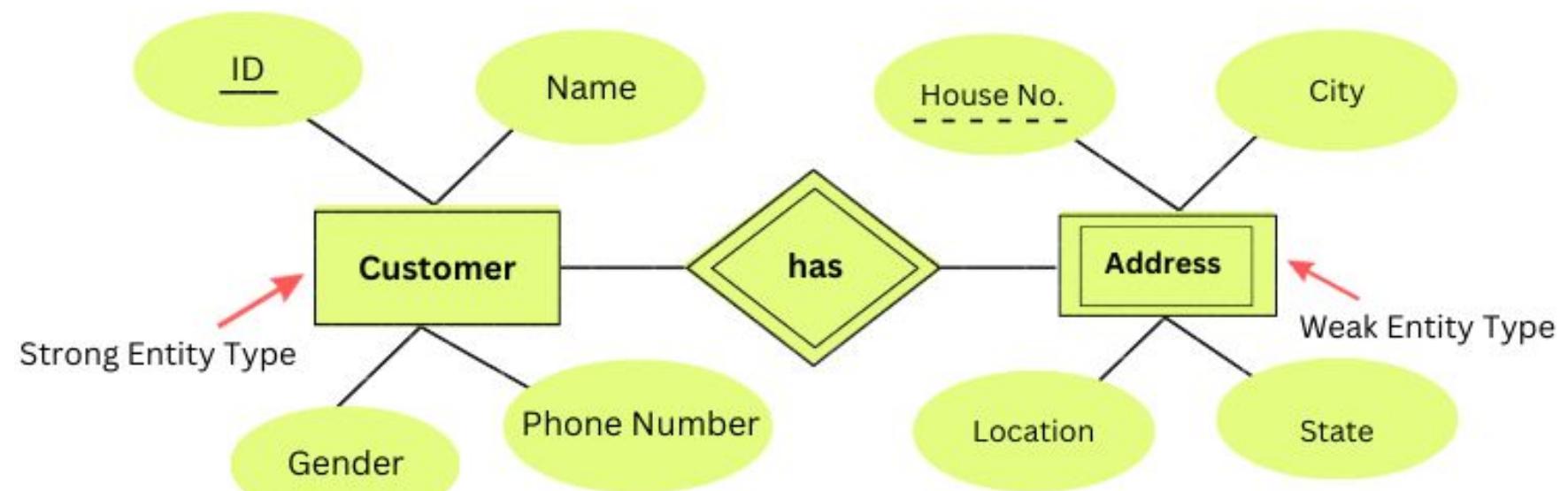
b) T F

c) F F

d) T T

Q. In the context of owner and weak entity sets in the ER (Entity-Relationship) data model, which one of the following statements is TRUE? (Gate 2024 CS) (1 Marks) (MCQ)

- (a) The weak entity set MUST have total participation in the identifying relationship
- (b) The owner entity set MUST have total participation in the identifying relationship
- (c) Both weak and owner entity sets MUST have total participation in the identifying relationship
- (d) Neither weak entity set nor owner entity set MUST have total participation in the identifying relationship



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 From ER Diagram To Relational Model

- Relationship(Binary)

- 1:n or n:1 No separate table is required, modify n side by taking pk of 1 side a foreign key on n side

Adhar_no	name	Age
1	A	19
2	B	18
3	C	20
4	D	20

Sim_id	Company	Number
1	A	OS
2	B	DBMS
3	C	TOC
4	D	CN

Conversion From ER Diagram Tom Relational Model

- Relationship(Binary)**

- (n-n) Separate table is required take pk of both table and declare their combination as a pk of new table

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

Edu_id	name	Subject
1	A	OS
2	B	DBMS
3	C	TOC
4	D	CN

Roll no	name	Age	Edu_id
1	A	19	1
1	A	19	3
1	A	19	4
2	B	18	2
2	B	18	3
3	C	20	2
4	D	20	3
4	D	20	4

Edu_id	name	Subject	Roll no
1	A	OS	1
2	B	DBMS	2
2	B	DBMS	3
3	C	TOC	1
3	C	TOC	2
3	C	TOC	4
4	D	CN	1
4	D	CN	4

Roll no	Edu_id
1	1
1	3
1	4
2	2
2	3
3	2
4	3
4	4

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

Edu_id	name	Subject
1	A	OS
2	B	DBMS
3	C	TOC
4	D	CN

Roll no	Edu_id
1	1
1	3
1	4
2	2
2	3
3	2
4	3
4	4

Conversion From ER Diagram Tom Relational Model

- **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 What should be the condition for total participation of the entity in a relation?

- a)** Maximum cardinality should be one
- b)** Minimum cardinality should be zero
- c)** Minimum cardinality should be one
- d)** None of these

Q In an Entity-Relationship (ER) model, suppose R is a many-to-one relationship from entity set E_1 to entity set E_2 . Assume that E_1 and E_2 participate totally in R and that the cardinality of E_1 is greater than the cardinality of E_2 . **(GATE-2018) (1 Marks)**

Which one of the following is true about R?

- (a)** Every entity in E_1 is associated with exactly one entity in E_2
- (b)** Some entity in E_1 is associated with more than one entity in E_2
- (c)** Every entity in E_2 is associated with exactly one entity in E_1
- (d)** Every entity in E_2 is associated with at most one entity in E_1

Q An ER model of a database consists of entity types A and B. These are connected by a relationship R which does not have its own attribute. Under which one of the following conditions, can the relational table for R be merged with that of A? **(GATE-2017) (1 Marks)**

- (a)** Relationship R is one-to-many and the participation of A in R is total
- (b)** Relationship R is one-to-many and the participation of A in R is partial
- (c)** Relationship R is many-to-one and the participation of A in R is total
- (d)** Relationship R is many-to-one and the participation of A in R is partial

Q Consider an Entity-Relationship (ER) model in which entity sets E_1 and E_2 are connected by an m: n relationship R_{12} , E_1 and E_3 are connected by a 1: n (1 on the side of E_1 and n on the side of E_3) relationship R_{13} . E_1 has two single-valued attributes a_{11} and a_{12} of which a_{11} is the key attribute. E_2 has two single-valued attributes a_{21} and a_{22} of which a_{21} is the key attribute. E_3 has two single valued attributes a_{31} and a_{32} of which a_{31} is the key attribute. The relationships do not have any attributes. If a relational model is derived from the above ER model, then the minimum number of relations that would be generated if all the relations are in 3NF is _____. **(GATE-2015) (2 Marks)**

a) 2

b) 3

c) 4

d) 5

**Q Given the basic ER and relational models, which of the following is INCORRECT? (GATE-2012)
(1 Marks)**

- (A) An attribute of an entity can have more than one value**
- (B) An attribute of an entity can be composite**
- (C) In a row of a relational table, an attribute can have more than one value**
- (D) In a row of a relational table, an attribute can have exactly one value or a NULL value**

Q Let E_1 and E_2 be two entities in an E/R diagram with simple single-valued attributes. R_1 and R_2 are two relationships between E_1 and E_2 , where R_1 is one-to-many and R_2 is many-to-many. R_1 and R_2 do not have any attributes of their own. What is the minimum number of tables required to represent this situation in the relational model? **(GATE-2005) (2 Marks) (NET-DEC-2015)**

a) 2

b) 3

c) 4

d) 5

Q Consider the entities ‘hotel room’, and ‘person’ with a many to many relationship ‘lodging’ as shown below:

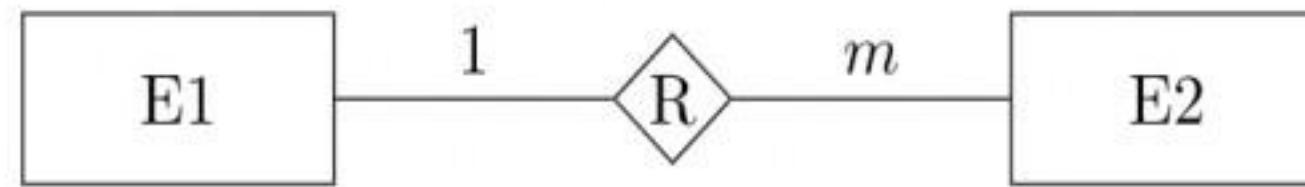


If we wish to store information about the rent payment to be made by person (s) occupying different hotel rooms, then this information should appear as an attribute of **(GATE-2005)(1 Marks)**

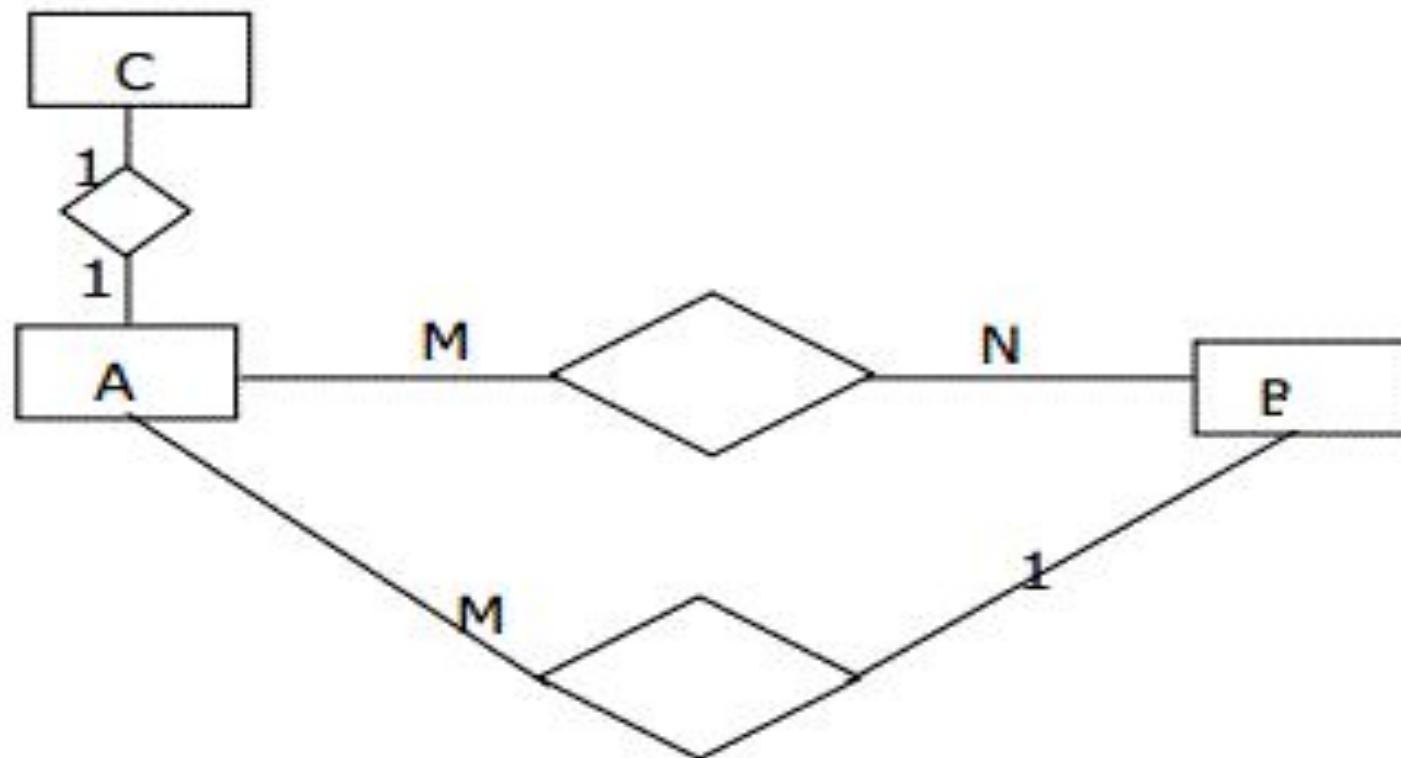
- (A) Person
- (B) Hotel Room
- (C) Lodging
- (D) None of these

Q Consider the following entity relationship diagram (ERD), where two entities E_1 and E_2 have a relation R of cardinality $1 : m$. The attributes of E_1 are A_{11}, A_{12} and A_{13} where A_{11} is the key attribute. The attributes of E_2 are A_{21}, A_{22} and A_{23} where A_{21} is the key attribute and A_{23} is a multi-valued attribute. Relation R does not have any attribute. A relational database containing minimum number of tables with each table satisfying the requirements of the third normal form (3NF) is designed from the above ERD. The number of tables in the database is **(GATE-2004)(2 Marks)**

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

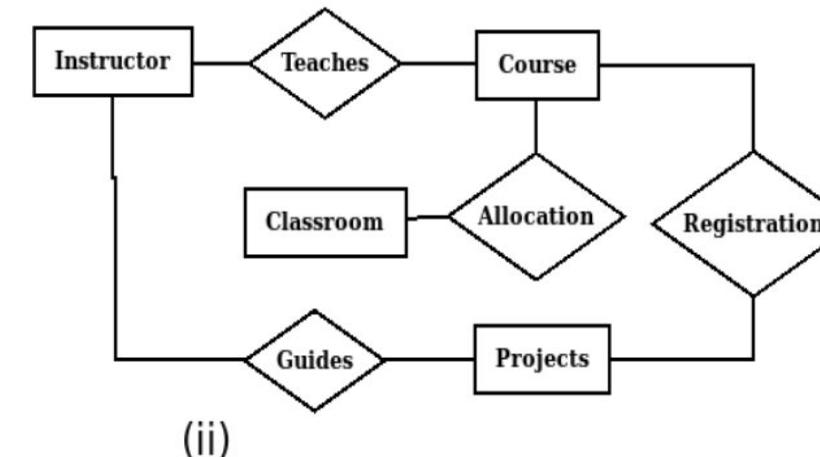
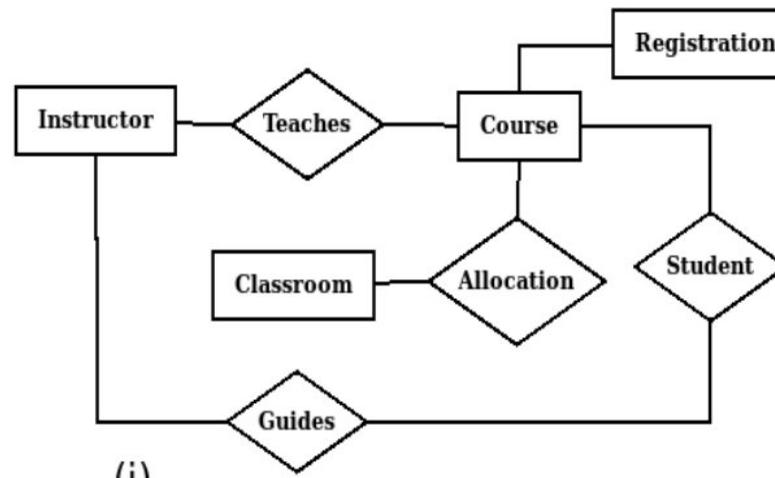


Q The minimum number of tables required to convert the following ER diagram to relation model is _____



Q. Let S be the specification: “Instructors teach courses. Students register for courses. Courses are allocated classrooms. Instructors guide students.” Which one of the following ER diagrams CORRECTLY represents? (Gate 2024,CS)(1 Marks)(MCQ)

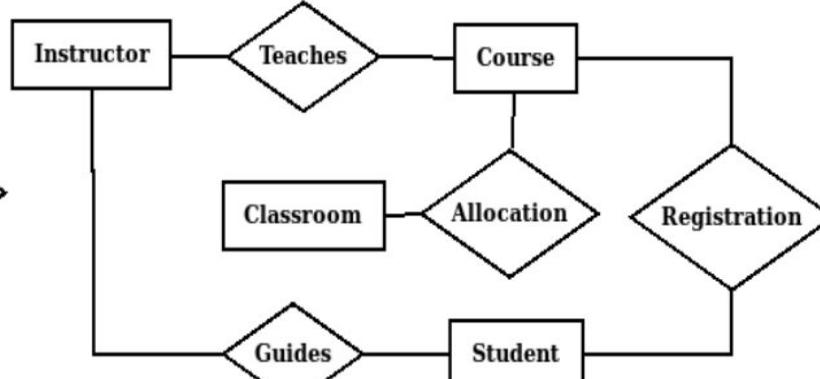
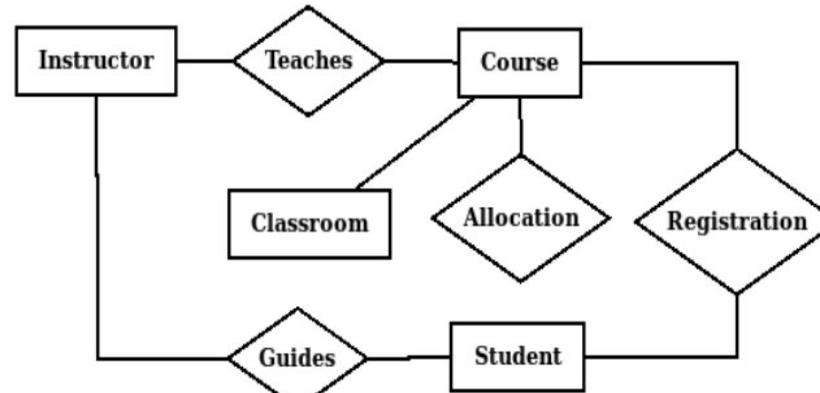
(a) (i)



(b) (ii)

(c) (iii)

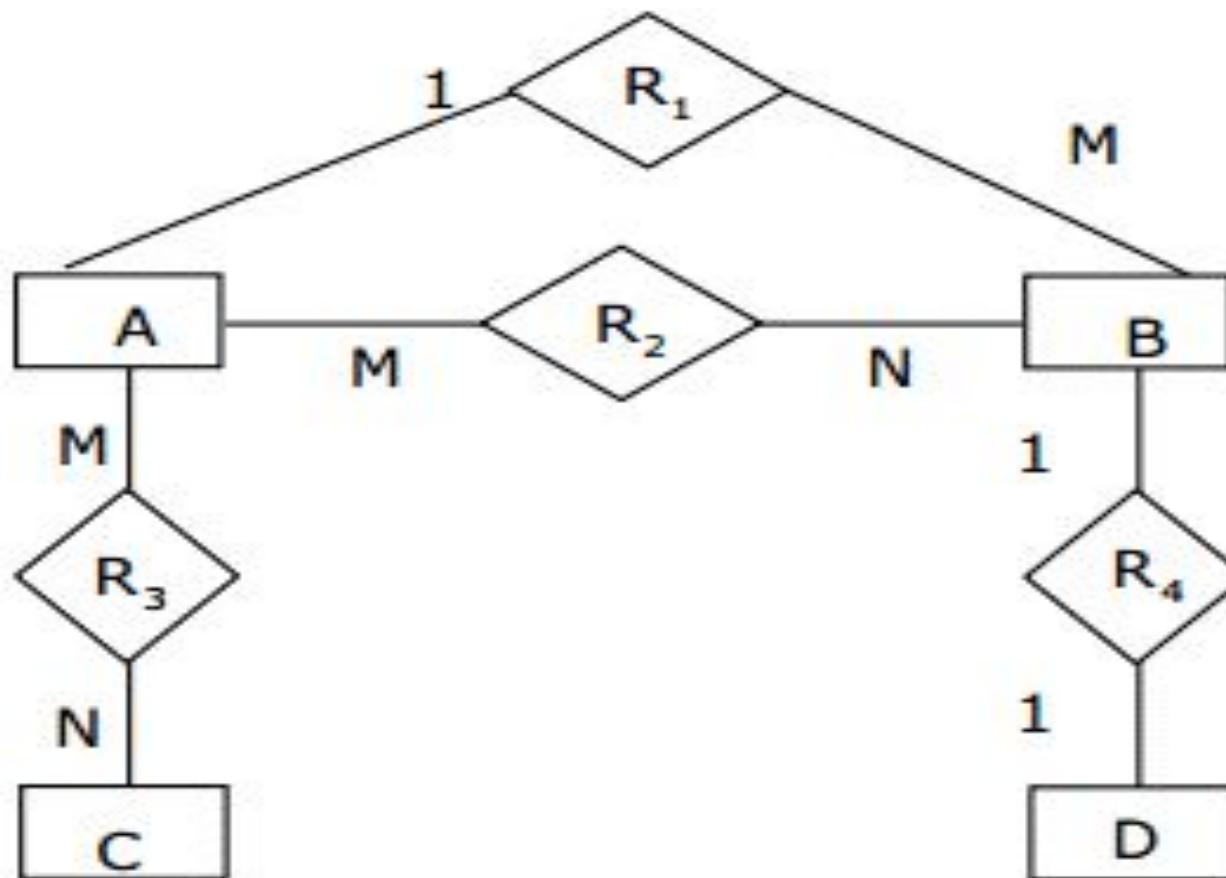
(d) (iv)



(iii)

(iv)

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



Trapes

- It is possible that even after all efforts there remain some problem with ER diagram. These problems are called trapes. There are two types of trapes in ER diagram.
- **FAN TRAP** - If two or more 1 to M relationships are emerging out from single entity. Then there will be a FAN trap.

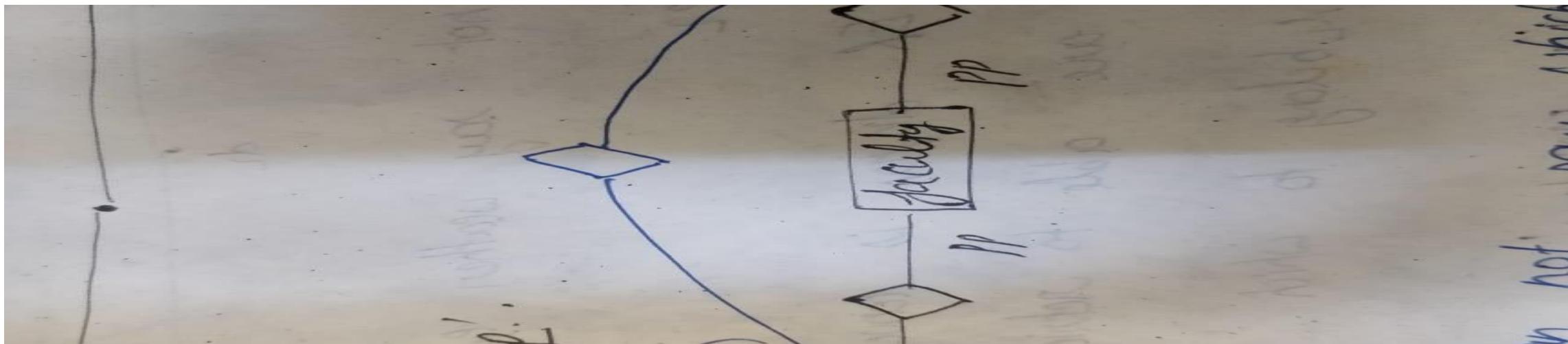


- A single site contains many departments and employs many staff. However, which staff work in a particular department
- The fan trap is resolved by restructuring the original ER model to represent the correct association.



- **CHASM TRAP** - If two directly related entities are connected through another(third) entity with partial participation then there is a chasm trap. So, logic suggests the existence of a relationship between entity sets, but the relationship does not exist between certain entity occurrences in ER diagram.
- A model suggests the existence of a relationship between entity types, but the pathway does not exist between certain entity occurrences. Following is the example in different notations.

- How to eliminate - Create direct relationship between these 2 entities.



- **ADVANTAGES OF E-R DIGRAM**

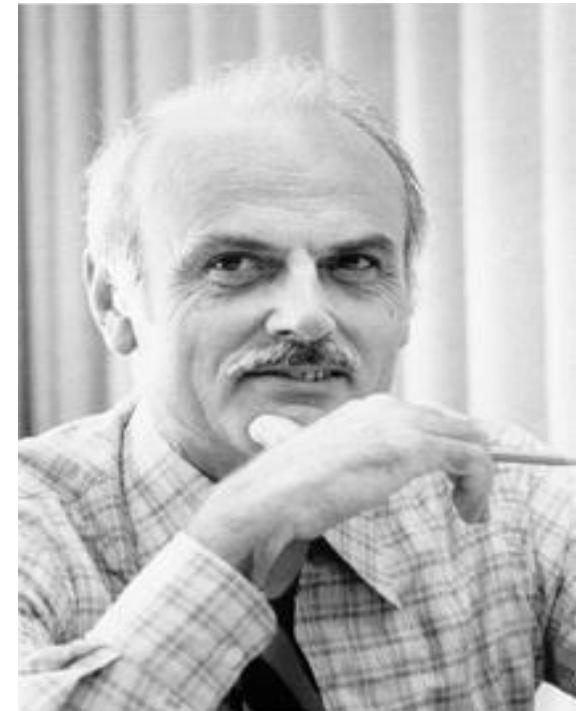
- Constructs used in the ER diagram can easily be transformed into relational tables.
- It is simple and easy to understand with minimum training.

- **DISADVANTAGE OF E-R DIGRAM**

- Loss of information content.
- Limited constraints representation.
- It is overly complex for small projects.

RELATIONAL DATABASE MANAGEMENT SYSTEM

- A Relational Database Management System (RDBMS) is a software system that uses a relational model to create, manage, and query data in databases through tables linked by defined relationships. Most modern commercial and open-source database applications are relational in nature.
- Based on the relational model specified by Edgar F. Codd. The father of modern relational database design in 1970.



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.

**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

- **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.

Rows/Tuples/Record/ Cardinality

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

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**
 - **Modification Anomalies**
 - **Deletion Anomalies**

- **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

- Without normalization data base system may be inaccurate, slow and inefficient and they might not produce the data we expect. Normalization may be simply defined as refinement process.
- Which includes creating tables and establishing relationships between those tables according to rules designed both to protect data and make the database more flexible by eliminating two factors;
 - Redundancy
 - Inconsistent Dependency

Conclusion

- Like every paragraph must have a single idea similarly every table must have a single idea and if a table contains more than one idea then that table must be decomposed until each table contains only one idea.
- We need some tools to approach this decomposition or normalization on large database which contains a number of table, and that tool is functional dependencies.



Br_code



Br_hod_name

Br_code



Br_hod_name

Functional Dependency कोई बताता नहीं, इसकी feel आ जाती है



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

- 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 Consider the relation X(P, Q, R, S, T, U) with the following set of functional dependencies

$F = \{$

$\{P, R\} \rightarrow \{S, T\},$

$\{P, S, U\} \rightarrow \{Q, R\}$

$\}$

Which of the following is the trivial functional dependency in F^+ is closure of F? **(GATE- 2016) (1 Marks)**

(a) $\{P, R\} \rightarrow \{S, T\}$

(b) $\{P, R\} \rightarrow \{R, T\}$

(c) $\{P, S\} \rightarrow \{S\}$

(d) $\{P, S, U\} \rightarrow \{Q\}$

1. α - Determinant (Determines β value).
2. β - Dependent (Dependent on α).
3. If $k \sqsubseteq R$, then K is a super key of R
4. Note: A functional dependency is a property of the relation schema R , not of a particular legal relation state/instance r of R .

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

Q Which of the following functional dependencies are satisfied by the instance? (GATE-2000) (2 Marks)

(A) $XY \rightarrow Z$ and $Z \rightarrow Y$

(B) $YZ \rightarrow X$ and $Y \rightarrow Z$

(C) $YZ \rightarrow X$ and $X \rightarrow Z$

(D) $XZ \rightarrow Y$ and $Y \rightarrow X$

X	Y	Z
1	4	2
1	5	3
1	6	3
3	2	2

- Shortcut Steps to find whether a FD from $\alpha \square \beta$ can be concluded on a given instance or not
 1. Find Weather all values of α are different or not, if yes then FD valid
 2. Find Weather all values of β are same or not, if yes then FD valid
 3. जब कुछ भी काम ना करे, Try to find two same values of α on which we get different values of β

Q Consider the following relation instance, Which of the following dependencies are satisfied by the above relation instance?

a) $A \square B, BC \square A$

b) $C \square B, CA \square B$

c) $B \square C, AB \square C$

d) $A \square C, BC \square A$

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

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

A) $A \sqsubset b$

B) $BC \sqsubset A$

C) $B \sqsubset C$

D) $AC \sqsubset B$

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

Q From the following instance of a relation scheme R (A, B, C), we can conclude that **(CS-2002) (2 Marks)**

- (A)** A functionally determines B and B functionally determines C
- (B)** A functionally determines B and B does not functionally determine C
- (C)** B does not functionally determine C
- (D)** A does not functionally determine B and B does not functionally determine C

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

Q From the following instance of the relation schema R (A, B, C) we can conclude that

- a)** A functionally determines B, B functionally determines C
- b)** B functionally determines C, C functionally determines A
- c)** A functionally determines B, but B doesn't functionally determine C
- d)** None of these

A	B	C
a1	b1	c1
a2	b1	c1
a3	b3	c3
a3	b3	c4
a5	b5	c5

Q Which of the following dependencies are satisfied by the relation instance?

A \square B

B \square C

B \square A

C \square B

C \square A

A \square C

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

Q Which of the following dependencies are satisfied by the relation instance?

$XZ \square X$

$XY \square Z$

$Z \square Y$

$Y \square Z$

$XZ \square Y$

X	Y	Z
1	4	3
1	5	3
4	6	3
3	2	2

Q. A functional dependency $F: X \rightarrow Y$ is termed as a useful functional dependency if and only if it satisfies all the following three conditions: **(Gate 2024 CS) (2 Marks) (NAT)**

- X is not the empty set.
- Y is not the empty set.
- Intersection of X and Y is the empty set.

For a relation R with 4 attributes, the total number of possible useful functional dependencies is _____

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

- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from F.
 - DENOTED BY F^+
- Set of all attributes Functionally determined by X either directly from FD'S or logically derived.

DIRECT METHOD

E.g. In a Relation R (A, B, C, D), with set of functional dependencies as-

{

$A \rightarrow B$

$B \rightarrow C$

$AB \rightarrow D$

}

$A^+ =$

Q R(ABCDEFG)

A □ B

BC □ DE

AEG □ G

(AC)⁺ =?

Q R(ABCDE)

A □ BC

CD □ E

B □ D

E □ A

(B)⁺ =

Q R(ABCDEF)

AB □ C

BC □ AD

D □ E

CF □ B

(AB)⁺ =

Q R(ABCDEFGH)

A ⊓ BC

CD ⊓ E

E ⊓ C

D ⊓ AEH

ABH ⊓ BD

DH ⊓ BC

(BCD)⁺ =

Q Consider the following functional dependencies over the relation R (ABCDEF)

A \square B

C \square DE

AC \square F

What is the closure of (AC)?

a) ACF

b) ACFDE

c) ACEFDB

d) ACFD

Q In a Relation **R** (A, B, C, D) Given

$$F = \{A \sqsubset B, B \sqsubset C, C \sqsubset D\}$$

To check whether, A \sqsubset C is valid or not ?

Q Suppose the following functional dependencies hold on a relation U with attributes P,Q,R,S, and T:

$$P \rightarrow QR$$

$$RS \rightarrow T$$

Which of the following functional dependencies can be inferred from the above functional dependencies? **(GATE 2021) (2 MARKS)**

(A) $PS \rightarrow T$

(B) $R \rightarrow T$

(C) $P \rightarrow R$

(D) $PS \rightarrow Q$

ARMSTRONG'S AXIOMS

1. An **axiom or postulate** is a statement that is taken to be true, to serve as a premise or starting point for further reasoning and arguments.
2. **Armstrong's axioms** are a set of axioms (or, more precisely, inference rules) used to infer all the functional dependencies on a relational database.
3. They were developed by William W. Armstrong in his 1974 paper.
4. The axioms are sound in generating only functional dependencies in the closure of a set of functional dependencies (denoted as F^+) when applied to that set (denoted as F).



Armstrong Axioms

- **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 $WX \rightarrow Z$
- **Composition:** If $X \rightarrow Y$ and $Z \rightarrow W$, then $XZ \rightarrow YW$

Q.The Symbol → indicates functional dependency in the context of a relational database.
Which of the following options is/are TRUE? (Gate 2024,CS)(2 Marks) (MSQ)

- (a) $(X,Y) \rightarrow (Z,W)$ implies $X \rightarrow (Z,W)$
- (b) $(X,Y) \rightarrow (Z,W)$ implies $(X,Y) \rightarrow Z$
- (c) $((X,Y) \rightarrow Z \text{ and } W \rightarrow Y)$ implies $(X,W) \rightarrow Z$
- (d) $(X \rightarrow Y \text{ and } Y \rightarrow Z)$ implies $X \rightarrow Z$

Why Armstrong axioms refers to the Sound and Complete

- By sound, we mean that given a set of functional dependencies F specified on a relation schema R , any dependency that we can infer from F by using the primary rules of Armstrong axioms holds in every relation state r of R that satisfies the dependencies in F .
- By complete, we mean that using primary rules of Armstrong axioms repeatedly to infer dependencies until no more dependencies can be inferred results in the complete set of all possible dependencies that can be inferred from F .

APPLICATION OF ATTRIBUTE CLOSURE

Equivalence of Two FD sets-

Two FD sets F_1 and F_2 are equivalent if –

$$F_1^+ = F_2^+$$

Or

$$F_1 \sqsubseteq F_2^+ \text{ and } F_2 \sqsubseteq F_1^+$$

Race एक जगह से शुरू होने जरूरी है

Q Consider the following set of fd R(ACDEH)

F	G
A \square C	A \square CD
AC \square D	E \square AH
E \square AD	
E \square H	

Q R(VWXYZ)

F	G
V □ W	V □ W
VW □ X	V □ X
Y □ VX	Y □ V
Y □ Z	Y □ Z

Q Consider the following set of fd R(ABCDE)

F	G
B \square CD	B \square CDE
AD \square E	A \square BC
B \square A	AD \square E

Q consider the following relation R(PQRS)

F	G
$P \square Q$	$P \square QR$
$Q \square R$	
$R \square S$	$R \square S$

Q consider the following relation R(ABCD)

F	G
A \square B	A \square BC
B \square C	B \square A
C \square A	C \square A

Q consider the following relation R(VWXYZ)

F	G
$W \square X$	$W \square XY$
$WX \square Y$	$Z \square WX$
$Z \square WY$	
$Z \square V$	

To find the MINIMAL COVER /CANONICAL COVER/IRREDUCIBLE SET

- Minimal cover- It means to eliminate any kind of redundancy from a FD set.
- A canonical cover of a set of functional dependencies, F is a simplified set of functional dependencies that has the same closure as the original set F.
- There may be any following type of redundancy in the set of functional dependencies:-
 - Complete production may be Redundant.
 - One or more than one attributes may be redundant on right hand side of a production.
 - One or more than one attributes may be redundant on Left hand side of a production.

Gate aspirant के हाथों से minimize होने के बाद



Q R(ABCD)

A -> B

C -> B

D -> ABC

AC -> D

Procedure to find MINIMAL COVER

- Use decomposition rule wherever applicable so that RHS of a production/FD contains only single attribute.
- Remove extraneous attribute on LHS of a production by finding the closure for every possible subset, if in any case the closure is same it means remaining attributes are redundant.
- For every production find the closure value of LHS of production keeping the production in a set, and next time ignoring the production to be in a set. If both closure set matches, it means the production is redundant.

माफ़ करो या साफ़ करो

Q The following functional dependencies hold true for the relational schema $\{V, W, X, Y, Z\}$: GATE (2017 SET 1)

$$V \rightarrow W$$

$$VW \rightarrow X$$

$$Y \rightarrow VX$$

$$Y \rightarrow Z$$

Which of the following is irreducible equivalent for this set of functional dependencies?

(a)	(b)	(c)	(d)
$V \rightarrow W$	$V \rightarrow W$	$V \rightarrow W$	$V \rightarrow W$
$V \rightarrow X$	$W \rightarrow X$	$V \rightarrow X$	$W \rightarrow X$
$Y \rightarrow V$	$Y \rightarrow V$	$Y \rightarrow V$	$Y \rightarrow V$
$Y \rightarrow Z$	$Y \rightarrow Z$	$Y \rightarrow X$	$Y \rightarrow X$
		$Y \rightarrow Z$	$Y \rightarrow Z$

Q Find the minimal cover for the FD set

$$\{AC \rightarrow BD, A \rightarrow C, B \rightarrow C, D \rightarrow C\}$$

a) $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, A \rightarrow D\}$

b) $\{A \rightarrow B, A \rightarrow D, A \rightarrow C, B \rightarrow D\}$

c) $\{A \rightarrow B, A \rightarrow D, A \rightarrow C, B \rightarrow C\}$

d) $\{A \rightarrow B, A \rightarrow D, D \rightarrow C, B \rightarrow C\}$

Q R(WXYZ)

X -> W

WZ -> XY

Y -> WXZ

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-



Super key

- Set of attributes using which we can identify each tuple uniquely is called Super key, i.e. the set of attributes used to differentiate each tuple of a relation.
- Let X be a set of attributes in a Relation R , if X^+ (Closure of X) determines all attributes of R then X is said to be Super key of R .
- There should be at least one Super key with Not Null constraint.

- A relation of ‘n’ attributes with every attribute being a super key, then there are 2^{n-1}
- Biggest Super Key possible in a Relation is a Set comprising all attributes of a Relation

Q The maximum number of super keys for the relation schema R(E, F, G, H) with E as the key is **(Gate-2014) (1 Marks)**

a) 5

E F G H

b) 6

c) 7

d) 8

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

$$AB \rightarrow C; \quad BC \rightarrow D; \quad C \rightarrow E;$$

The number of super keys in the relation R is _____. **(GATE 2022) (1 MARKS)**

A B C D E

Candidate key

1. Minimum set of attributes that differentiates the tuple of a Relation. No proper subset as super key Also called as **MINIMAL SUPER KEY**.
2. There should be at least one candidate key with Not Null constraint.
3. Prime attribute - Attributes that are member of candidate Keys are called Prime attributes.

Primary key

1. One of the candidate keys is selected by database administrator as a Primary Key.
2. Primary Key attribute are not allowed to have Null values.
3. At most one Primary Key per table is allowed in RDMS.
4. Candidate key which are not chosen as primary key is alternate key.

Q Which of the following is NOT a superkey in a relational schema with attributes V, W, X, Y, Z and primary key VY? **(GATE – 2016) (1 Marks)**

(a) VXYZ

(b) VWXZ

(c) VWXY

(d) VWXYZ

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	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

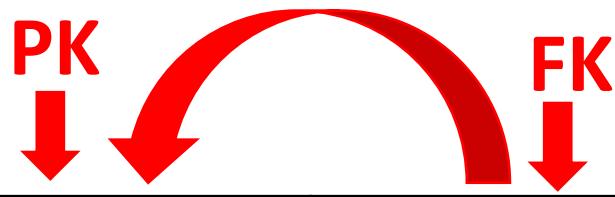
PK
↓

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



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

Q Consider the following tables T_1 and T_2 . In table T_1 , **P** is the primary key and **Q** is the foreign key referencing **R** in table T_2 with on-delete cascade and on-update cascade. In table T_2 , **R** is the primary key and **S** is the foreign key referencing **P** in table T_1 with on-delete set NULL and on-update cascade. In order to delete record $\langle 3,8 \rangle$ from table T_1 , the number of additional records that need to be deleted from table T_1 is _____. (GATE- 2017) (1 Marks)

T1		T2	
P	Q	R	S
2	2	2	2
3	8	8	3
7	3	3	2
5	8	9	7
6	9	5	7
8	5	7	2
9	8		

Q The following table has two attributes A and C where A is the primary key and C is the foreign key referencing A with on-delete cascade.

The set of all tuples that must be additionally deleted to preserve referential integrity when the tuple (2,4) is deleted is: **(Gate-2005) (2 Marks)**

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

- a)** (3,4) and (6,4)
- b)** (5,2) and (7,2)
- c)** (5,2),(7,2) and (9,5)
- d)** (3,4),(4,3) and (6,4)

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

- **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 necessary contain unique values.

Q R(ABCD)

A □ B

A B C D

B □ C

C □ A

Q R(ABCD)

AB □ CD

A

B

C

D

D □ A

Q R(ABCDEF)

AB □ C

A

B

C

D

E

F

C □ D

B □ AE

Q R(ABC)

AB □ C

A

B

C

C □ A

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(ABCDEFGHIJ)

AB □ C

A B C D E F G H I J

AD □ GH

BD □ EF

A □ I

H □ J

Q R(ABCDE)

CE □ D

A B C D E

D □ B

C □ A

Q R(ABCDEFGH)

A \square BC

A B C D E F G H

ABE \square CDGH

C \square GD

D \square G

E \square F

Q R(ABCDE)

A □ B

A

B

C

D

E

BC □ E

DE □ A

Q R(ABCD)

AB □ CD

A

B

C

D

C □ A

D □ B

Q R(ABCDE)

A □ B

A

B

C

D

E

BC □ E

DE □ A

Q R(ABCDE)

A

B

C

D

E

AB □ CD

D □ A

BC □ DE

Q R(ABCDE)

BC □ ADE

A

B

C

D

E

D □ B

Q R(ABCDEF)

AB □ C

A

B

C

D

E

F

DC □ AE

E □ F

Q R(ABCDEF)

AB □ C

A

B

C

D

E

F

C □ D

D □ BE

E □ F

F □ A

Q R(WXYZ)

Z □ W

W

X

Y

Z

Y □ XZ

XW □ Y

Q R(VWXYZ)

Z □ Y

V

W

X

Y

Z

Y □ Z

X □ YV

VW □ X

Q R(ABCDEF)

ABC □ **D**

A

B

C

D

E

F

ABD □ **E**

CD □ **F**

CDF □ **B**

BF □ **D**

Q R(ABCDE)

A □ BC

A

B

C

D

E

CD □ E

B □ D

E □ A

Q R(ABCDEF)

A BCDEF

A

B

C

D

E

F

BC ADEF

DEF ABC

Q Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} \rightarrow {G}, {F} \rightarrow {I, J}, {E, H} \rightarrow {K, L}, K \rightarrow {M}, L \rightarrow {N}} on R. What is the key for R? (GATE- 2014) (1 Marks)

A) {E, F}

E F G H I J K L M N

B) {E, F, H}

C) {E, F, H, K, L}

D) {E}

Q Given the STUDENTS relation as shown below.

<i>StudentID</i>	<i>StudentName</i>	<i>StudentEmail</i>	<i>StudentAge</i>	<i>CPI</i>
2345	Shankar	shankar@math	X	9.4
1287	Swati	swati@ee	19	9.5
7853	Shankar	shankar@cse	19	9.4
9876	Swati	swati@mech	18	9.3
8765	Ganesh	ganesh@civil	19	8.7

For (StudentName, StudentAge) to be the key for this instance, the value X should not be equal to _____ **(GATE- 2014) (1 Marks)**

Q Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values. $F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$ is a set of functional dependencies (FDs) so that F^+ is exactly the set of FDs that hold for R. How many candidate keys does the relation R have? (**GATE- 2013**) (2 Marks)

(A) 3

A B C D E F G H

(B) 4

(C) 5

(D) 6

Q Consider a relational table with a single record for each registered student with the following attributes.

1. *Registration_Num*: Unique registration number of each registered student
2. *UID*: Unique identity number, unique at the national level for each citizen
3. *BankAccount_Num*: Unique account number at the bank. A student can have multiple accounts or join accounts. This attribute stores the primary account number.
4. *Name*: Name of the student
5. *Hostel_Room*: Room number of the hostel

Which one of the following option is **INCORRECT?** (GATE- 2011) (1 Marks)

- A) *BankAccount_Num* is candidate key
- B) *Registration_Num* can be a primary key
- C) *UID* is candidate key if all students are from the same country
- D) If S is a superkey such that $S \cap \text{UID}$ is NULL then $S \cup \text{UID}$ is also a superkey

Q Consider a relation scheme $R = (A, B, C, D, E, H)$ on which the following functional dependencies hold: $\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$. What are the candidate keys of R ? (GATE- 2005) (1 Marks)

(A) AE, BE

A B C D E H

(B) AE, BE, DE

(C) AEH, BEH, BCH

(D) AEH, BEH, DEH

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? **(GATE 2021) (1 MARKS)** [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 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

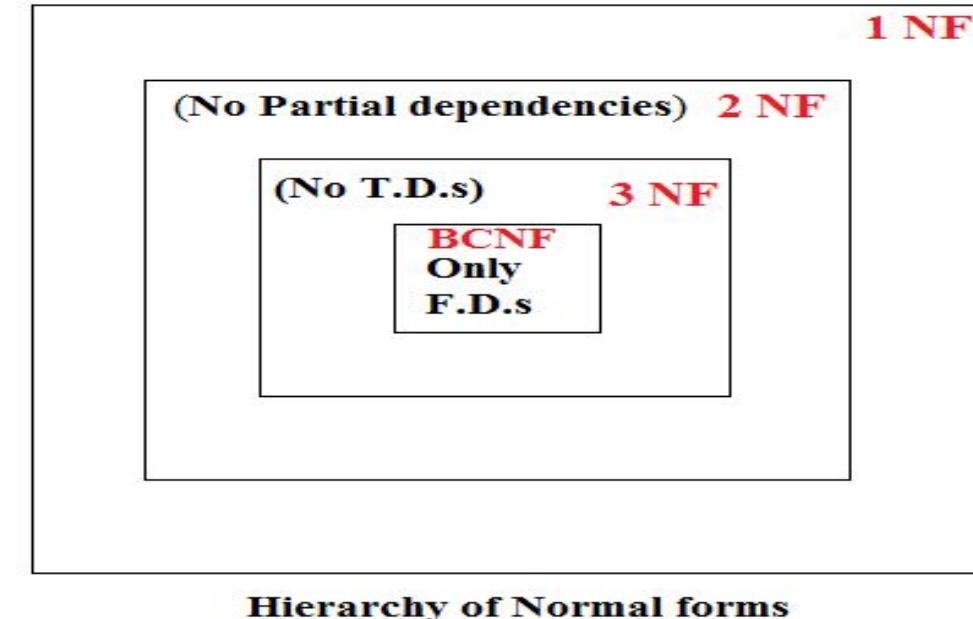
A B C D E F

b) 3

c) 4

d) 5

- **Normalization of data** (Decomposition of Relation) can be considered a process of analyzing the given relation schema to achieve the desirable properties of minimizing redundancy using Decomposition.
- The tool we use for normalization is functional dependencies and candidate keys.
- Functional dependency can be used only to normalize up to BCNF.
- A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be **normalized** to any desired degree.
- 1NF>>2NF>>3NF>>BCNF



FIRST NORMAL FORM

- A Relation table is said to be in first normal form iff each attribute in each cell have single value(atomic). Means a Relation should not contain any multivalued or composite attributes.
- Other implications of first normal form
 - Every row should be unique, that is no two rows should have the same values of all the attributes.
 - There must be a primary key.
 - Every column should have a unique name
 - Order of row and column is irrelevant

Q.Which of the following statements about a relation R in first normal form (1NF) is /are TRUE? (Gate 2024,CS)(1 Marks)(MSQ)

- (a) R can have a multi-attribute key**
- (b) R cannot have a foreign key**
- (c) R cannot have a composite attribute**
- (d) R cannot have more than one candidate key**

Name	F_name	Age	Br_code
A	P	19	101
B	Q	18	101
B	R	20	101
D	R	20	102

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)

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

Eg R(ABCD)

AB \square 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 ⊓ D, A ⊓ C

Q R(A, B, C) B \square 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

#उन_हवाओं_से_भी_जल्द_सामना_होगा
#जो_हमारे_खिलाफ_चल_रही_हैं_



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

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

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

A ⊓ B

B ⊓ C [transitive dependency]

C ⊓ D [transitive dependency]

THIRD NORMAL FORM DIRECT DEFINATION

- 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. Consider the following relational schema along with all the functional dependencies that hold on them.

R1(A,B,C,D,E): {D→E, EA→B, EB→C}

R2(A,B,C,D): {A→D, A→B, C→A}

Which of the following statement(s) is/are TRUE? **(Gate 2025)**

- A**) R1 is in 3NF
- B**) R2 is in 3NF
- C**) R1 is NOT in 3NF
- D**) R2 is NOT in 3NF

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 $\alpha \rightarrow \beta$

α must be a super key

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

{

$AB \rightarrow C$ [No violation of 2NF, 3NF, BCNF]

$C \rightarrow D$ [violation of BCNF, C not a candidate/super key]

$D \rightarrow A$ [violation of BCNF, D not a candidate/super key]

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

Some important note points on Normalization:

- If a relation R does not contain any non- trivial dependency, then R Is in BCNF.
- A Relation with two attributes is always in BCNF.
- A relation schema R consist of only simple candidate key then, R is always in 2NF but may or may not be in 3NF or BCNF.
- A Relation schema R consist of only prime attributes then R is always in 3NF, but may or may not be in BCNF.
- A relation schema R in 3NF and with only simple candidate keys, then R surely in BCNF.

Q R(ABC) (AB, BC)

AB □ C

A B C

C □ A

Q R(ABCD)(AB)

AB □ C

A B C D

B □ D

Q R(ABCDE)(CE)

CE □ D

A B C D E

D □ B

C □ A

Q R(ABCDE)(ACD, BCD, CDE)

A □ B

A B C D E

BC □ E

DE □ A

Q R(ABCD)(AB, AD, BC, CD)

AB □ CD

A B C D

C □ A

D □ B

Q R(ABCDE)(AB)

AB □ C

A B C D E

B □ D

D □ E

Q R(ABCDEFGH)(AE)

A \square BC

A B C D E F G H

ABE \square CDGH

C \square GD

D \square G

E \square F

Q R(WXYZ)(Y, XW, XZ)

Z □ W

W X Y Z

Y □ XZ

XW □ Y

Q R(ABCDEF)(ABC, ACD)

ABC □ D

A B C D E F

ABD □ E

CD □ F

CDF □ B

Q R(ABCDE)(AB)

AB □ C

A B C D E

B □ D

D □ E

Q R(ABCDE)(A, E, BC, CD)

A □ BC

A

B

C

D

E

CD □ E

B □ D

E □ A

Q (ABCDE)(ACD, BCD, CDE)

A □ B

A B C D E

BC □ E

DE □ A

Q R(ABCDE)(ABD)

A

B

C

D

E

BD \square E

A \square C

Q R(ABCDEF) (A, BC, DEF)

A □ BCDEF

A B C D E F

BC □ ADEF

DEF □ ABC

Q R(ABCDE)(ac)

A □ B

A

B

C

D

E

B □ E

C □ D

Q R(ABCDE)(AB, BC, BD)

AB \square CD

A

B

C

D

E

D \square A

BC \square DE

Q R(ABCD)(AB, BD)

AB \square CD

A

B

C

D

D \square A

Q R(ABCDEF)(BF)

AB □ C

A B C D E F

C □ D

B □ AE

Q R(ABCDEFGHIJ)(AB)

AB □ C

A B C D E F G H I J

A □ DE

B □ F

F □ GH

D □ IJ

Q R(ABCDE)(BC, CD)

BC □ ADE

A

B

C

D

E

D □ B

Q R(ABCD)(AD, BD, CD)

A □ B

A

B

C

D

B □ C

C □ A

Q R(ABCDEF)(ABD, BCD)

AB \square C

A B C D E F

DC \square AE

E \square F

Q R(VWXYZ)(VW, XW)

Z □ Y

V W X Y Z

Y □ Z

X □ YV

VW □ X

Q R(ABCDE)(AC)

A □ B

A B C D E

B □ E

C □ D

Q R(ABCDEF)(C, D, AB, BE, BF)

AB □ C

A B C D E F

C □ D

D>BE

E>F

F>A

Q In a relational data model, which one of the following statements is TRUE? (GATE 2022) (1 MARKS)

- (A) A relation with only two attributes is always in BCNF.
- (B) If all attributes of a relation are prime attributes, then the relation is in BCNF.
- (C) Every relation has at least one non-prime attribute.
- (D) BCNF decompositions preserve functional dependencies.

Q Consider the following four relational schemas. For each schema, all non-trivial functional dependencies are listed. The underlined attributes are the respective primary keys. **(GATE-2018) (2 Marks)**

Schema I: Registration (rollno, courses)

Field ‘courses’ is a set-valued attribute containing the set of courses a student has registered for.

Non-trivial functional dependency

rollno → courses

Schema II: Registration (rollno, coursid, email)

Non-trivial functional dependencies

rollno, courseid → email

email → rollno

Schema III: Registration (rollno, courseid, marks, grade)

Non-trivial functional dependencies

rollno, courseid, → marks, grade

marks → grade

Schema IV: Registration (rollno, courseid, credit)

Non-trivial functional dependencies

rollno, courseid → credit

courseid → credit

Which one of the relational schemas above is in 3NF but not in BCNF?

- (a) Schema 1** **(b) Schema 2**
(c) Schema 3 **(d) Schema 4**

Q Which one of the following statements if FALSE? (GATE- 2017) (1 Marks)

- a) Any relation with two attributes is in BCNF**
- b) A relation in which every key has only one attribute is in 2NF**
- c) A prime attribute can be transitively dependent on a key in a 3NF relation**
- d) A prime attribute can be transitively dependent on a key in a BCNF relation**

Q A database of research articles in a journal uses the following schema. **(GATE- 2016) (2 Marks)**

(VOLUME, NUMBER, STARTPAGE, ENDPAGE, TITLE, YEAR, PRICE)

The primary key is (VOLUME, NUMBER, STARTPAGE, ENDPAGE) and the following functional dependencies exist in the schema.

(VOLUME, NUMBER, STARTPAGE, ENDPAGE) → TITLE

(VOLUME, NUMBER) → YEAR

(VOLUME, NUMBER, STARTPAGE, ENDPAGE) → PRICE

The database is redesigned to use the following schemas.

(VOLUME, NUMBER, STARTPAGE, ENDPAGE, TITLE, PRICE)

(VOLUME, NUMBER, YEAR) Which is the weakest normal form that the new database satisfies, but the old one does not

(a) 1NF

(b) 2NF

(c) 3NF

(d) BCNF

Q Given the following two statements:

S_1 : Every table with two single-valued attributes is in 1NF, 2NF, 3NF and BCNF.

S_2 : $AB \rightarrow C$, $D \rightarrow E$, $E \rightarrow C$ is a minimal cover for the set of functional dependencies $AB \rightarrow C$, $D \rightarrow E$, $AB \rightarrow E$, $E \rightarrow C$.

Which one of the following is CORRECT? (GATE- 2014) (1 Marks)

A) S_1 is TRUE and S_2 is FALSE.

B) Both S_1 and S_2 are TRUE.

C) S_1 is FALSE and S_2 is TRUE.

D) Both S_1 and S_2 are FALSE.

Q Which of the following is TRUE? (GATE- 2012) (1 Marks)

- a) Every relation in 3NF is also in BCNF**

- b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R**

- c) Every relation in BCNF is also in 3NF**

- d) No relation can be in both BCNF and 3NF**

Q Relation R with an associated set of functional dependencies, F is decomposed into BCNF. The redundancy (arising out of functional dependencies) in the resulting set relations is. **(GATE- 2002)**
(1 Marks)

- a)** Zero
- b)** More than zero but less than that of an equivalent 3NF decomposition
- c)** Proportional to the size of F^+
- d)** Indeterminate

Relation R has eight attributes **ABCDEFGH**. Fields of R contain only atomic values.

$F = \{CH \rightarrow G, A \rightarrow BC, B \rightarrow CFH, E \rightarrow A, F \rightarrow EG\}$ is a set of functional dependencies (FDs) so that F^+ is exactly the set of FDs that hold for R.

The relation R is:

- (a)** in 1NF, but not in 2NF
- (b)** in 2NF, but not in 3NF
- (c)** in 3NF, but not in BCNF
- (d)** in BCNF

Q Consider the relation schema R (A B C D) with following FD set

$F = \{A \rightarrow BC, C \rightarrow D\}$; The relation R is in _____

a) 1 NF

A B C D

b) 2 NF

c) 3 NF

d) BCNF

Q Consider the relation R (ABCDE) with the FD set $F = \{A \rightarrow CE, B \rightarrow D, AE \rightarrow D\}$. Identify the highest normal form satisfied by the relation R.

a) 1 NF

A B C D E

b) 2 NF

c) 3 NF

d) BCNF

Q Consider the relation schema R (A B C D) with following FD set

$F = \{A \rightarrow BC, C \rightarrow D\}$; The relation R is in _____

a) 1 NF

A B C D

b) 2 NF

c) 3 NF

d) BCNF

Q Consider the relation R (ABCDE) with the FD set $F = \{A \rightarrow CE, B \rightarrow D, AE \rightarrow D\}$. Identify the highest normal form satisfied by the relation R.

a) 1 NF

A B C D E

b) 2 NF

c) 3 NF

d) BCNF

Q Consider the following relational schema:

Suppliers (Sid: integer, sname:string, city:string, street:string)

Parts (pid: integer, pname:string, color:string)

Catalog (sid: integer, pid:integer, cost:real)

Assume that, in the suppliers relation above, each supplier and each street within a city has a unique name, and (sname, city) forms a candidate key. No other functional dependencies are implied other than those implied by primary and candidate keys. Which one of the following is TRUE about the above schema? **(GATE- 2009) (1 Marks)**

a) The schema is in BCNF

Sid	sname	city	street
-----	-------	------	--------

b) The schema is in 3NF but not in BCNF

pid	pname	color
-----	-------	-------

c) The schema is in 2NF but not in 3NF

sid	pid	cost
-----	-----	------

d) The schema is not in 2NF

Q Consider the following relational schemas for a library database:

Book (Title, Author, Catalog_no, Publisher, Year, Price)

Collection (Title, Author, Catalog_no) with the following functional dependencies:

I. Title, Author → Catalog_no

Title	Author	Catalog_No	Publisher	Year	Price
-------	--------	------------	-----------	------	-------

II. Catalog_no → Title, Author, Publisher, Year

III. Publisher, Title, Year → Price

Assume (Author, Title) is the key for both schemas.

Which one of the following is true? (NET-JUNE-2014) (GATE- 2008)

Title	Author	Catalog_No
-------	--------	------------

(A) Both Book and Collection are in BCNF.

(B) Both Book and Collection are in 3NF.

(C) Book is in 2NF and Collection in 3NF.

(D) Both Book and Collection are in 2NF.

Q The relation scheme Student Performance (name, courseNo, rollNo, grade) has the following functional dependencies:

name, courseNo → grade

rollNo, courseNo → grade

name → rollNo

rollNo → name

Name	CourseNo	RollNo	Grade
------	----------	--------	-------

The highest normal form of this relation scheme is **(GATE- 2004) (1 Marks)**

a) 2 NF

b) 3 NF

c) BCNF

d) 4 NF

Q Consider the following functional dependencies in a database (GATE- 2003) (1 Marks)

$\text{Data_of_Birth} \rightarrow \text{Age}$

$\text{Age} \rightarrow \text{Eligibility}$

$\text{Name} \rightarrow \text{Roll_number}$

$\text{Roll_number} \rightarrow \text{Name}$

$\text{Course_number} \rightarrow \text{Course_name}$

$\text{Course_number} \rightarrow \text{Instructor}$

$(\text{Roll_number}, \text{Course_number}) \rightarrow \text{Grade}$

Roll_No Name DOB Age

The relation $(\text{Roll_number}, \text{Name}, \text{Date_of_birth}, \text{Age})$ is:

(A) In second normal form but not in third normal form

(B) In third normal form but not in BCNF

(C) In BCNF

(D) None of the above

Q. Consider a relational schema team (name, city, owner), with functional dependencies {name→city, name→owner}.

The relation team is decomposed into two relations, t1(name, city) and t2(name, owner). Which of the following statement(s) is/are TRUE? **(Gate 2025)**

- A) The relation team is NOT in BCNF.
- B) The relations t1 and t2 are in BCNF.
- C) The decomposition constitutes a lossless join.
- D) The relation team is NOT in 3 NF.

Multivalued Dependency

- Multivalued dependencies are denoted by, $A \multimap B$, Means, for every value of A, there may exist more than one value of B.
- A **trivial multivalued dependency** $X \multimap Y$ is one where either Y is a subset of X , or X and Y together form the whole set of attributes of the relation.
- If there is functional dependency from $A \rightarrow B$, then there will also a multivalued functional dependency from $A \rightarrow\rightarrow B$.

E.g. let the constraint specified by MVD in relation **Student** as

S_name $\square \square$ **Club_name**

S_name $\square \square$ **P_no**

S_Name	Club_name
Kamesh	Dance
Kamesh	Guitar

S_Name	P_no
Kamesh	123
Kamesh	789

S_Name	Club_name	P_no
Kamesh	Dance	123
Kamesh	Guitar	123
Kamesh	Dance	789
Kamesh	Guitar	789

NOTE: The above Student schema is in BCNF as no functional dependency holds on EMP,
but still redundancy due to MVD.

- Each row indicates that a given restaurant can deliver a given variety. The table has no non-key attributes because its only key is {Restaurant, Variety, Delivery Area}. Therefore, it meets all normal forms up to BCNF.
- If we assume, however, that Variety offered by a restaurant are not affected by delivery area (i.e. a restaurant offers all Variety it makes to all areas it supplies), then it does not meet 4NF. The problem is that the table features two non-trivial multivalued dependencies on the {Restaurant} attribute (which is not a super key). The dependencies are:
 - {Restaurant} $\square\square$ {Variety}
 - {Restaurant} $\square\square$ {Delivery Area}

Restaurant Delivery Permutations		
Restaurant	Variety	Delivery Area
Chatora Sweets	Samosa	Hatibagan Market
Chatora Sweets	Samosa	Chandni Chowk
Chatora Sweets	Samosa	Koramangala
Chatora Sweets	Dosa	Hatibagan Market
Chatora Sweets	Dosa	Chandni Chowk
Chatora Sweets	Dosa	Koramangala
Moolchand	Ladoo	Koramangala
Moolchand	Dosa	Koramangala
Thaggū	Samosa	Hatibagan Market
Thaggū	Samosa	Chandni Chowk
Thaggū	Ladoo	Hatibagan Market
Thaggū	Ladoo	Chandni Chowk

- If we have two or more multivalued *independent* attributes in the same relation schema, we get into a problem of having to repeat every value of one of the attributes with every value of the other attribute to keep the relation state consistent and to maintain the independence among the attributes involved. This constraint is specified by a multivalued dependency.

Delivery Areas By Restaurant	
Restaurant	Delivery Area
Chatora Sweets	Hatibagan Market
Chatora Sweets	Chandni Chowk
Chatora Sweets	Koramangala
Moolchand	Koramangala
Thaggu	Hatibagan Market
Thaggu	Chandni Chowk

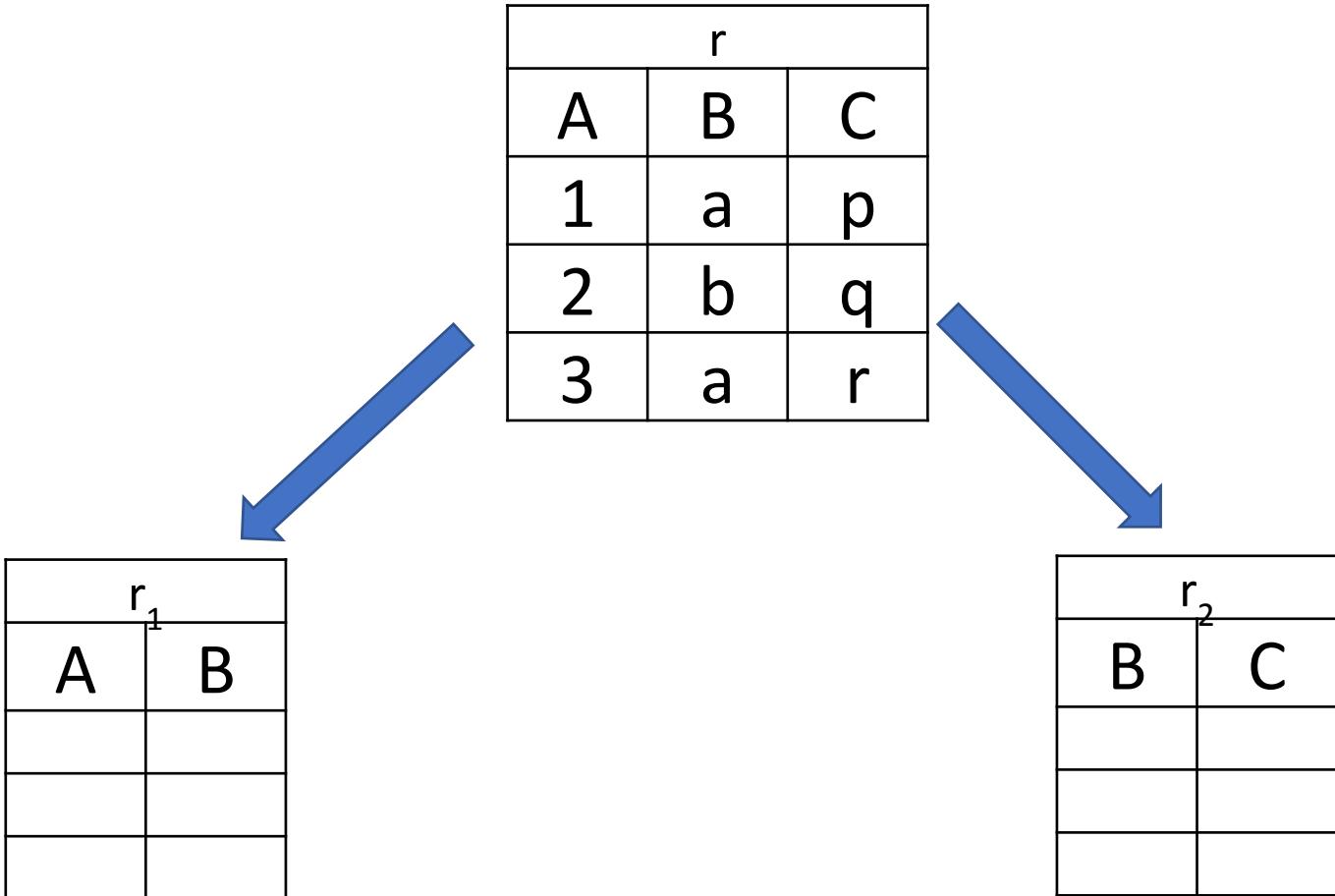
Varieties By Restaurant	
Restaurant	Pizza Variety
Chatora Sweets	Samosa
Chatora Sweets	Dosa
Moolchand	Ladoo
Moolchand	Dosa
Thaggu	Samosa
Thaggu	Ladoo

- A relation is in 4NF iff
 - It is in BCNF
 - There must not exist any non-trivial multivalued dependency.
- Each MVD is decomposed in separate table, where it becomes trivial MVD.
- A 1992 paper by Margaret S. Wu notes that the teaching of database normalization typically stops short of 4NF, perhaps because of a belief that tables violating 4NF (but meeting all lower normal forms) are rarely encountered in business applications.
- This belief may not be accurate, however. Wu reports that in a study of forty organizational databases, over 20% contained one or more tables that violated 4NF while meeting all lower normal forms.

Lossy/Lossless-Dependency Preserving Decomposition

- Because of a normalization a table is Decomposed into two or more tables, but during this decomposition we must ensure satisfaction of some properties out of which the most important is lossless join property / decomposition.

- if we decompose a table r into two tables r_1 and r_2 because of normalization then at some later stage if we want to join(combine) (natural join) these tables r_1 and r_2 , then we must get back the original table r , without any extra or less tuple. But some information may be lost during retrieval of original relation or table. For e.g.



r		
A	B	C
1	a	p
2	b	q
3	a	r

r ₁	
A	B
1	a
2	b
3	a

A	B	C

r ₂	
B	C
a	p
b	q
a	r

- Decomposition is lossy if $R_1 \bowtie R_2 \supset R$

- Decomposition is lossy if $R \supset R_1 \bowtie R_2$

- Decomposition is lossless if $R_1 \bowtie R_2 = R$ "The decomposition of relation R into R_1 and R_2 is lossless when the join of R_1 and R_2 yield the same relation as in R." which guarantees that the spurious (extra or less) tuple generation problem does not occur with respect to the relation schemas created after decomposition.
- This property is extremely critical and must be achieved at any cost.
- lossless Decomposition / NonAdditive Join Decomposition

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

How to check for lossless join decomposition using FD set, following conditions must hold:

- Union of Attributes of R_1 and R_2 must be equal to attribute of R. Each attribute of R must be either in R_1 or in R_2 . $\text{Att}(R_1) \cup \text{Att}(R_2) = \text{Att}(R)$
- Intersection of Attributes of R_1 and R_2 must not be NULL. $\text{Att}(R_1) \cap \text{Att}(R_2) \neq \emptyset$
- Common attribute must be a key for at least one relation (R_1 or R_2)
 - $\text{Att}(R_1) \cap \text{Att}(R_2) \sqsubseteq (R_1)$ or $\text{Att}(R_1) \cap \text{Att}(R_2) \sqsubseteq (R_2)$

Dependency Preserving Decomposition

- Let relation R be decomposed into Relations $R_1, R_2, R_3 \dots \dots \dots R_N$ with their respective functional Dependencies set as $F_1, F_2, F_3 \dots \dots \dots F_N$, then the Decomposition is Dependency Preserving iff
 - $\{F_1 \cup F_2 \cup F_3 \cup F_4 \dots \dots \cup F_N\}^+ = F^+$
- Dependency preservation property, although desirable, is sometimes sacrificed.

Q R (A, B, C)

A □ B, B □ C, C □ A

R₁(A, B) AND R₂(B, C)

Q Let the set of functional dependencies $F = \{QR \rightarrow S, R \rightarrow P, S \rightarrow Q\}$ hold on a relation schema $X = (PQRS)$. X is not in BCNF. Suppose X is decomposed into two schemas Y and Z , where $Y = (PR)$ and $Z = (QRS)$.

Consider the two statements given below.

I. Both Y and Z are in BCNF

II. Decomposition of X into Y and Z is dependency preserving and lossless

Which of the above statements is/are correct? **(GATE- 2019) (1 Marks)**

(a) I only

(b) Neither I nor II

(c) II only

(d) Both I and II

Q Relation R is decomposed using a set of functional dependencies, F and relation S is decomposed using another set of functional dependencies G. One decomposition is definitely BCNF, the other is definitely 3NF, but it is not known which is which. To make a guaranteed identification, which one of the following tests should be used on the decompositions? (Assume that the closures of F and G are available). **(Gate-2002) (2 Marks)**

(A) Dependency-preservation

(B) Lossless-join

(C) BCNF definition

(D) 3NF definition

Q R(A,B,C,D) is a relation. Which of the following does not have a lossless join, dependency preserving BCNF decomposition? **(Gate-2001) (1 Marks)**

(A) A \square B, B \square CD

(B) A \square B, B \square C, C \square D

(C) AB \square C, C \square AD

(D) A \square BCD

Q Consider a schema R(A,B,C,D) and functional dependencies A->B and C->D
Then the decomposition of R into $R_1(AB)$ and $R_2(CD)$ is **(GATE-2001) (2 Marks)**

- A) dependency preserving and lossless join
- (B) lossless join but not dependency preserving**
- (C) dependency preserving but not lossless join**
- (D) not dependency preserving and not lossless join**

Q Consider the relation $R(P,Q,S,T,X,Y,Z,W)$ with the following functional dependencies.

$$PQ \rightarrow X; P \rightarrow YX; Q \rightarrow Y; Y \rightarrow ZW$$

$$D_1 : R = [(P, Q, S, T); (P, T, X); (Q, Y); (Y, Z, W)]$$

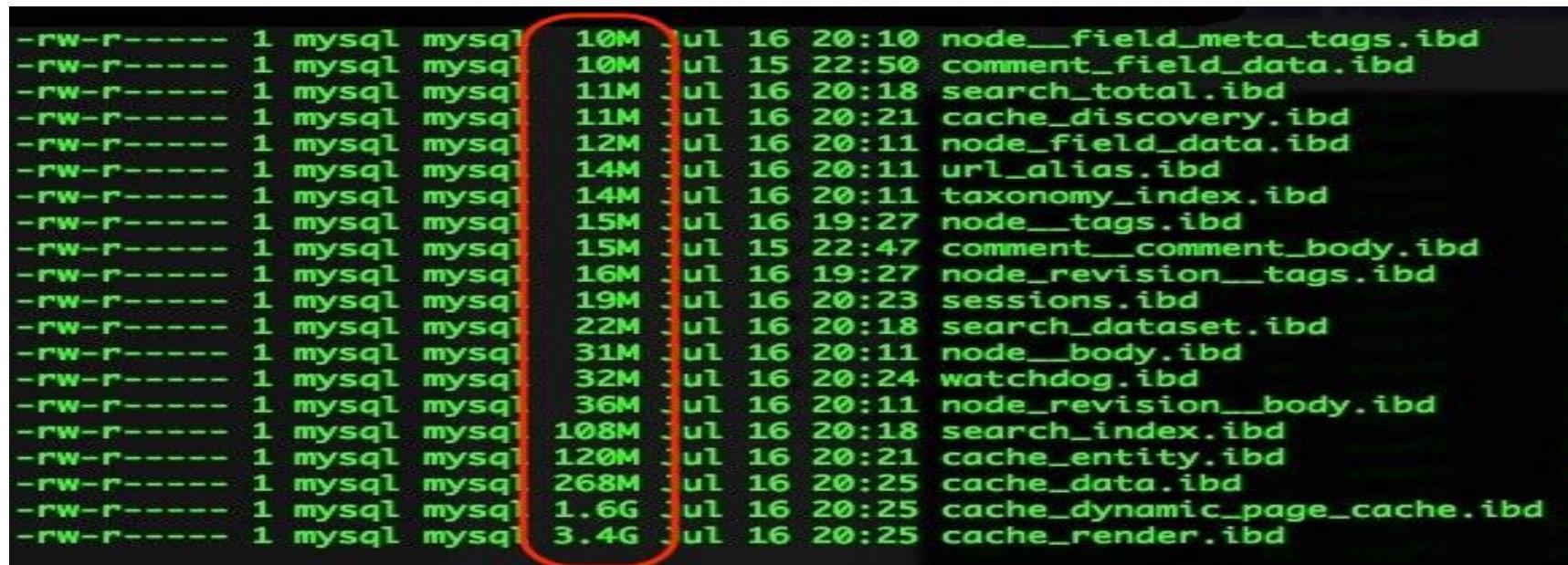
$$D_2 : R = [(P, Q, S); (T, X); (Q, Y); (Y, Z, W)]$$

Consider the decomposition of the relation R into the constituent relations according to the following two decomposition schemes. Which one of the following options is correct? **(GATE 2021) (2 MARKS)**

- a) D1 is a lossless decomposition, but D2 is a lossy decomposition
- b) D1 is a lossy decomposition, but D2 is a lossless decomposition
- c) Both D1 and D2 are lossless decompositions
- d) Both D1 and D2 are lossy decompositions

Indexing

- Theoretically relational database is derived from set theory, and in a set the order of elements in a set is irrelevant, so does in relations(tables). But in practice implementation we have to specify the order.
- A number of properties, like search, insertion and deletion will depend on the order in which elements are stored in the tables. There are only two ways in which elements can be stored in a table ordered (Sorted) or unordered (Unsorted).



```
-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
```

File organization/ Organization of records in a file

Ordered file organization/Unordered file organization

- All the records in the file are ordered on some search key field.
 - Here binary search is possible. (give example of book page searching)
 - Maintenance (insertion & deletion) is costly, as it requires reorganization of entire file.
 - Notes that we will get binary search only if we are using that key for searching on which indexing is done, otherwise it will behave as unsorted file
-
- All the records are inserted usually in the end of the file so not ordered according to any field, Because of this only linear search is possible, searching is slow.
 - Maintenance (insertion & deletion) is easy, as it does not require re organization of entire file.
-
- Reason for indexing - For a large file when it contains a large number of records which will eventually acquire large number of blocks, then its access will become slow.

- Additional auxiliary access structure is called indexes, a data technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. 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

Q Suppose we have ordered file with records stored $r = 30,000$ on a disk with Block Size $B = 1024$ B. File records are of fixed size and are unspanned with record length $R = 100$ B. Suppose that ordering key field of file is 9 B long and a block pointer is 6 B long, Implement primary indexing?

- Index typically provides secondary access path, which provide alternative way to access the records without affecting the physical placement of records in the main file.
- The size of index file is way smaller than that of the main file, as index file record contain only two columns key (attribute in which searching is done) and block pointer (base address of the block of main file which contains the record holding the key), while main file contains all the columns.
- Index can be created on any field of relation (primary key, non-key)
- One index file is designed according to an attribute, means more than one index file can be designed for a main file.
- Index file is always ordered, irrespective of weather main file is ordered or unordered. So that we can take the advantage of binary search.
- Indexing gives the advantage of faster time, but space taken by index file will be an overhead.
- Number of access required to search the correct block of main file is $\log_2(\text{number of blocks in index file}) + 1$

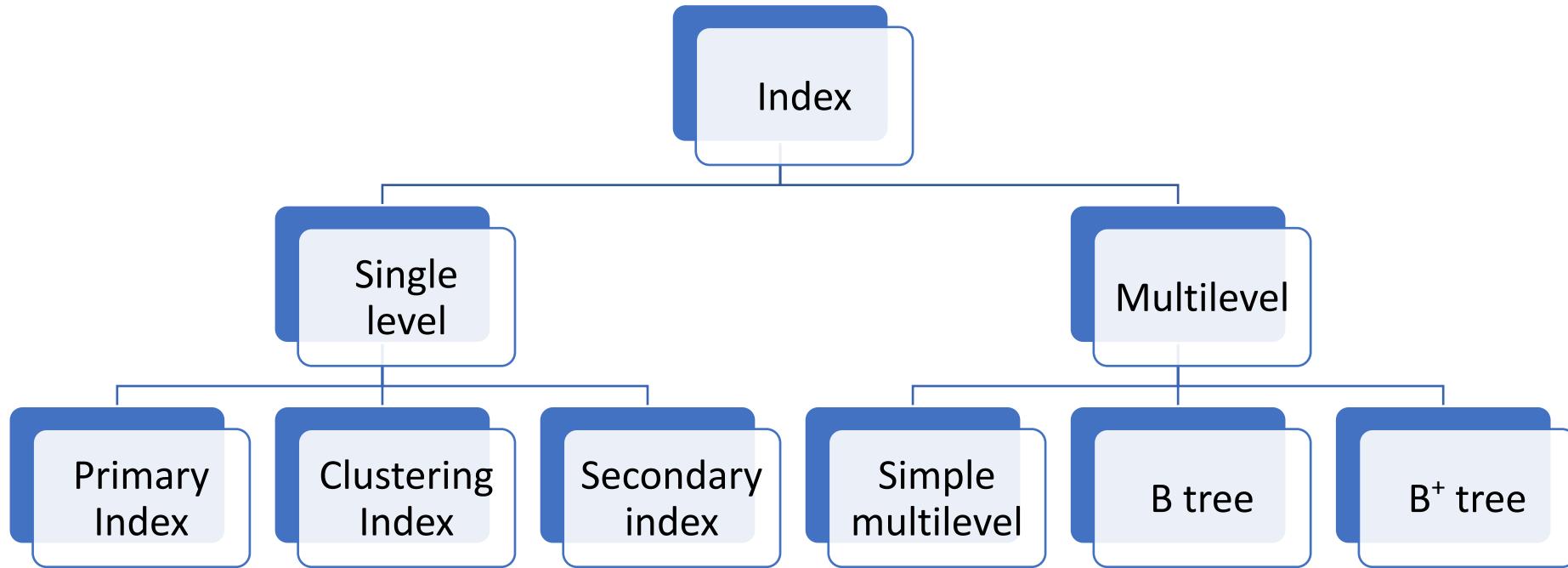
Indexing can be classified on number of criteria's one of them could be

- **Dense Index:** In dense index, there is an entry in the index file for every search key value in the main file. This makes searching faster but requires more space to store index records itself. Note that it is not for every record, it is for every search key value. Sometime number of records in the main file > number of search keys in the main file, for example if search key is repeated.
- **Sparse Index:** If an index entry is created only for some records of the main file, then it is called sparse index. No. of index entries in the index file < No. of records in the main file. Note:
- dense and sparse are not complementary to each other, sometimes it is possible that a record is both dense and sparse.

Basic term used in Indexing

- BLOCKING FACTOR = No. of Records per block= $\lceil \text{block size}/\text{record size} \rceil$
- No of blocks required by file = $\lceil \text{no of records} / \text{blocking factor} \rceil$
- If file is unordered then no of block assesses required to reach correct block which contain the desired record is $O(n)$, where n is the number of blocks.
- if file is unordered then no of block assesses required to reach correct block which contain the desired record is $O(\log_2 n)$, where n is the number of blocks.

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 An index is clustered, if (GATE-2013) (1 Marks)

- (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

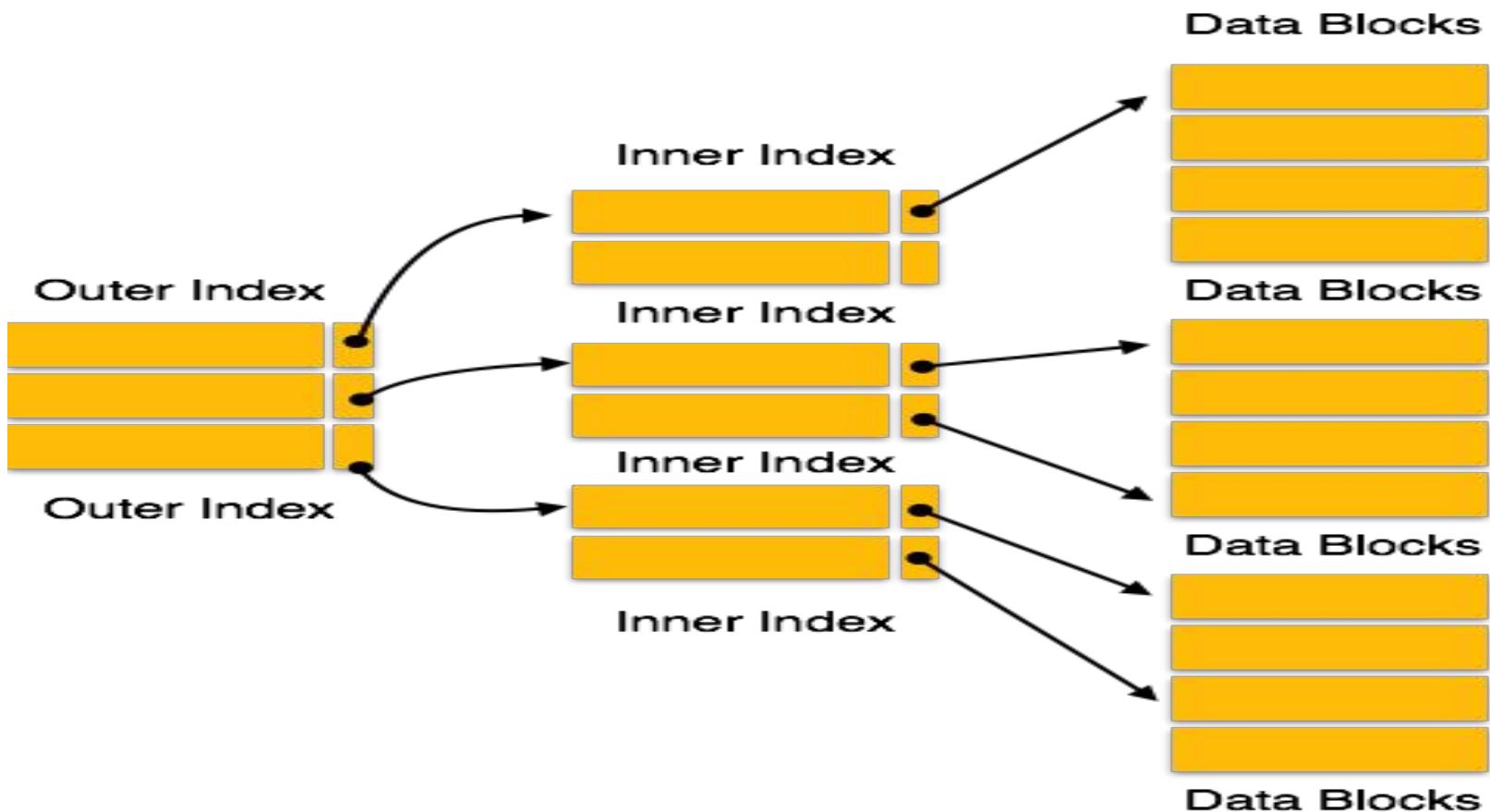
- Most common scenarios, suppose that we already have a primary indexing on primary key, but there is frequent query on some other attributes, so we may decide to have one more index file with some other attribute.
- Main file is ordered according to the attribute on which indexing is done(unordered).
- Secondary indexing can be done on key or non-key attribute.
- No of entries in the index file is same as the number of entries in the main file.
- It is an example of dense indexing.

Q Suppose we have ordered file with records stored $r = 30,000$ on a disk with Block Size $B = 1024$ B. File records are of fixed size and are unspanned with record length $R = 100$ B. Suppose that ordering key field of file is 9 B long and a block pointer is 6 B long, Implement Secondary indexing?

Q A data file consisting of 1,50,000 student-records is stored on a hard disk with block size of 4096 bytes. The data file is sorted on the primary key RollNo. The size of a record pointer for this disk is 7 bytes. Each student-record has a candidate key attribute called ANum of size 12 bytes. Suppose an index file with records consisting of two fields, ANum value and the record pointer to the corresponding student record, is built and stored on the same disk. Assume that the records of data file and index file are not split across disk blocks. The number of blocks in the index file is _____ .(GATE 2021) (1 MARKS)

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 now we understand that an index file is always sorted in nature and will be searched frequently, and sometimes index files can be so large that even we want to index the index file (Multilevel index), therefore we must search best data structure to meet our requirements.
- There are number of options in data structure like array, stack, link list, graph, table etc. but we want a data structure which support frequent insertion deletion, and modify it self accordingly but at the same time also provide speed search and give us the advantage of having a sorted data.

- If we look at the data structures option then tree seem to be the most appropriate but every kind of tree in the available option have some problems either simple tree or binary search tree or AVL tree, so we end up on designing new data structure called B-tree which are kind of specially designed for sorted stored index files in databases.
- In general, with multilevel indexing, we require dynamic structure, b and b^+ tree is generalized implementation of multilevel indexing, which are dynamic in nature, that is increasing and decreasing number of records. In the first level index file can be easily supported by other level index.
- B tree and B^+ tree also provides efficient search time, as the height of the structure is very less and they are also perfectly balanced.

Q Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is ordered on a non-key field, and the file organization is unpanned. The file is stored in a file system with block size 1024 bytes, and the size of a block pointer is 10 bytes. If the secondary index is built on the key field of the file, and a multi-level index scheme is used to store the secondary index, the number of first-level and second-level blocks in the multi-level index are respectively **(GATE-2008) (1 Marks)**

- a) 8 and 0
- b) 128 and 6
- c) 256 and 4
- d) 512 and 5

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 $\lceil m/2 \rceil$ 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).

Root

Rules	MAX	MIN
CHILD	m	0
DATA	$m-1$	1

Internal except Root

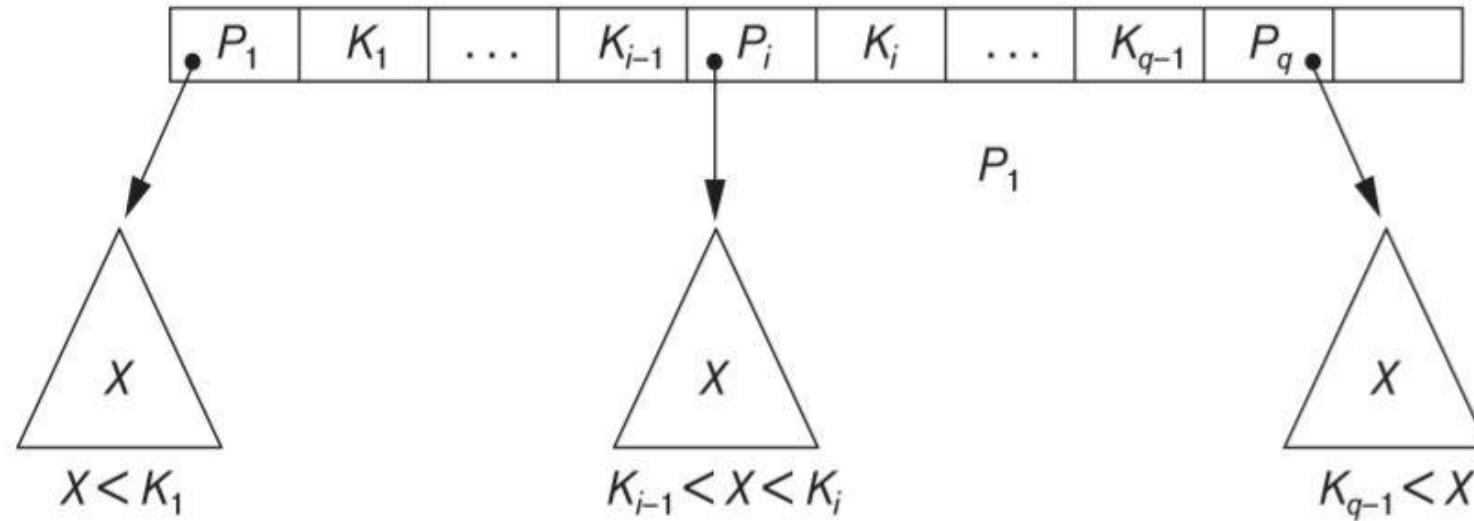
Rules	MAX	MIN
CHILD	m	$\lceil m/2 \rceil$
DATA	$m-1$	$\lceil m/2 \rceil - 1$

Leaf

Rules	MAX	MIN
CHILD	0	0
DATA	$m-1$	$\lceil m/2 \rceil - 1$

Insertion in B-TREE

- A B-tree starts with a single root node (which is also a leaf node) at level 0 (zero). Once the root node is full with $m - 1$ search key values and we attempt to insert another entry in the tree, the root node splits into two nodes at level 1.
- Only the middle value is kept in the root node, and the rest of the values are split evenly between the other two nodes. When a non-root node is full and a new entry is inserted into it, that node is split into two nodes at the same level, and the middle entry is moved to the parent node along with two pointers to the new split nodes.
- If the parent node is full, it is also split. Splitting can propagate all the way to the root node, creating a new level if the root is split.

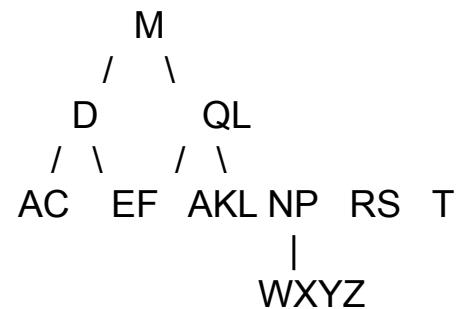


Q Consider the following elements 5, 10, 12, 13, 14, 1, 2, 3, 4
insert them into an empty b-tree of order = 3.

Q Consider the following elements 5, 10, 12, 13, 14, 1, 2, 4, 20, 18, 19, 17, 16, 15, 25, 23, 24 insert them into an empty b-tree of order = 5.

Q: Consider the Following B-tree of order $m = 6$, delete the following nodes H, T, R, E, A, C, S in sequence?

B-tree structure:



Conclusion

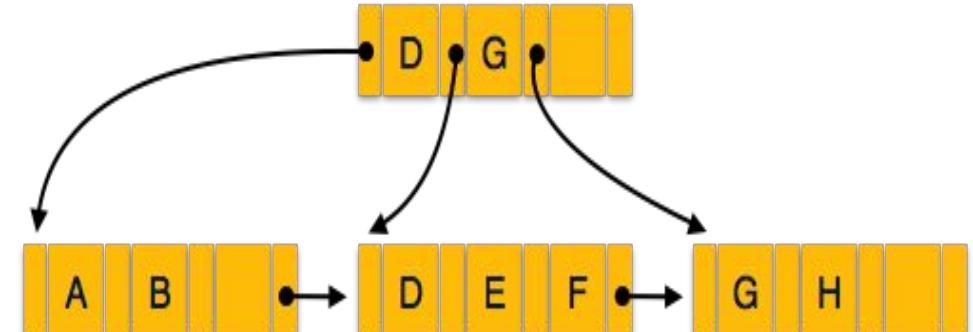
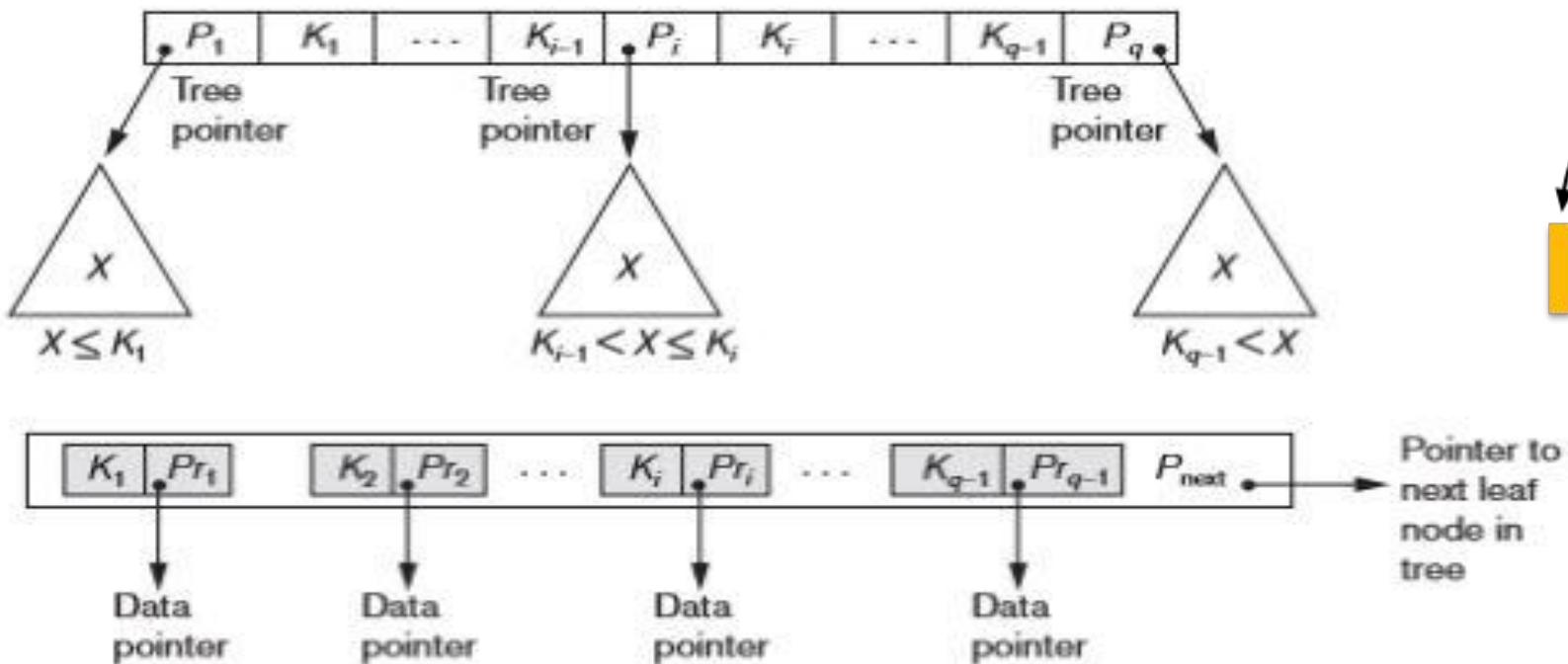
- Very less internal fragmentation, memory utilization is very good.
- Less number of nodes(blocks) are used and height is also optimized, so access will be very fast.
- Difficulty of traversing the key sequentially. Means B-TREE do not hold good for range-based queries of database.

B[±]Tree

Q Consider the following elements 5, 10, 12, 13, 14, 1, 2, 3, 4 insert them into an empty b⁺ tree of order = 3.

Insertion in B[±] Tree

- Start from root node and proceed towards leaf using the logic of binary search tree. Value is inserted in the leaf.
- If overflow condition occurs pick the median and push it into the parent node. Also copy the median or key inserted in parent node to the left or right child node.
- Repeat this procedure until tree is maintained.



The following key values are inserted into a B⁺-tree in which order of the internal nodes is 3, and that of the leaf nodes is 2, in the sequence given below. The order of internal nodes is the maximum number of tree pointers in each node, and the order of leaf nodes is the maximum number of data items that can be stored in it. The B⁺-tree is initially empty.

10, 3, 6, 8, 4, 2, 1

The maximum number of times leaf nodes would get split up as a result of these insertions is

- a) 2
- b) 3
- c) 4
- d) 5

(GATE-2009) (2 Marks)

Q. In a B+ tree, the requirement of at least half-full (50%) node occupancy is relaxed for which one of the following cases? (Gate 2024,CS) (1 Marks)(MCQ)

- (a) Only the root node**
- (b) All leaf nodes**
- (c) All internal nodes**
- (d) Only the leftmost leaf node**

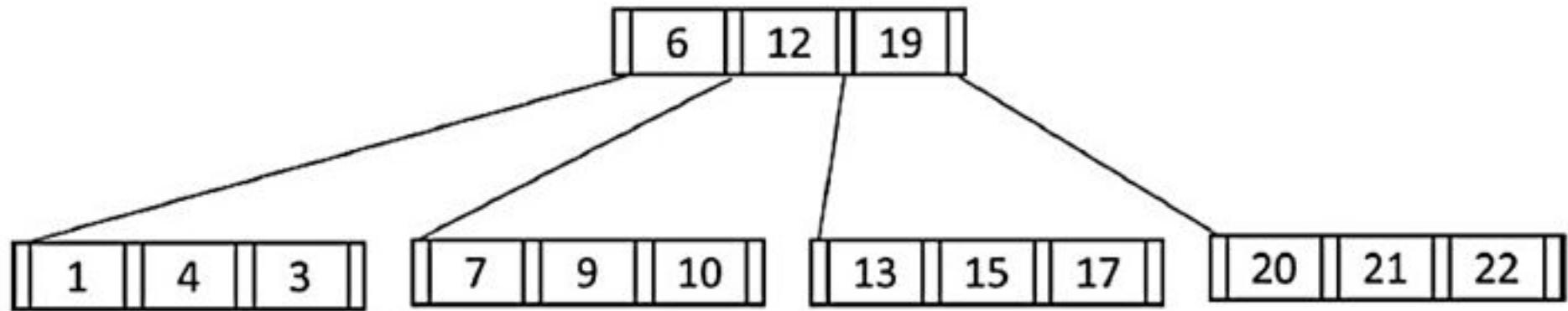
Q. In a B^+ -tree where each node can hold at most four key values, a root to leaf path consists of the following nodes:

$$A=(49, 77, 83, -) \quad B=(7, 19, 33, 44) \quad C=(20*, 22*, 25*, 26*)$$

The *-marked keys signify that these are data entries in a leaf. Assume that a pointer between keys k_1 and k_2 points to a subtree containing keys in $[k_1, k_2]$, and that when a leaf is created, the smallest key in it is copied up into the parent. A record with key value 23 is inserted into the

B^+ -tree. The smallest key value in the parent of the leaf that contains 25* is _____ (Answer in integer) **(Gate 2025)**

Q. Consider the following B+ tree with 5 nodes, in which a node can store at most 3 key values
The value 23 is now inserted in the B+ tree. Which of the following option(s) is/are CORRECT?
(Gate 2025)



- A) None of the nodes will split.
- B) At least one node will split and redistribute.
- C) The total number of nodes will remain same.
- D) The height of the tree will increase.

Q In a B+ tree, if the search-key value is 8 bytes long, the block size is 512 bytes and the block pointer is 2 bytes, then the maximum order of the B+ tree is.

(GATE-2017) (2 Marks)

Q Consider B+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that can be accommodated in each non-leaf node of the tree is **(Gate-2015) (2 Marks)**

Q The order of a leaf node in a B^+ -tree is the maximum number of (value, data record pointer) pairs it can hold. Given that the block size is 1K bytes, data record pointer is 7 bytes long, the value field is 9 bytes long and a block pointer is 6 bytes long, what is the order of the leaf node? **(GATE-2007) (1 Marks)**

a) 63

b) 64

c) 67

d) 68

Q In a database file structure, the search key field is 9 bytes long, the block size is 512 bytes, a record pointer is 7 bytes and a block pointer is 6 bytes. The largest possible order of a non-leaf node in a B+ tree implementing this file structure is **(Gate-2006) (2 Marks)**

(A) 23

(B) 24

(C) 34

(D) 44

Q The order of an internal node in a B+ tree index is the maximum number of children it can have. Suppose that a child pointer takes 6 bytes, the search field value takes 14 bytes, and the block size is 512 bytes. What is the order of the internal node? **(Gate-2004) (2 Marks)**

(A) 24

(B) 25

(C) 26

(D) 27

Q Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes the maximum number of tree pointers in a B-tree index node. Assume that K is 10 bytes long; disk block size is 512 bytes; each data pointer P_D is 8 bytes long and each block pointer P_B is 5 bytes long. In order for each B-tree node to fit in a single disk block, the maximum value of p is **(Gate-2004) (2 Marks)**

(A) 20

(B) 22

(C) 23

(D) 32

Q A B+ -tree index is to be built on the Name attribute of the relation STUDENT. Assume that all student names are of length 8 bytes, disk block are size 512 bytes, and index pointers are of size 4 bytes. Given this scenario, what would be the best choice of the degree (i.e. the number of pointers per node) of the B+ -tree? **(Gate-2002) (2 Marks)**

(A) 16

(B) 42

(C) 43

(D) 44

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? **(GATE-2019) (1 Marks)**

(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

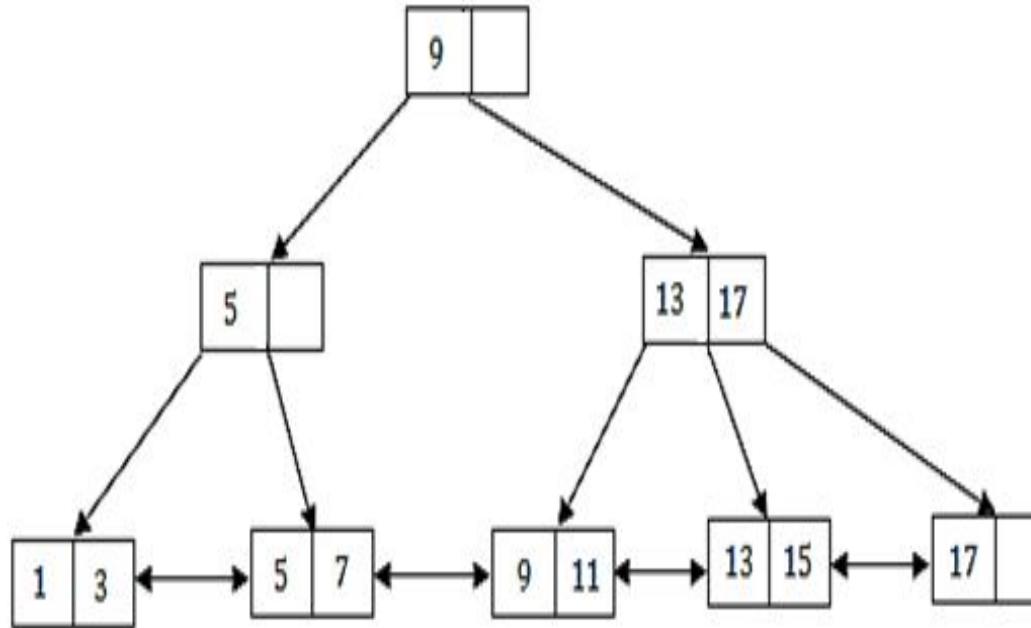
(d) Key values in each node are kept in sorted order

Q B⁺ Trees are considered **BALANCED** because **(GATE-2016) (1 Marks)**

- 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

Q (GATE-2015) (1 Marks)

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 Which of the following is correct? (Gate-1999) (1 Marks)

- a)** B-trees are for storing data on disk and B+ trees are for main memory.
- b)** Range queries are faster on B+ trees.
- c)** B-trees are for primary indexes and B++ trees are for secondary indexes.
- d)** The height of a B+ tree is independent of the number of records.

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).

Root

Rules	MAX	MIN
CHILD	m	0
DATA	$m-1$	1

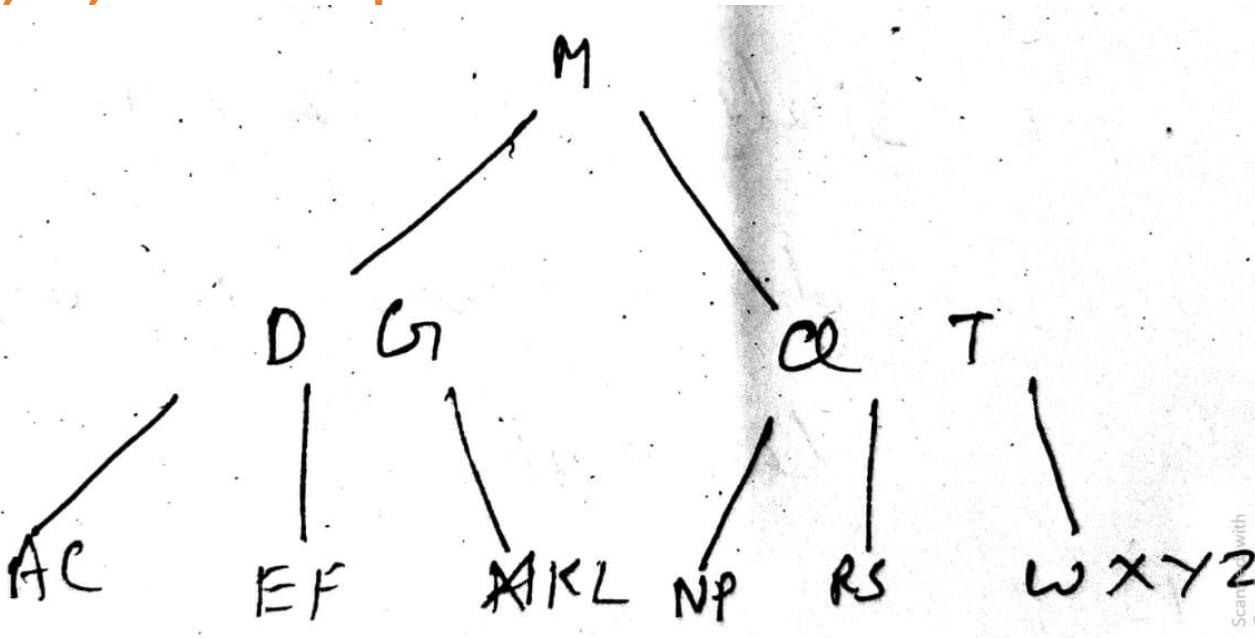
Internal except Root

Rules	MAX	MIN
CHILD	m	$\lceil m/2 \rceil$
DATA	$m-1$	$\lceil m/2 \rceil - 1$

Leaf

Rules	MAX	MIN
CHILD	0	0
DATA	$m-1$	$\lceil m/2 \rceil - 1$

Q Consider the Following B-tree of order m=6, delete the following nodes H, T, R, E, A, C, S in sequence?



Deletion in B-TREE

- If the deletion is from the leaf node and leaf node is satisfying the minimal condition even after the deletion, then delete the value directly.
- If deletion from leaf node renders leaf node in minimal condition, then first search the extra key in left sibling and then in the right sibling. Largest value from left sibling or smallest value from right sibling is pushed into the root node and corresponding value can be fetched from parent node to leaf node.
- If the deletion is to be from internal node, then first we check for the extra key in the left and then in the right child. If we find one, we fetch the value in the required node. And delete the key.

- If deletion of a value causes a node to be less than half full, it is combined with its neighboring nodes, and this can also propagate all the way to the root. Hence, deletion can reduce the number of tree levels.
- It has been shown by analysis and simulation that, after numerous random insertions and deletions on a B-tree, the nodes are approximately 69 percent full when the number of values in the tree stabilizes. This is also true of B^+ trees.
- If this happens, node splitting and combining will occur only rarely, so insertion and deletion become quite efficient. If the number of values grows, the tree will expand without a problem—although splitting of nodes may occur, so some insertions will take more time.

Q Consider the following elements 5, 8, 1, 7, 3, 12, 9, 6 insert them into an empty b^+ tree of order = 3. and then delete following nodes in sequence 9, 8, 12?

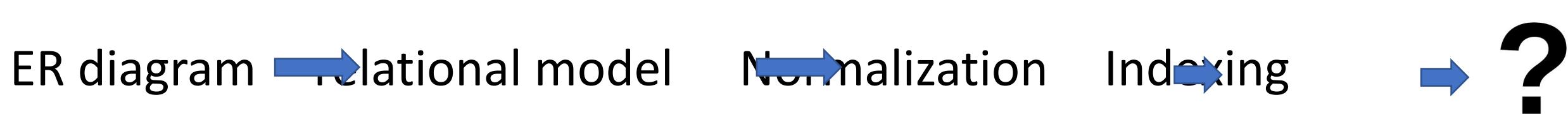
Q Consider a B⁺-tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node? **(GATE-2010) (1 Marks)**

Q. Which of the following file organizations is/are I/O efficient for the scan operation in DBMS? (Gate 2024 CS) (1 Marks)(MSQ)

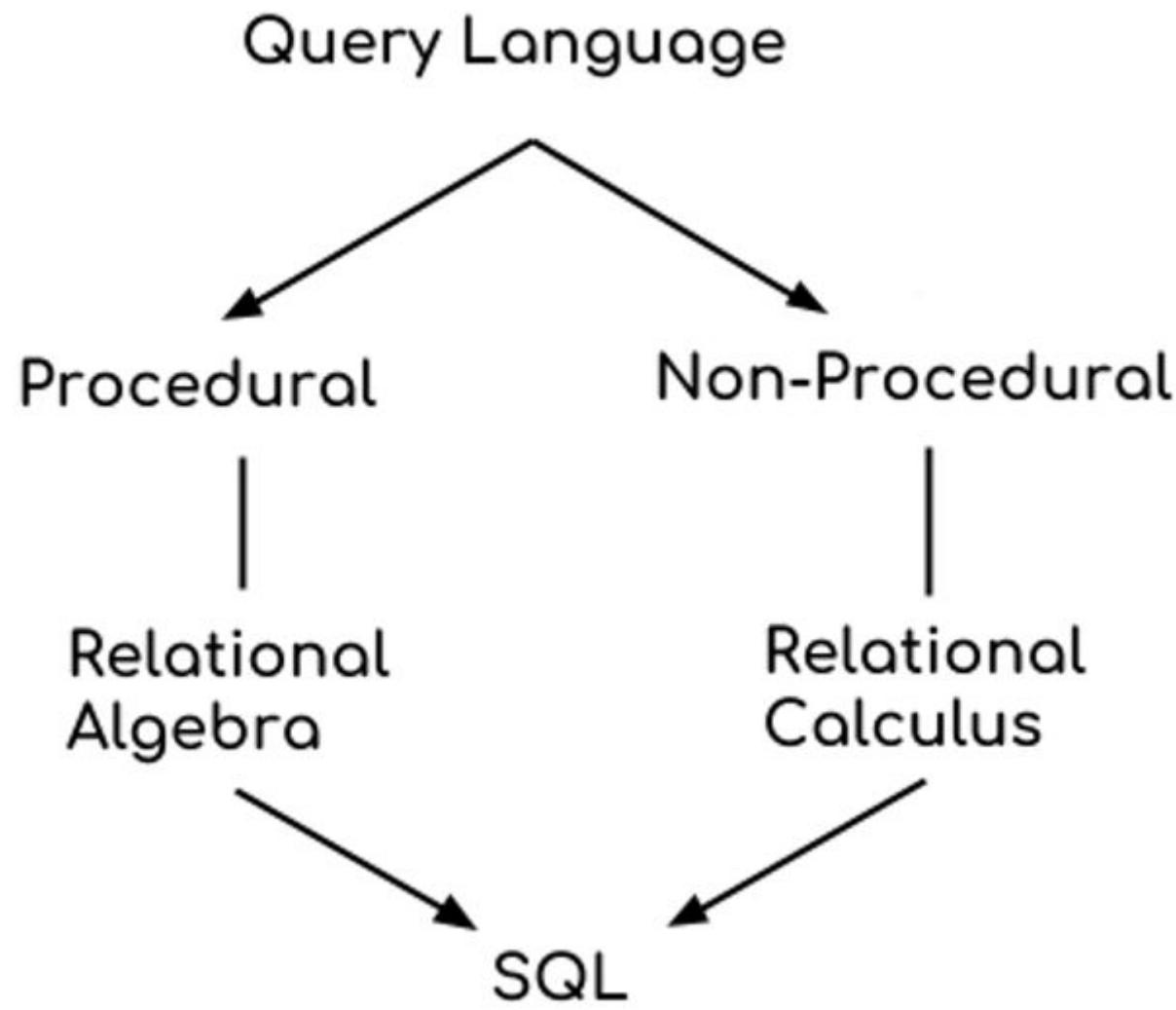
- (a) Sorted
- (b) Heap
- (c) Unclustered tree index
- (d) Unclustered hash index

Query Language

- After designing a data base, that is ER diagram followed by conversion in relational model followed by normalization and indexing, now next task is how to store, retrieve and modify data in the data base.
- Thought here we will be concentrating more on the retrieval part. Query languages are used for this purpose.



- **Query languages, data query languages or database query languages (DQLs)** are computer languages using which user request some information from the database. A well known example is the **Structured Query Language (SQL)**.



- [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;
- TSQL 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]

- **Procedural Query Language**

- Here users instruct the system to performs a sequence of operations on the data base in order to compute the desired result. Means user provides both what data to be retrieved and how data to be retrieved. e.g. Relational Algebra.

- **Non-Procedural Query Language**

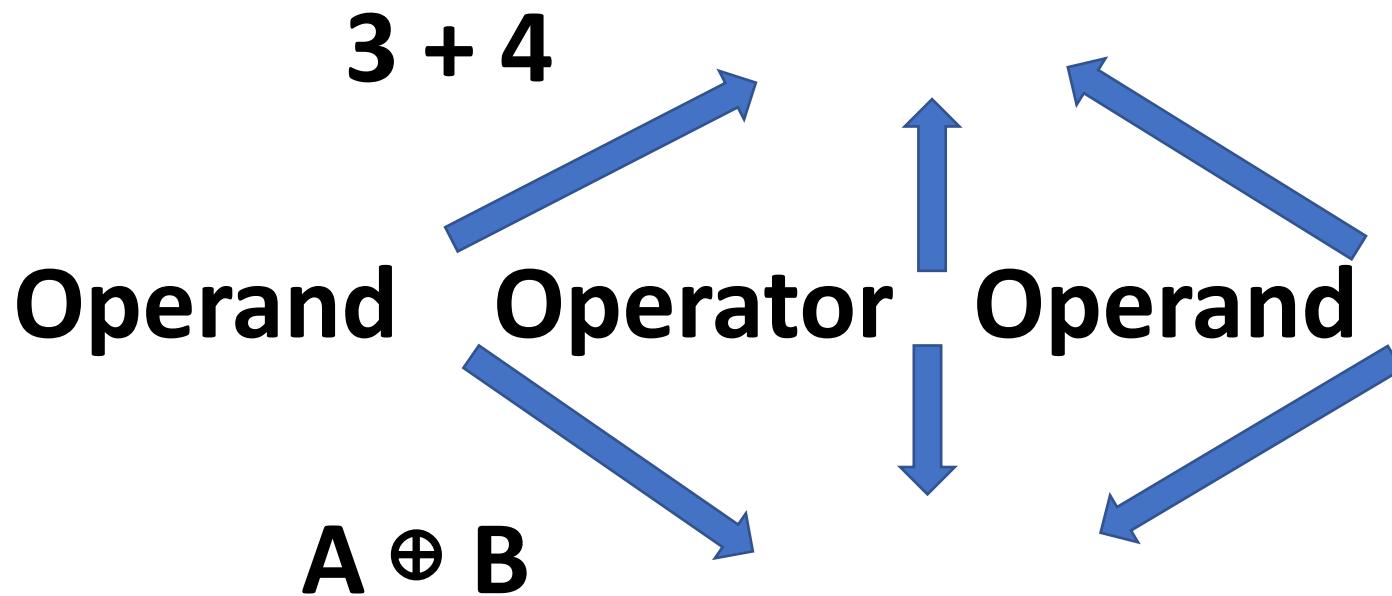
- In nonprocedural language, the user describes the desired information without giving a specific procedure for obtaining that information. What data to be retrieved e.g. Relational Calculus. **Tuple relational calculus, Domain relational calculus** are declarative query languages based on mathematical logic

- Relational Algebra (Procedural) and Relational Calculus (non-procedural) are mathematical system/ query languages which are used for query on relational model.
- RA and RC are not executed in any computer they provide the fundamental mathematics on which SQL is based.
- SQL (structured query language) works on RDBMS, and it includes elements of both procedural or non-procedural query language.

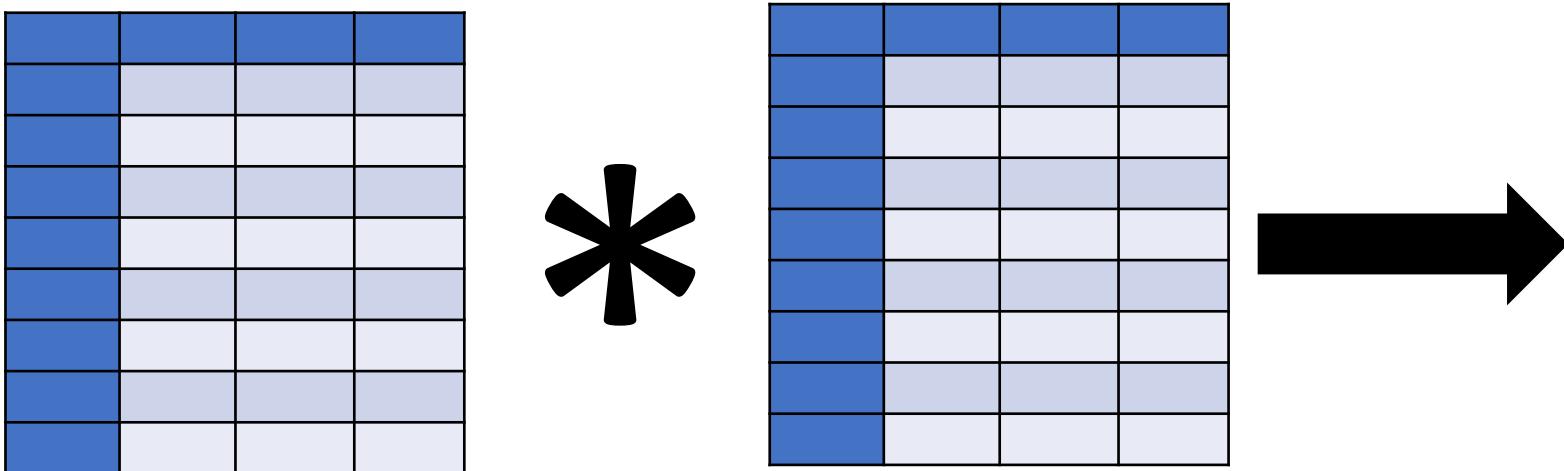
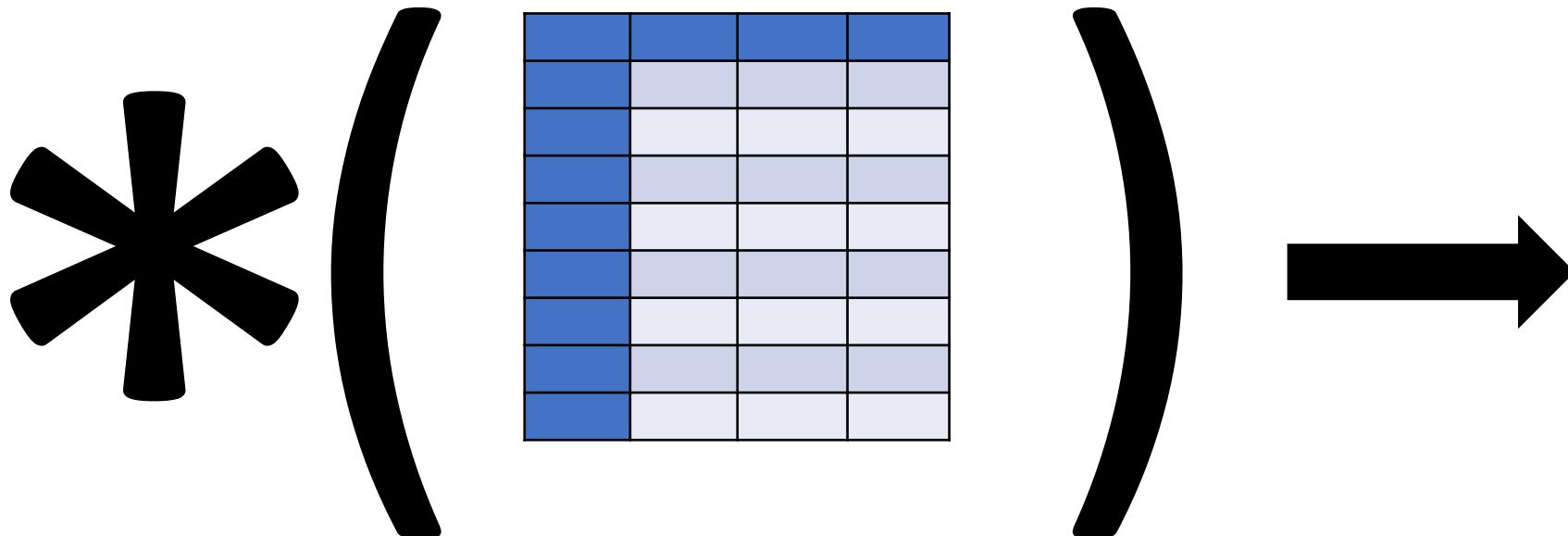
Relational model	RDBMS
RA, RC	SQL
Algo	Code
Conceptual	Reality
Theoretical	Practical
Chess	Battle Field

RELATIONAL ALGEBRA

- RA like any other mathematical system provides a number of operators and use relations (tables) as operands and produce a new relation as their result.



- Every operator in the RA accepts (one or two) relation/table as input arguments and returns always a single relation instance as the result without a name.



- It also does not consider duplicity by default as it is based on set theory. Same query is written in RA and SQL the result may be different as SQL considers duplication.
- As it is pure mathematics no use of English keywords. Operators are represented using symbols.

BASIC / FUNDAMENTAL OPERATORS

- The fundamental operations in the relational algebra are **select, project, union, set difference, Cartesian product, and Rename**.

Name	Symbol
Select	(σ)
Project	(Π)
Union	(\cup)
Set difference	$(-)$
Cross product	(\times)
Rename	(ρ)

DERIVED OPERATORS

- There are several other operations namely: **set intersection, natural join, and assignment.**

Name	Symbol	DERIVED FROM
Join	(\bowtie)	(X)
Intersection	(\cap)	(-) $A \cap B = A - (A - B)$
Division	(\div)	(X, -, Π)
Assignment	(=)	

- The **select**, **project**, and **rename** operations are called **unary operations**, because they operate on one relation.
- **Union**, **Cartesian product** and **set difference** operate on pairs of relations and are, therefore, called **binary operations**.
- Relational algebra also provides the framework for query optimization.

Relational schema - A **relation schema** R, denoted by $R(A_1, A_2, \dots, A_n)$, is made up of a relation name R and a list of attributes, A_1, A_2, \dots, A_n . Each **attribute** A_i is the name of a role played by some domain D in the relation schema R. It is used to describe a Relation.

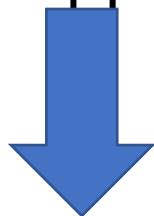
E.g. Schema representation of Table **Student** is as –

STUDENT (NAME, ID, CITY, COUNTRY, HOBBY).

Relational Instance - Relations with its data at particular instant of time.

The Project Operation (Vertical Selection)

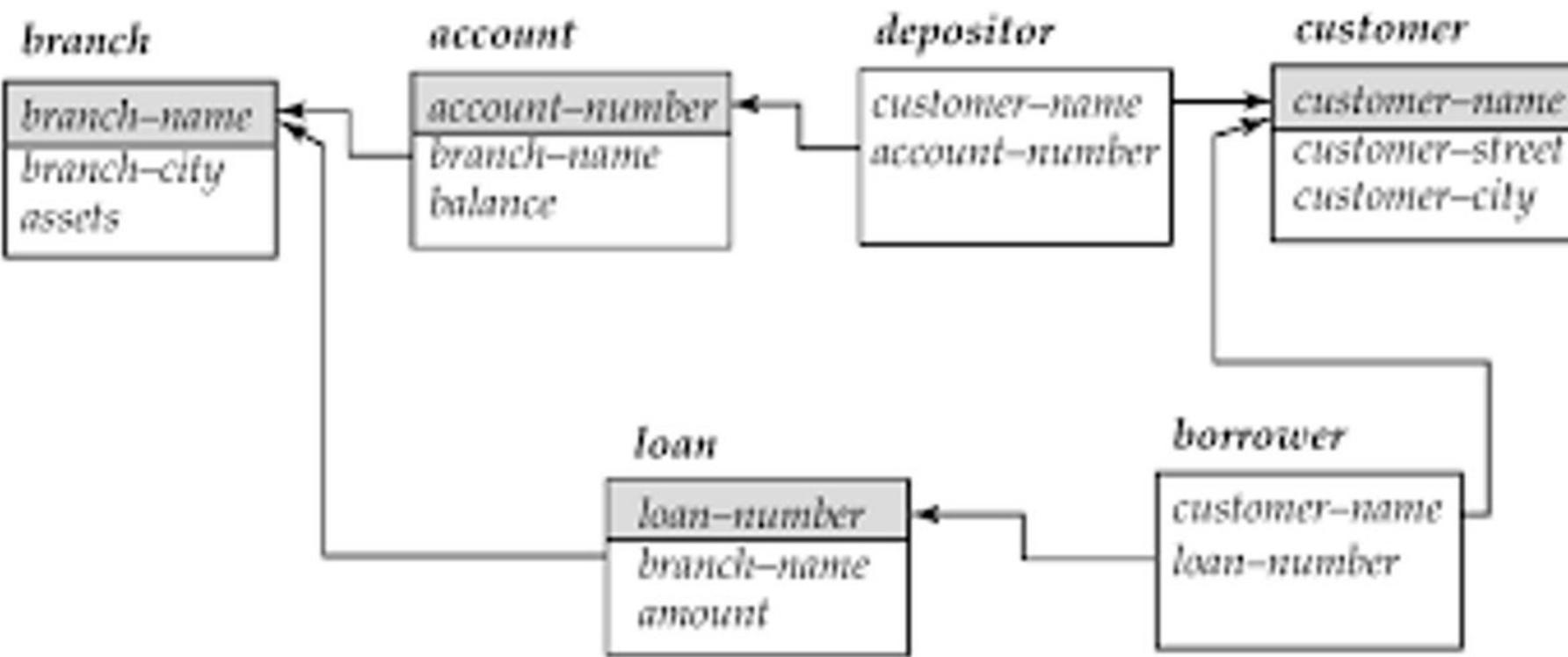
- Main idea behind project operator is to select desired columns.
- The project operation is a unary operation that returns its argument relation, with certain attributes left out.
- Projection is denoted by the uppercase Greek letter pi (Π).
- $\Pi_{\text{column_name}} (\text{table_name})$



π

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

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



- We list those attributes that we wish to appear in the result as a subscript to Π , argument relation follows in parentheses.
 - Minimum number of columns selected can be 1, Maximum selected Columns can be n - 1.



- **Some points to remember**

- Eliminates duplicate rows in a result relation by default.
- $\prod_{A_1, A_2, A_n} (r)$, A_1, A_2, \dots, A_n refers to the set of attributes to be projected.
- It is not commutative.

Q Suppose $R_1(A, B)$ and $R_2(C, D)$ are two relation schemas. Let r_1 and r_2 be the corresponding relation instances. B is a foreign key that refers to C in R_2 . If data in r_1 and r_2 satisfy referential integrity constraints, which of the following is ALWAYS TRUE? **(Gate-2012) (2 Marks)**

- a) $\prod_B(r_1) - \prod_C(r_2) = \emptyset$
- b) $\prod_C(r_2) - \prod_B(r_1) = \emptyset$
- c) $\prod_B(r_1) = \prod_C(r_2)$
- d) $\prod_B(r_1) - \prod_C(r_2) \neq \emptyset$

Q Suppose $R_1(A, B)$ and $R_2(C, D)$ are two relation schemas. Let r_1 and r_2 be the corresponding relation instances. B is a foreign key that refers to C in R_2 . If data in r_1 and r_2 satisfy referential integrity constraints, which of the following is ALWAYS TRUE? (Gate-2012) (2 Marks)

- a) $\prod_B(r_1) - \prod_C(r_2) = \emptyset$ b) $\prod_C(r_2) - \prod_B(r_1) = \emptyset$
- c) $\prod_B(r_1) = \prod_C(r_2)$ d) $\prod_B(r_1) - \prod_C(r_2) \neq \emptyset$

Roll no(A)	Br_code(B)
1	101
2	101
3	101
4	102

Br_code(C)	Br_name(D)
101	CS
102	EC
103	ME

The Select Operation (Horizontal Selection)



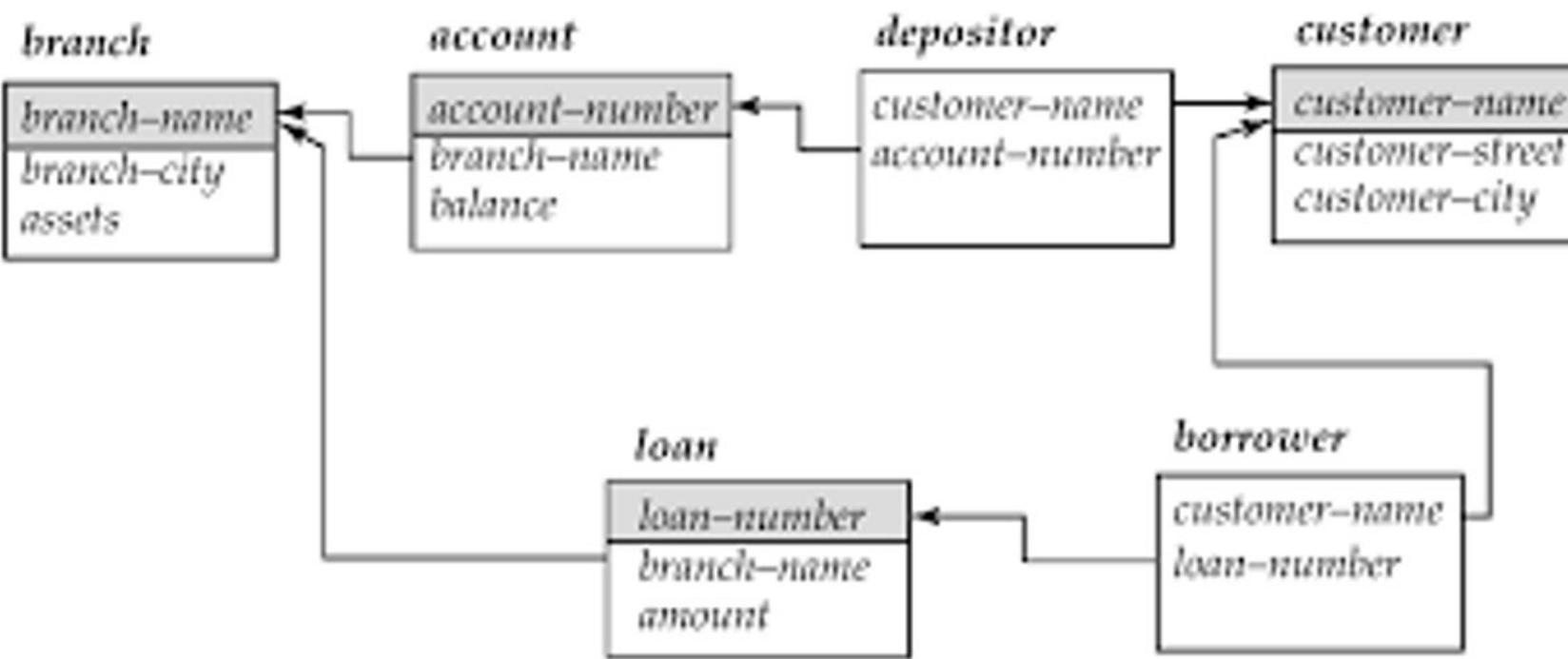
- The select operation selects tuples that satisfy a given predicate/Condition p.
- Lowercase Greek letter sigma (σ) is used to denote selection.
- It is a unary operator.
- Eliminates only tuples/rows.
- $\sigma_{\text{condition}} (\text{table_name})$



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

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

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



- Predicate appears as a subscript to σ , the argument relation is in parentheses after the σ .
- Commutative in Nature, $\sigma_{p_2 \wedge p_1}(r) = \sigma_{p_1 \wedge p_2}(r) = \sigma_{p_1}(\sigma_{p_2}(r)) = \sigma_{p_2}(\sigma_{p_1}(r))$



- Some points to remember

- We allow comparisons using $=$, \neq , $<$, $>$, \leq and \geq in the selection predicate.
- Using the connectives and (\wedge), or (\vee), and not (\neg), we can combine several predicates into a larger predicate.
- Minimum number of tuples selected can be 0, Maximum selected tuples can be all.
- Degree (Result relation) = degree (parent relation), where degree refers to no. of attributes.
- $0 \leq$ cardinality (result relation) \leq cardinality (parent relation), where cardinality refers to no. of tuples.

Q What is the optimized version of the relation algebra expression $\pi_{A_1}(\pi_{A_2}(\sigma_{F_1}(\sigma_{F_2}(r))))$, where A_1, A_2 are sets of attributes in r with $A_1 \subset A_2$ and F_1, F_2 are Boolean expressions based on the attributes in r ? **(Gate-2014) (2 Marks)**

a) $\pi_{A_1}(\sigma_{(F_1 \wedge F_2)}(r))$

b) $\pi_{A_1}(\sigma_{(F_1 \vee F_2)}(r))$

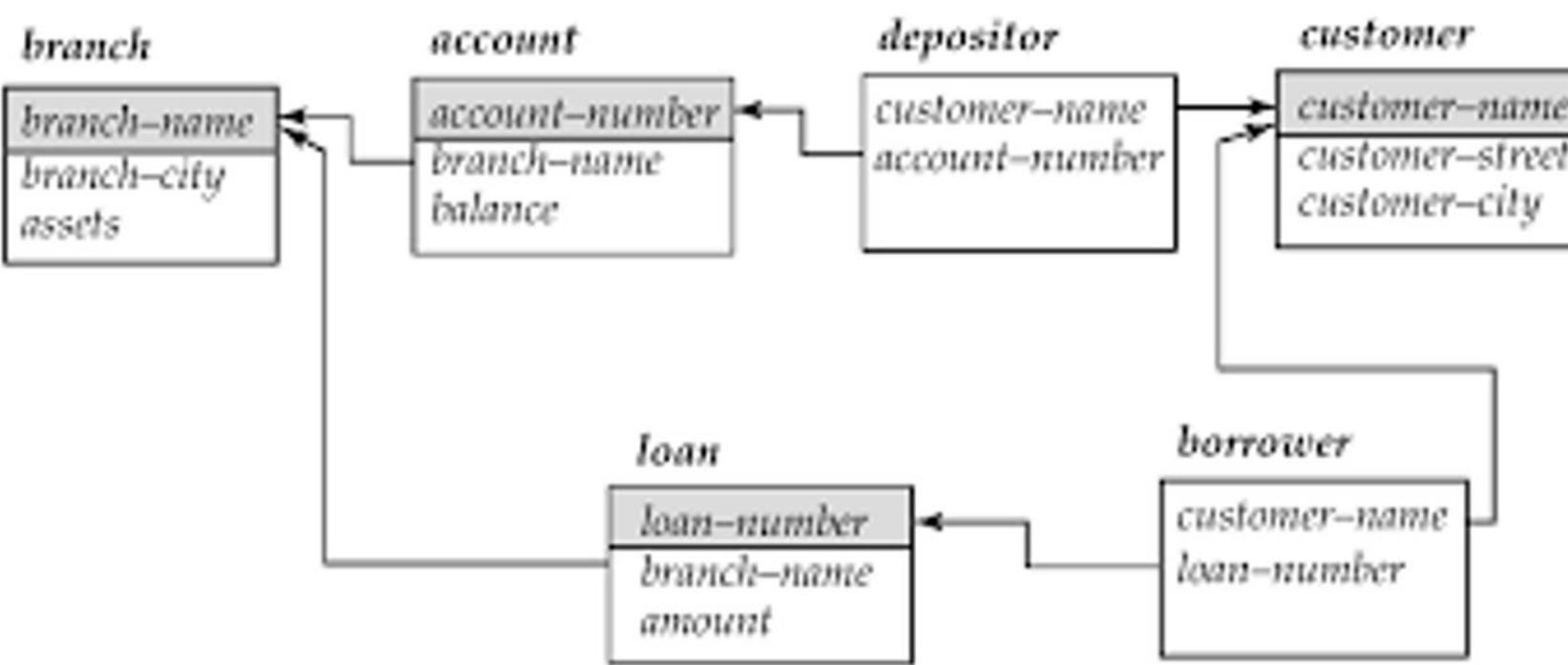
c) $\pi_{A_2}(\sigma_{(F_1 \wedge F_2)}(r))$

d) $\pi_{A_2}(\sigma_{(F_1 \vee F_2)}(r))$

Q Which of the following query transformations (i.e., replacing the l.h.s. expression by the r.h.s. expression) is incorrect? R_1 and R_2 are relations. C_1, C_2 are selection conditions and A_1, A_2 are attributes of R_1 . **(Gate-1998) (2 Marks)**

- a) $\sigma_{C_1}(\sigma_{C_2} R_1) \rightarrow \sigma_{C_2}(\sigma_{C_1}(R_1))$
- b) $\sigma_{C_1}(\pi_{A_1} R_1) \rightarrow \pi_{A_1}(\sigma_{C_1}(R_1))$
- c) $\sigma_{C_1}(R_1 \cup R_2) \rightarrow \sigma_{C_1}(R_1) \cup \sigma_{C_1}(R_2)$
- d) $\pi_{A_1}(\sigma_{C_1}(R_1)) \rightarrow \sigma_{C_1}(\pi_{A_1}(R_1))$

Q Write a RELATIONAL ALGEBRA query to find all the customer name who have a loan or an account or both?

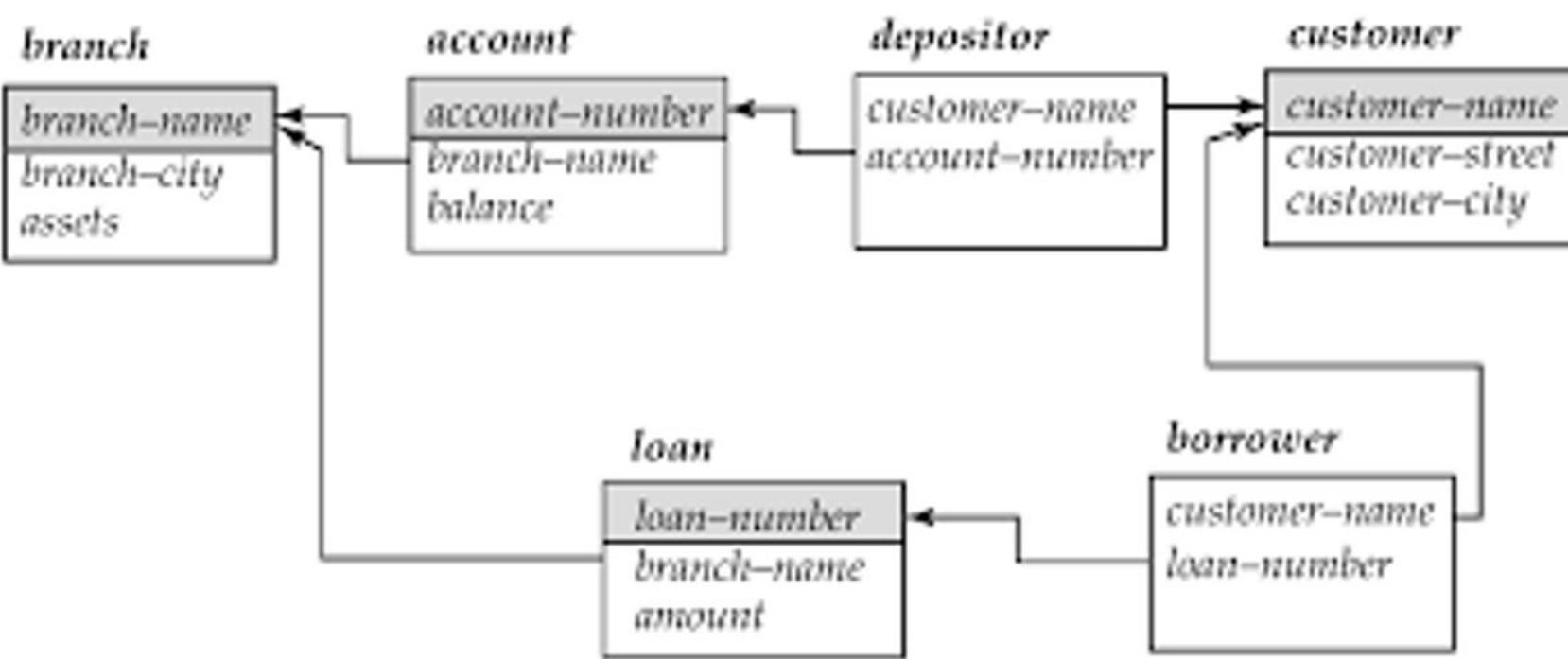


The Union Operation



- It is a binary operation, denoted, as in set theory, by \cup .
- Written as, Expression₁ \cup Expression₂, $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$
- For a union operation $r \cup s$ to be valid, we require that two conditions hold:
 - The relations r and s must be of the same arity. That is, they must have the same number of attributes, the domains of the i th attribute of r and the i th attribute of s must be the same, for all i .
 - Mainly used to fetch data from different relations.
- $\text{Deg}(R \cup S) = \text{Deg}(R) = \text{Deg}(S)$
- $\text{Max}(|R|, |S|) \leq |R \cup S| \leq (|R|+|S|)$

Q Write a RELATIONAL ALGEBRA query to find all the customer name who have a loan but do not have an account?



The Set-Difference Operation

- The set-difference operation, denoted by $-$, allows us to find tuples that are in one relation but are not in another. It is a binary operator.
- The expression $r - s$ produces a relation containing those tuples in r but not in s .
- We must ensure that set differences are taken between compatible relations.
- For a set-difference operation $r - s$ to be valid, we require that the relations r and s be of the same arity, and that the domains of the i th attribute of r and the i th attribute of s be the same, for all i .
 - $0 \leq |R - S| \leq |R|$

The Cartesian-Product Operation

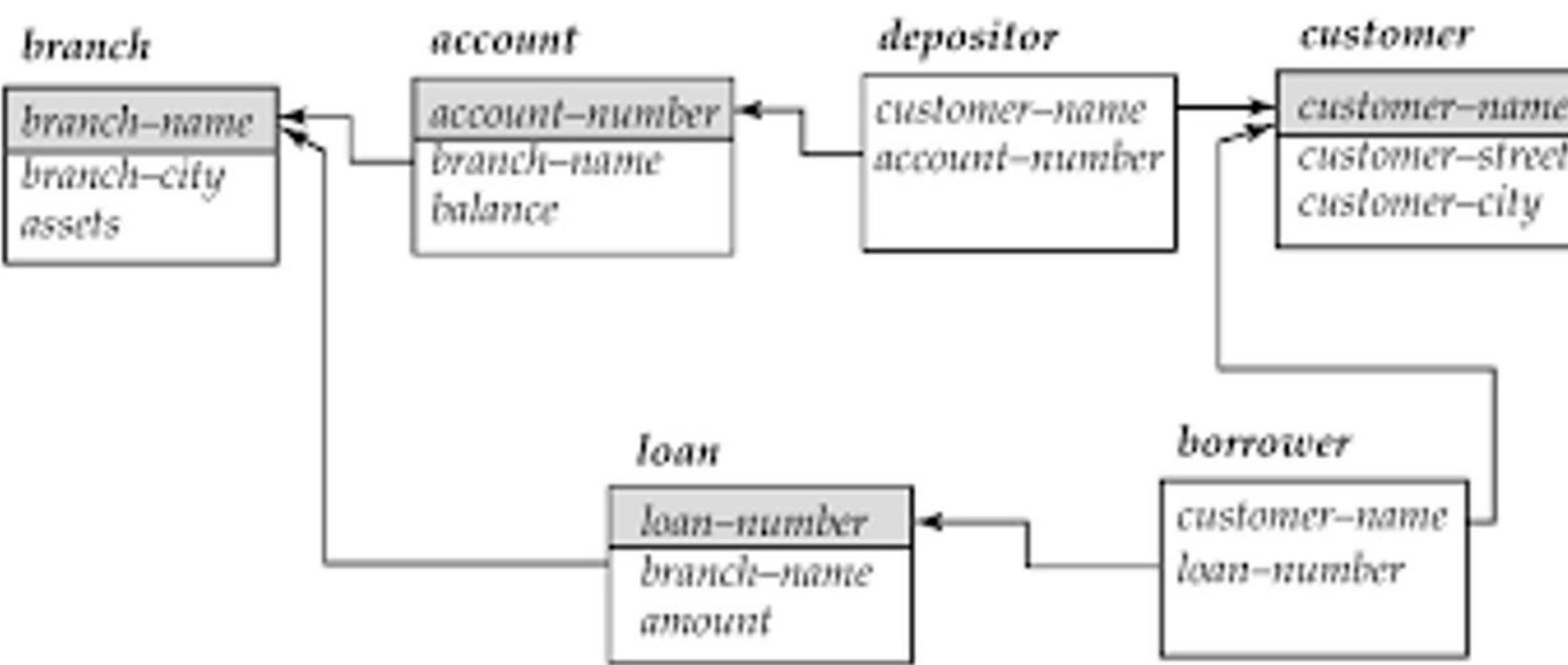
- The Cartesian-product operation, denoted by a cross (\times), allows us to combine information from any two relations.
 - It is a binary operator; we write the Cartesian product of relations R_1 and R_2 as $R_1 \times R_2$.
 - Cartesian-product operation associates every tuple of R_1 with every tuple of R_2 .
 - $R_1 \times R_2 = \{rs \mid r \in R_1 \text{ and } s \in R_2\}$, contains one tuple $\langle r, s \rangle$ (concatenation of tuples r and s) for each pair of tuples $r \in R_1, s \in R_2$.

R ₁	
A	B
1	P
2	Q
3	R

R ₂	
B	C
Q	X
R	Y
S	Z

- $R_1 \times R_2$ returns a relational instance whose schema contains all the fields of R_1 (in order as they appear in R_1) and all fields of R_2 (in order as they appear in R_2).
- If R_1 has m tuples and R_2 has n tuples the result will be having = $m * n$ tuples.
- Same attribute name may appear in both R_1 and R_2 , we need to devise a naming schema to distinguish between these attributes.

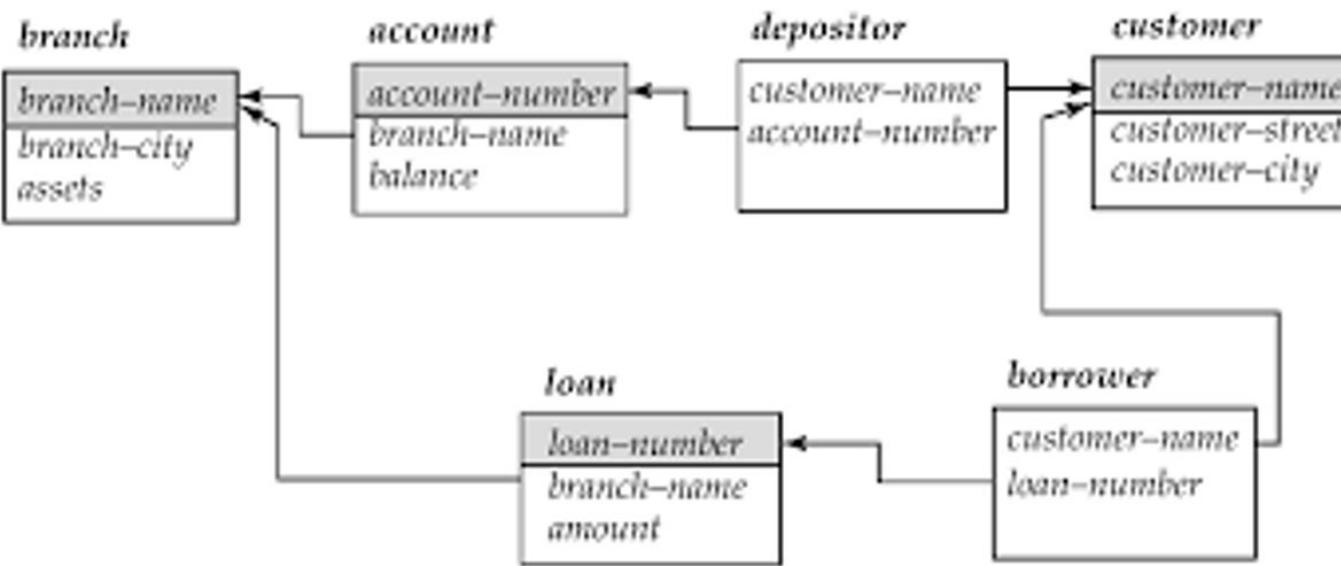
Q Write a RELATIONAL ALGEBRA query to find the name of all the customers along with account balance, who have an account in the bank?



Q Write a RELATIONAL ALGEBRA query to find the name of all the customers along with loan amount, who have a loan in the bank?

Q Write a RELATIONAL ALGEBRA query to find all loan_no along with amount and branch_name, which is situated in Delhi?

Q Write a RELATIONAL ALGEBRA query to find the name of the customer who have an account in the branch situated in Delhi and balance greater than 1000?



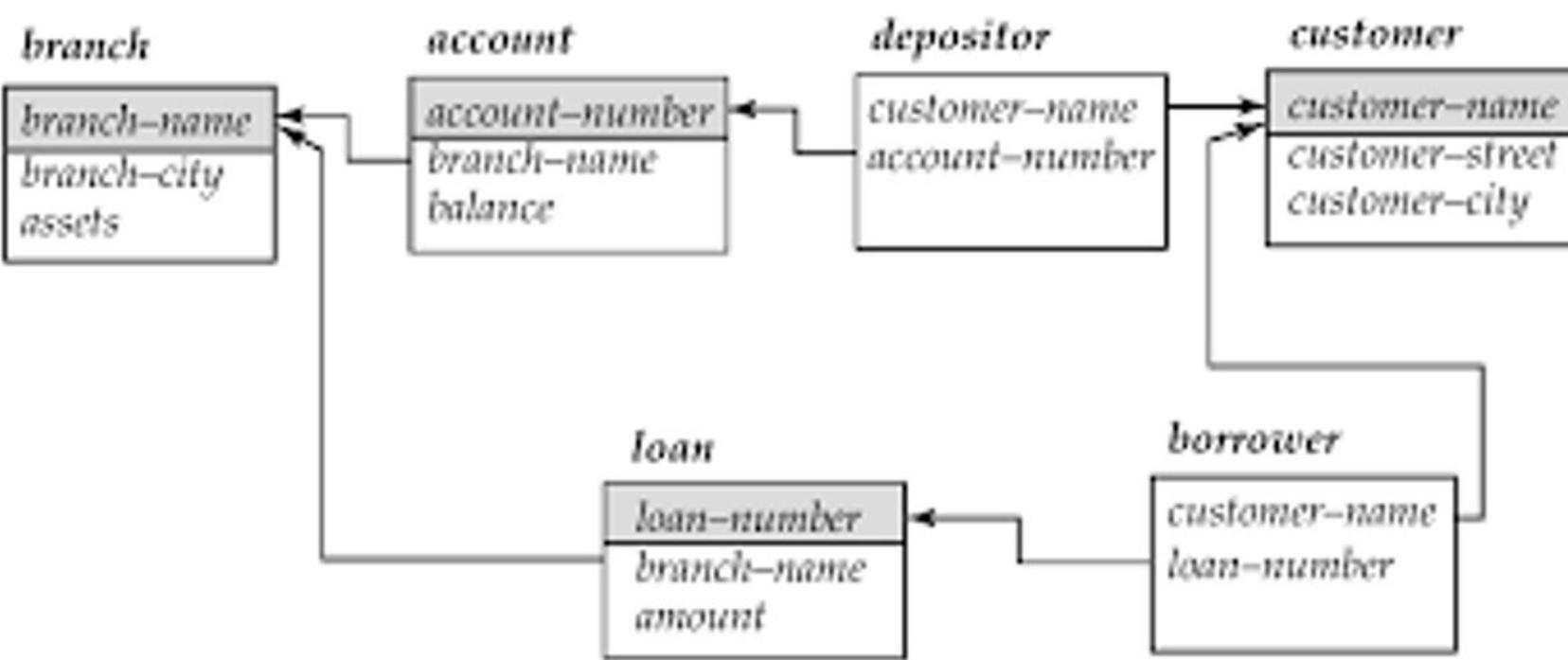
Important Point

- $\rho_{\text{Learner}}(\text{Student})$
 - This Query do not change the name of the table in the original data base, but create a new copy of the table student with name Learner

Additional/Derived Relational-Algebra Operations

- If we restrict ourselves to just the fundamental operations, certain common queries are lengthy to express. Therefore, we use additional operations.
- These additional operations do not add any power to the algebra.
- They are used to simplify the queries.

Q Write a RELATIONAL ALGEBRA query to find all the customer name who have both a loan and an account?



Set-Intersection Operation



- We will be using \cap symbol to denote set intersection.
- $r \cap s = r - (r - s)$
- Set intersection is not a fundamental operation and does not add any power to the relational algebra.
- $r \cap s = \{t \mid t \in r \text{ and } t \in s\}$
- $0 \leq |R \cap S| \leq \min(|R|, |S|)$

Q Let R1 (A, B, C) and R2 (D, E) be two relation schema, where the primary keys are shown underlined, and let C be a foreign key in R1 referring to R2. Suppose there is no violation of the above referential integrity constraint in the corresponding relation instances r1 and r2. Which one of the following relational algebra expressions would necessarily produce an empty relation? **(Gate-2004) (1 Marks)**

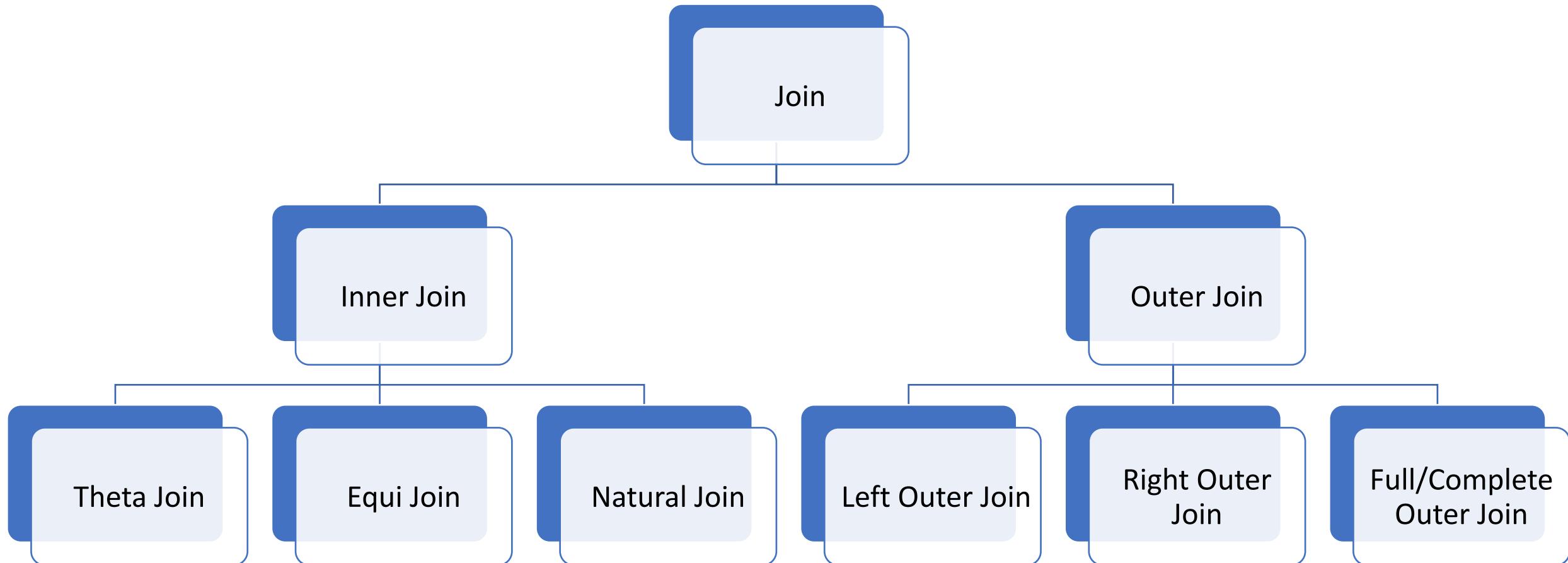
a) $\Pi_{\underline{D}}(r_2) - \Pi_{\underline{C}}(r_1)$

b) $\Pi_{\underline{C}}(r_1) - \Pi_{\underline{D}}(r_2)$

c) $\Pi_{\underline{D}}(r_1 \bowtie_{C \neq D} r_2)$

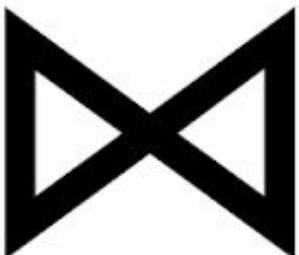
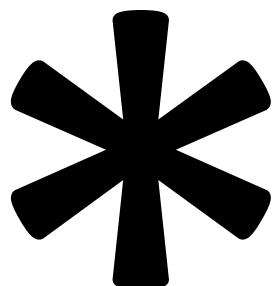
d) $\Pi_{\underline{C}}(r_1 \bowtie_{C=D} r_2)$

Join Operation



The Natural-Join Operation

- The natural join is a binary operation that allows us to combine certain selections and a Cartesian product into one operation.
- The natural join of r and s , denoted by $r \bowtie s$
- The standard definition of NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.



- The natural join of r and s is a relation on schema R \cup S formally defined as follows:

$$r \bowtie s = \prod_{R \cup S} (\sigma_{r.A_1=s.A_1 \wedge r.A_2=s.A_2 \wedge \dots \wedge r.A_n=s.A_n} (r \times s))$$

R₁	
A	B
1	P
2	Q
3	R

R₂	
B	C
Q	X
R	Y
S	Z

R₁ \bowtie R₂		

Notes

- The natural join is **associative** in nature.

$$(instructor \bowtie teaches) \bowtie course = instructor \bowtie (teaches \bowtie course)$$

- The natural join is a Lossy operator

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

Q Consider the following relations A, B and C:

A		
ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B		
ID	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C		
ID	Phone	Area
10	2200	02
99	2100	01

How many tuples does the result of the following relational algebra expression contain? Assume that the schema of $A \cup B$ is the same as that of A. **(Gate-2007) (2 Marks)**

$$(A \cup B) \bowtie_{A.Id > 40 \vee C.Id < 15} C$$

- a) 7 b) 4 c) 5 d) 9

Q.Consider the following two relations,R(A,B) and S(A,C): (Gate 2024,CS)(1 Mark) (NAT)

The total number of tuples obtained by evaluating the following expressions

$\sigma_{B < C} (R \bowtie_{R.A=S.A} S)$

is _____

R	
A	B
10	20
20	30
30	40
30	50
50	95

S	
A	C
10	90
30	45
40	80

Theta join

- The theta join/ Conditional join operation is a variant of the Cartesian product operation that allows us to combine a selection and a Cartesian product into a single operation.
- The theta join operation $r \bowtie_{\theta} s$ is defined as follows: $r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$

- The general form of a Theta Join operation on two relations $R(A_1, A_2, \dots, A_n)$ and $S(B_1, B_2, \dots, B_m)$ is

$$R \bowtie_{\theta} S$$

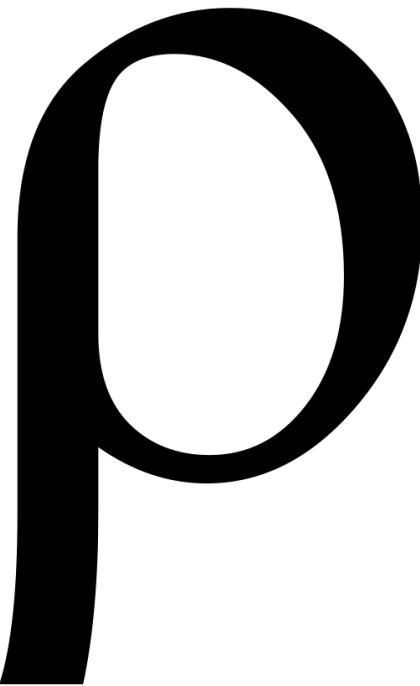
- The result of the JOIN is a relation Q with $n + m$ attributes $Q(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m)$ in that order; Q has one tuple for each combination of tuples—one from R and one from S —whenever the combination satisfies the join condition.

- A general join condition is of the form
 - $<\text{Condition}>\text{AND } <\text{Condition}>\text{ AND...AND } <\text{Condition}>$
 - where each $<\text{Condition}>$ is of the form $A_i \theta B_j$, A_i is an attribute of R, B_j is an attribute of S, A_i and B_j have the same domain, and θ (theta) is one of the comparison operators $\{=, <, \leq, \neq, >, \geq\}$.
- A JOIN operation with such a general join condition is called a THETA JOIN. Tuples whose join attributes are NULL or for which the join condition is FALSE do not appear in the result.

Equi Join

- The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is `=`, is called an EQUIJOIN.

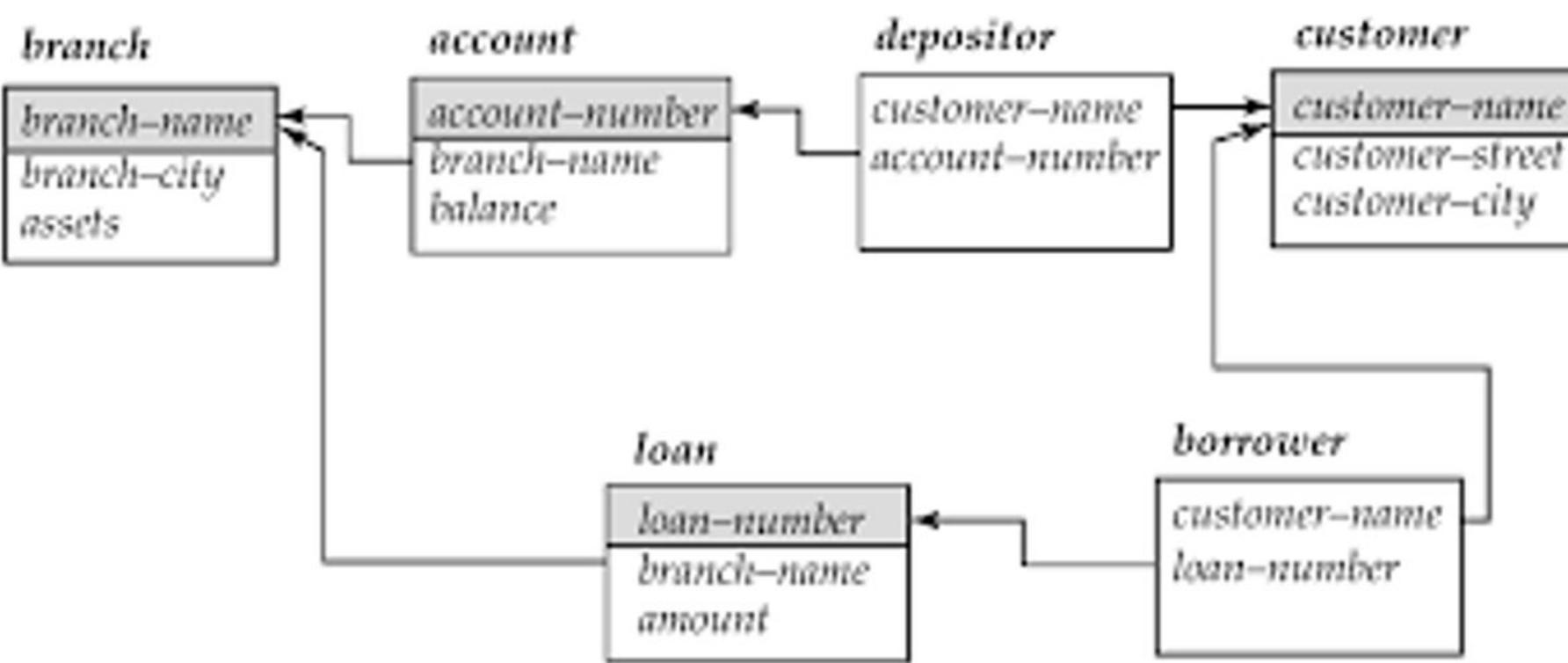
Rename Operation



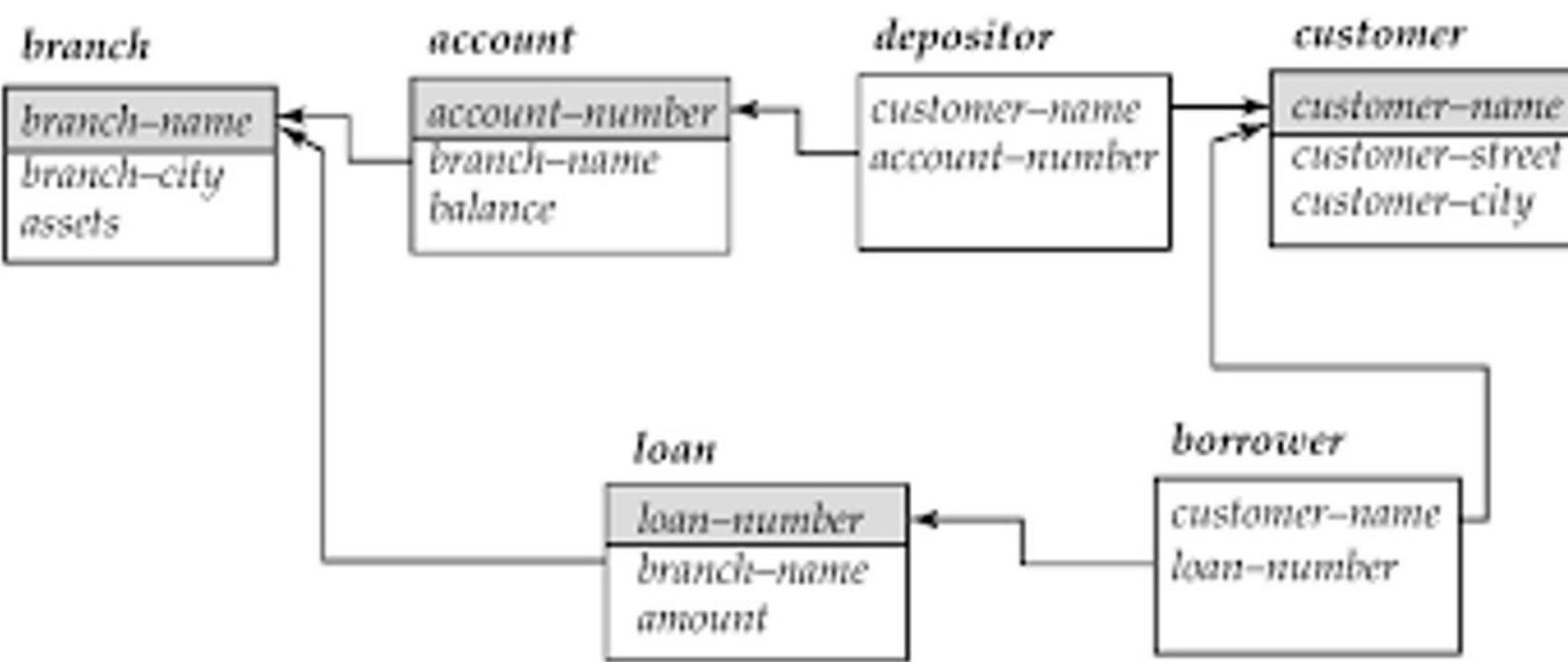
- The results of relational algebra are also relations but without any name.
- The rename operation allows us to rename the output relation. It is denoted with small Greek letter **rho** ρ . Where the result of expression **E** is saved with name of **x**.
- $\rho_{x(A_1, A_2, A_3, A_4, \dots, A_N)}(E)$
- $\rho_{\text{Learner}}(\text{Student})$
- $\rho_{\text{Learner}(Stu_ID, User_Name, Age)}(\text{Student}(Roll_No, Name, Age))$

- Deposit = (Branch-name, Account-number, Customer-name, Balance)
- $\rho_{\text{Plus_Cust}(\text{Name})}(\pi_{\text{customer-name}}(\sigma_{\text{balance} > 10000}(\text{Deposit}))$

Q Write a RELATIONAL ALGEBRA query to find the account_no along with balance with 8% interest as total amount, with table name as balance sheet?



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



Q The following functional dependencies hold for relations R(A, B, C) and S(B, D, E).

$B \rightarrow A$

$A \rightarrow C$

The relation R contains 200 tuples and the relation S contains 100 tuples. What is the maximum number of tuples possible in the natural join $R \bowtie S$? **(Gate-2010) (2 Marks)**

a) 100

b) 200

c) 300

d) 2000

Q Let R and S be two relations with the following schema (Gate-2008) (2 Marks)

R (P, Q, R1, R2, R3)

S (P, Q, S1, S2)

where {P, Q} is the key for both schemas. Which of the following queries are equivalent?

i) $\Pi_P(R \bowtie S)$

ii) $\Pi_P(R) \bowtie \Pi_P(S)$

iii) $\Pi_P(\Pi_{P,Q}(R) \cap \Pi_{P,Q}(S))$

iv) $\Pi_P(\Pi_{P,Q}(R) - (\Pi_{P,Q}(R) - \Pi_{P,Q}(S)))$

a) Only I and II

c) Only I, II and III

b) Only I and III

d) Only I, III and IV

Q Let r be a relation instance with schema $R = (A, B, C, D)$. We define $r_1 = \prod_{A, B, C}(r)$ and $r_2 = \prod_{A, D}(r)$. Let $s = r_1 * r_2$ where $*$ denotes natural join. Given that the decomposition of r into r_1 and r_2 is lossy, which one of the following is TRUE? (Gate-2005) (1 Marks)

- (A) $s \subset r$
- (B) $r \cup s = s$
- (C) $r \subset s$
- (D) $r * s = s$

Q Let r and s be two relations over the relation schemes R and S respectively, and let A be an attribute in R . The relational algebra expression $\sigma_{A=a}(r \bowtie s)$ is always equal to **(Gate-2001) (1 Marks)**

a) $\sigma_{A=a}(r)$

b) r

c) $\sigma_{A=a}(r) \bowtie s$

d) None of the above

Q Consider the join of a relation R with a relation S. If R has m tuples and S has n tuples, then the maximum and minimum sizes of the join respectively are:

(Gate-1999) (1 Marks)

(A) $m+n$ and 0

(B) mn and 0

(C) $m+n$ and $m-n$

(D) mn and $m+n$

Q. The relation schema, Person(pid,city), describes the city of residence for every person uniquely identified by pid. The following relational algebra operators are available: selection, projection, cross product, and rename. To find the list of cities where at least 3 persons reside, using the above operators, the minimum number of cross product operations that must be used is **(Gate 2024 CS) (2 Marks) (MSQ)**

(a) 1

(b) 2

(c) 3

(d) 4

Outer join Operations

- The outer-join operation is an extension of the join operation to deal with missing information.
- The outer join operation works in a manner similar to the natural join operation, but preserves those tuples that would be lost in a join by creating tuples in the result containing null values.
- We can use the outer-join operation to avoid this loss of information.

- There are actually three forms of the operation:
 - left outer join, denoted \bowtie
 - right outer join, denoted \ltimes
 - full outer join, denoted \bowtie^*

Left Outer Join

- The left outer join () takes all tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with null values for all other attributes from the right relation, and adds them to the result of the natural join.

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	B	C

Right Outer Join

- The left outer join (\bowtie) takes all tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with null values for all other attributes from the right relation, and adds them to the result of the natural join.

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

Full Outer Join

- The full outer join(\bowtie) does both the left and right outer join operations, padding tuples from the left relation that did not match any from the right relation, as well as tuples from the right relation that did not match any from the left relation, and adding them to the result of the join.

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

Q Consider the relations $r(A, B)$ and $s(B, C)$, where $s.B$ is a primary key and $r.B$ is a foreign key referencing $s.B$. Consider the query

Q: $r \bowtie (\sigma_{B < 5}(s))$

Let LOJ denote the natural left outer-join operation. Assume that r and s contain no null values.

Which of the following is NOT equivalent to Q? **(Gate-2018) (2 Marks)**

a) $\sigma_{B < 5}(r \bowtie s)$

b) $\sigma_{B < 5}(r \text{ LOJ } s)$

c) $r \text{ LOJ } (\sigma_{B < 5}(s))$

d) $\sigma_{B < 5}(r) \text{ LOJ } s$

Q Consider two relations $R_1(A, B)$ with the tuples $(1, 5), (3, 7)$ and $R_2(A, C) = (1, 7), (4, 9)$. Assume that $R(A, B, C)$ is the full natural outer join of R_1 and R_2 . Consider the following tuples of the form (A, B, C)

$a = (1, 5, \text{null}),$

$b = (1, \text{null}, 7),$

$c = (3, \text{null}, 9),$

$d = (4, 7, \text{null}),$

$e = (1, 5, 7),$

$f = (3, 7, \text{null}),$

$g = (4, \text{null}, 9).$

Which one of the following statements is correct? **(Gate-2015) (1 Marks)**

(A) R contains a, b, e, f, g but not c, d

(B) R contains a, b, c, d, e, f, g

(C) R contains e, f, g but not a, b

(D) R contains e but not f, g

DIVISION

- In general, the DIVISION operation is applied to two relations $R(Z) \div S(X)$, where the attributes of R are a subset of the attributes of S; that is, $X \subseteq Z$.
- $Z=\{A,B\}$
- $X=\{A\}$
- $Y = Z-X = \{B\}$
- This means that, for a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with every tuple in S.
- Note that in the formulation of the DIVISION operation, the tuples in the denominator relation S restrict the numerator relation R by selecting those tuples in the result that match all values present in the denominator.

R	A	B
a1	b1	
a2	b1	
a3	b1	
a4	b1	
a1	b2	
a3	b2	
a2	b3	
a3	b3	
a4	b3	
a1	b4	
a2	b4	
a3	b4	

S	A
a1	
a2	
a3	
a4	

T	B
a1	b1
a2	b1
a3	b4

- $R \div S = \{ t[a_1, \dots, a_n] : t \in R \wedge \forall s \in S ((t[a_1, \dots, a_n] \cup s) \in R) \}$
- where $\{a_1, \dots, a_n\}$ is the set of attribute names unique to R and $t[a_1, \dots, a_n]$ is the restriction of t to this set. It is usually required that the attribute names in the header of S are a subset of those of R because otherwise the result of the operation will always be empty.

Completed	
Student	Task
Fred	Database1
Fred	Database2
Fred	Compiler1
Eugene	Database1
Eugene	Compiler1
Sarah	Database1
Sarah	Database2

DBProject	
Task	
Database1	
Database2	

Completed	
÷	
DBProject	
Student	
Fred	
Sarah	

$$\pi_{\text{Student}}(R) = \{\pi_{\text{Student}}[(\pi_{\text{Student}}(R) \times S) - \pi_{\text{Student,Task}}(R)]\}$$

Completed	
Student	Task
Fred	Database1
Fred	Database2
Fred	Compiler1
Eugene	Database1
Eugene	Compiler1
Sarah	Database1
Sarah	Database2

DBProject	
Task	
Database1	
Database2	

Completed	
÷	
DBProject	
Student	
Fred	
Sarah	

$$\begin{aligned}
 T &:= \pi_{\text{Student}}(R) \times S \\
 U &:= T - \pi_{\text{Student,Task}}(R) \\
 V &:= \pi_{\text{Student}}(U) \\
 W &:= \pi_{\text{Student}}(R) - V
 \end{aligned}$$

Q Consider the given table R and S find the number of elements retrieved by the query

Table R

A	B
1	Dog
1	Cat
1	Cow
2	Cat
4	Cat
3	Dog
4	Dog
2	Dog
4	Cow

Table S

B
Cat
Dog

$$\Pi_{A,B}(R) \div \Pi_B(S) \underline{\hspace{100pt}}$$

Q Consider the following three relations in a relational database.

Employee(eId, Name), Brand(bId, bName), Own(eId, bId)

Which of the following relational algebra expressions return the set of eIds who own all the brands? **(GATE 2022) (2 MARKS)**

(A) $\Pi_{eId}(\Pi_{eId,bId}(Own) / \Pi_{bId}(Brand))$

(B) $\Pi_{eId}(Own) - \Pi_{eId}((\Pi_{eId}(Own) \times \Pi_{bId}(Brand)) - \Pi_{eId,bId}(Own))$

(C) $\Pi_{eId}(\Pi_{eId,bId}(Own) / \Pi_{bId}(Own))$

(D) $\Pi_{eId}((\Pi_{eId}(Own) \times \Pi_{bId}(Own)) / \Pi_{bId}(Brand))$

Q Consider a database that has the relation schema CR (StudentName, CourseName). An instance of the schema CR is as given below.

The following query is made on the database.

$$\begin{aligned} T_1 &\leftarrow \pi_{\text{CourseName}} (\sigma_{\text{StudentName}=\text{SA}}(\text{CR})) \\ T_2 &\leftarrow \text{CR} \div T_1 \end{aligned}$$

StudentName	CourseName
SA	CA
SA	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
SF	CB
SF	CC

The number of rows in T_2 is _____ . (Gate-2017) (1 Marks)

Q A relation $r(A,B)$ in a relational database has 1200 tuples. The attribute A has integer values ranging from 6 to 20, and the attribute B has integer values ranging from 1 to 20. Assume that the attributes A and B are independently distributed. The estimated number of tuples in the output of $\sigma_{(A>10) \vee (B=18)}(r)$ is _____. **(GATE 2021)**

(a) 820

(b) 1200

(c) 960

(d) 1000

Q Consider the following relations P(X,Y,Z), Q(X,Y,T) and R(Y,V). How many tuples will be returned by the following relational algebra query? (Gate-2019) (2 Marks)

$$\prod_x (\sigma_{(P.Y=R.Y \wedge R.V=V2)}(P \times R)) - \prod_x (\sigma_{(Q.Y=R.Y \wedge Q.T>2)}(Q \times R))$$

P		
X	Y	Z
X1	Y1	Z1
X1	Y1	Z2
X2	Y2	Z2
X2	Y4	Z4

Q		
X	Y	T
X2	Y1	2
X1	Y2	5
X1	Y1	6
X3	Y3	1

R	
Y	V
Y1	V1
Y3	V2
Y2	V3
Y2	V2

Q Consider the relational schema given below, where *eId* of the relation *dependent* is a foreign key referring to *emplId* of the relation *employee*. Assume that every employee has at least one associated dependent in the dependent relation. **(Gate-2014) (2 Marks)**

employee (*emplId*, *empName*, *empAge*)

dependent(*depId*, *eId*, *depName*, *depAge*)

Consider the following relational algebra query:

$\Pi_{\text{emplId}}(\text{employee}) - \Pi_{\text{emplId}}(\text{employee} \bowtie_{(\text{emplId} = \text{eID}) \wedge (\text{empAge} \leq \text{depAge})} \text{dependent})$

The above query evaluates to the set of *emplIds* of employees whose age is greater than that of

- (A) some dependent.
- (B) all dependents.
- (C) some of his/her dependents
- (D) all of his/her dependents.

Q Consider a join (relation algebra) between relations $r(R)$ and $s(S)$ using the nested loop method. There are 3 buffers each of size equal to disk block size, out of which one buffer is reserved for intermediate results. Assuming $\text{size}(r(R)) < \text{size}(s(S))$, the join will have fewer number of disk block accesses if **(Gate-2014) (2 Marks)**

- a)** relation $r(R)$ is in the outer loop.
- b)** relation $s(S)$ is in the outer loop.
- c)** join selection factor between $r(R)$ and $s(S)$ is more than 0.5.
- d)** join selection factor between $r(R)$ and $s(S)$ is less than 0.5.

Q Consider a relational table r with sufficient number of records, having attributes $A_1, A_2 \dots, A_n$ and let $1 \leq p \leq n$. Two queries Q_1 and Q_2 are given below. **(Gate-2011) (2 Marks)**

$Q_1: \Pi_{A_1, \dots, A_p} (\sigma_{A_p=c} (r))$ where c is a constant

$Q_2: \Pi_{A_1, \dots, A_p} (\sigma_{c_1 \leq A_p \leq c_2} (r))$ where c_1 and c_2 are constants.

The database can be configured to do ordered indexing on A_p or hashing on A_p . Which of the following statements is TRUE?

- a) Ordered indexing will always outperform hashing for both queries
- b) Hashing will always outperform ordered indexing for both queries
- c) Hashing will outperform ordered indexing on Q_1 , but not on Q_2
- d) Hashing will outperform ordered indexing on Q_2 , but not on Q_1

Q Information about a collection of students is given by the relation $\text{studInfo}(\text{studId}, \text{name}, \text{sex})$. The relation $\text{enroll}(\text{studId}, \text{courseId})$ gives which student has enrolled for (or taken) that course(s). Assume that every course is taken by at least one male and at least one female student. What does the following relational algebra expression represent? **(Gate-2007) (2 Marks)**

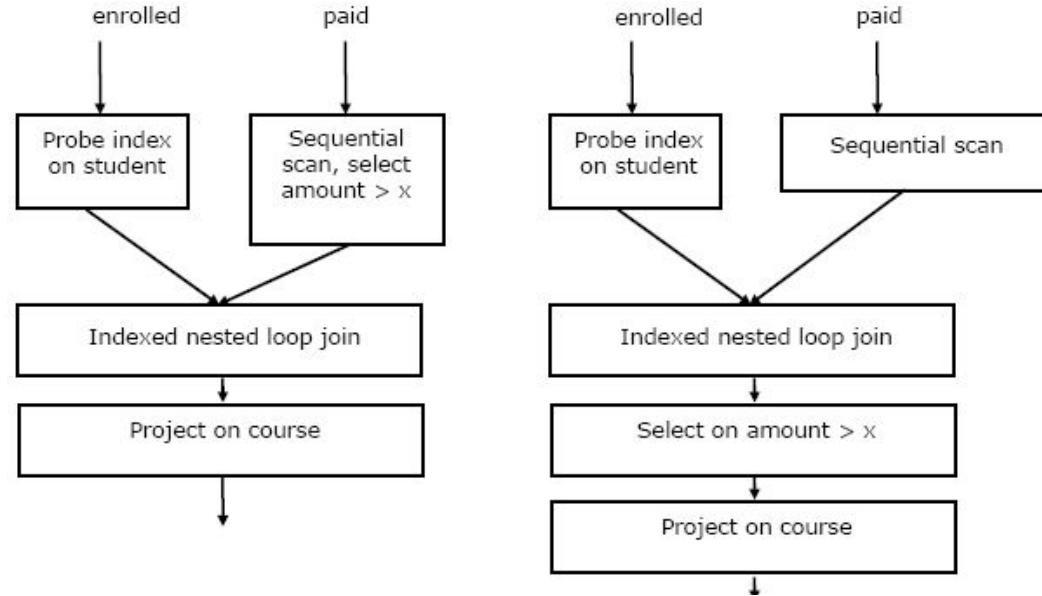
$$\pi_{\text{courseId}}((\pi_{\text{studId}}(\sigma_{\text{sex}=\text{"female"}}(\text{studInfo})) \times \pi_{\text{courseId}}(\text{enroll})) - \text{enroll})$$

- (A) Courses in which all the female students are enrolled.
- (B) Courses in which a proper subset of female students are enrolled.
- (C) Courses in which only male students are enrolled.
- (D) None of the above

Q Consider a selection of the form $\sigma_{A \leq 100}(r)$, where r is a relation with 1000 tuples. Assume that the attribute values for A among the tuples are uniformly distributed in the interval [0,500]. Which one of the following options is the best estimate of the number of tuples returned by the given selection query? **(Gate-2007) (2 Marks)**

- a) 50
- b) 100
- c) 15
- d) 200

Q Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount), where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Assume that amounts 6000, 7000, 8000, 9000 and 10000 were each paid by 20% of the students. Consider these query plans (Plan 1 on left, Plan 2 on right) to “list all courses taken by students who have paid more than x”. **(Gate-2006) (2 Marks)**



A disk seek takes 4ms, disk data transfer bandwidth is 300 MB/s and checking a tuple to see if amount is greater than x takes 10 micro-seconds. Which of the following statements is correct?

- (A)** Plan 1 and Plan 2 will not output identical row sets for all databases.
- (B)** A course may be listed more than once in the output of Plan 1 for some databases
- (C)** For $x = 5000$, Plan 1 executes faster than Plan 2 for all databases.
- (D)** For $x = 9000$, Plan 1 executes slower than Plan 2 for all databases.

Q Consider the relation Student (name, sex, marks), where the primary key is shown underlined, pertaining to students in a class that has at least one boy and one girl. What does the following relational algebra expression produce? (Note: ρ is the rename operator). **(Gate-2004) (2 Marks)**

$$\pi_{\text{name}} \{ \sigma_{\text{sex=female}} (\text{Student}) \} - \pi_{\text{name}} (\text{Student} \bowtie_{(\text{sex=female} \wedge \text{x=male} \wedge \text{marks} \leq m)} \rho_{n,x,m} (\text{Student}))$$

- a) names of girl students with the highest marks
- b) names of girl students with more marks than some boy student
- c) names of girl students with marks not less than some boy student
- d) names of girl students with more marks than all the boy students

Q Given the relations (Gate-2000) (2 Marks)

employee (name, salary, dept-no), and

department (dept-no, dept-name, address),

Which of the following queries cannot be expressed using the basic relational algebra operations (σ , π , \times , \bowtie , U , \cap , $-$)?

- a) Department address of every employee**

- b) Employees whose name is the same as their department name**

- c) The sum of all employees' salaries**

- d) All employees of a given department**

Q The following relation records the age of 500 employees of a company, where empNo (indicating the employee number) is the key:

empAge(empNo, age)

Consider the following relational algebra expression:

$\prod_{empNo}(empAge \bowtie_{(age > age1)} \rho_{empNo1, age1}(empAge))$

What does the above expression generate? **(GATE 2021) (2 MARKS)**

- (A)** Employee numbers of only those employees whose age is the maximum
- (B)** Employee numbers of only those employees whose age is more than the age of exactly one other employee
- (C)** Employee numbers of all employees whose age is not the minimum
- (D)** Employee numbers of all employees whose age is the minimum

Q Suppose the adjacency relation of vertices in a graph is represented in a table Adj (X, Y). Which of the following queries cannot be expressed by a relational algebra expression of constant length? **(Gate-2001) (1 Marks)**

- (A) List of all vertices adjacent to a given vertex
- (B) List all vertices which have self-loops
- (C) List all vertices which belong to cycles of less than three vertices
- (D) List all vertices reachable from a given vertex

Introduction to SQL

- There are a number of database query languages in use, either commercially or experimentally, around 50 are used popularly. We will study the most widely used query language ‘SQL’.
- Structured Query Language is a domain-specific language (not general purpose) used in programming and design for managing data held in a relational database management system (RDBMS).
- Although we refer to the SQL language as a “query language,” it can do much more than just query a database. It can define the structure of the data base, modify data in the database, specify security constraints and number of other tasks.
- Originally based upon relational algebra(procedural) and tuple relational calculus (Non-procedural) mathematical model.

Overview of the SQL Query Language

1. IBM developed the original version of SQL, originally called Sequel (*Structured English Query Language*), as part of the System R project in the early 1970s.
2. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query Language) (some other company has trademark on the word sequel). SQL has clearly established itself as *the* standard relational database language.
3. In 1986, the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published an SQL standard, called SQL-86.
4. The next version of the standard was SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL: 2016 and most recently SQL:2019.

Parts of SQL

- **Data-definition language (DDL).** The SQL DDL provides commands for defining relation schemas, deleting relations, and modifying relation schemas.
 - **Integrity.** The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints are disallowed. e.g. not null which means Primary key value cannot be null.
 - **View definition.** The SQL DDL includes commands for defining views. e.g. sorting result in ascending or descending order with order by clause.
 - **Authorization.** The SQL DDL includes commands for specifying access rights to relations and views like read only, read/write. E.g. grant etc.

- **Data-manipulation language (DML).** The SQL DML provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database. e.g. insert, delete command etc.
- **Transaction control.** SQL includes commands for specifying the beginning and ending of transactions. E.g. commit, rollback, savepoint etc.
- **Embedded SQL** and **dynamic SQL**. Embedded and dynamic SQL define how SQL statements can be embedded within general-purpose programming languages, such as C, C++, and Java.

Q Which of the following is/are correct? (GATE-1999) (1 Marks)

- (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**

Basic Structure of SQL Queries

- For any SQL as query, input and output both are relations. Number of relations inputs to a query will be at least one, but output will always be a single relation without any name unless specified, but columns will have names from input tables.
- The basic structure of an SQL query consists of three clauses: select, from, and where. The query takes its input the relations listed in the from clause, operates on them as specified in the where and select clauses, and then produces a relation as the result without any name unless specified.
- A typical SQL query has the form.

Select A_1, A_2, \dots, A_n
from r_1, r_2, \dots, r_m
Where P;

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

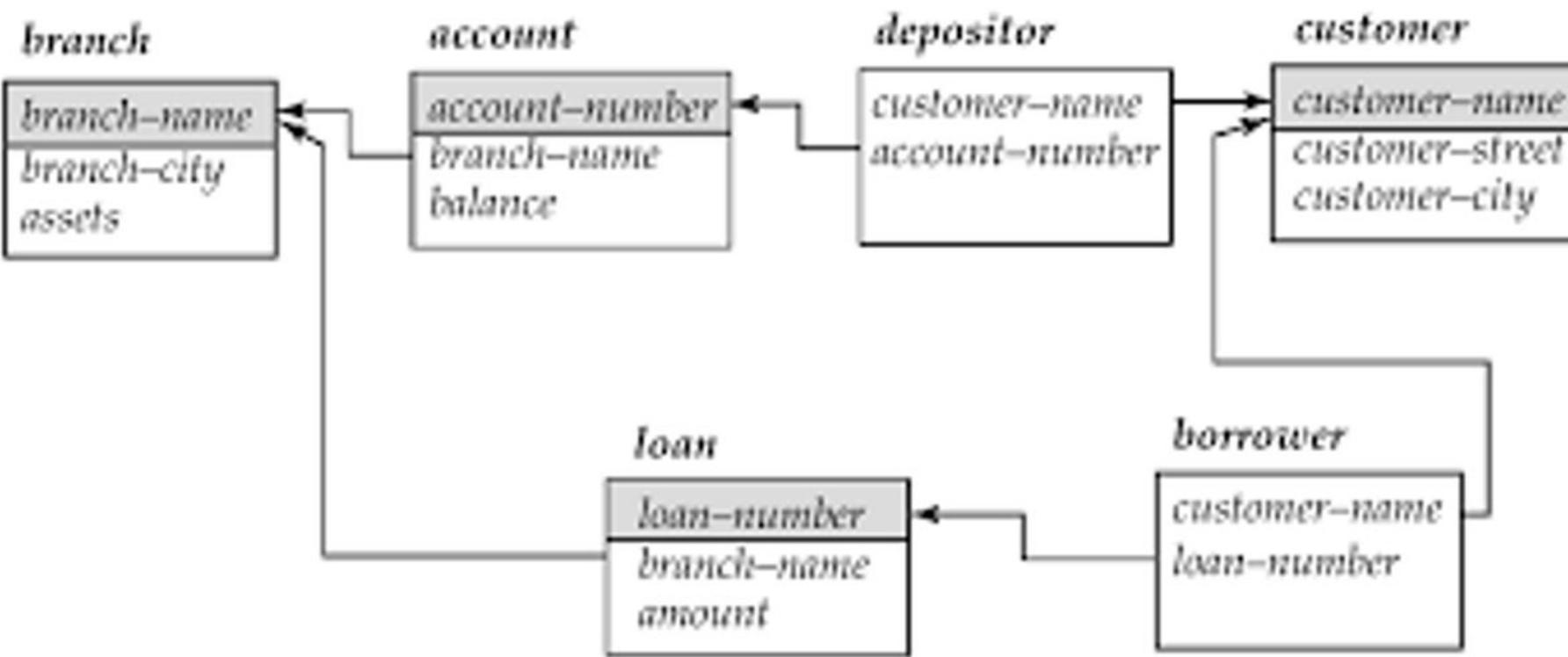
Select A ₁ , A ₂ ,..., A _n	(Column name)
from r ₁ , r ₂ ,..., r _m	(Relation/table name)
Where P;	(Condition)

- It is to be noted only select and from are mandatory clauses, and if not required then it is not essential to write where. If the **where** clause is omitted, the predicate *P* is **true**.
- SQL in general is not case sensitive i.e. it doesn't matter whether we write query in upper or lower case.
- In the formal, mathematical definition of the relational model, a relation is a set. Thus, duplicate tuples would never appear in relations.
- In practice, duplicate elimination is time-consuming. Therefore, SQL allows duplicates in relations as well as in the results of SQL expressions. In those cases where we want to force the elimination of duplicates, we insert the keyword **distinct** after **select**, will discuss in detail later.
- SQL allows us to use the keyword **all** to specify explicitly that duplicates are not removed, Since duplicate retention is the default, we shall not use **all** in our examples

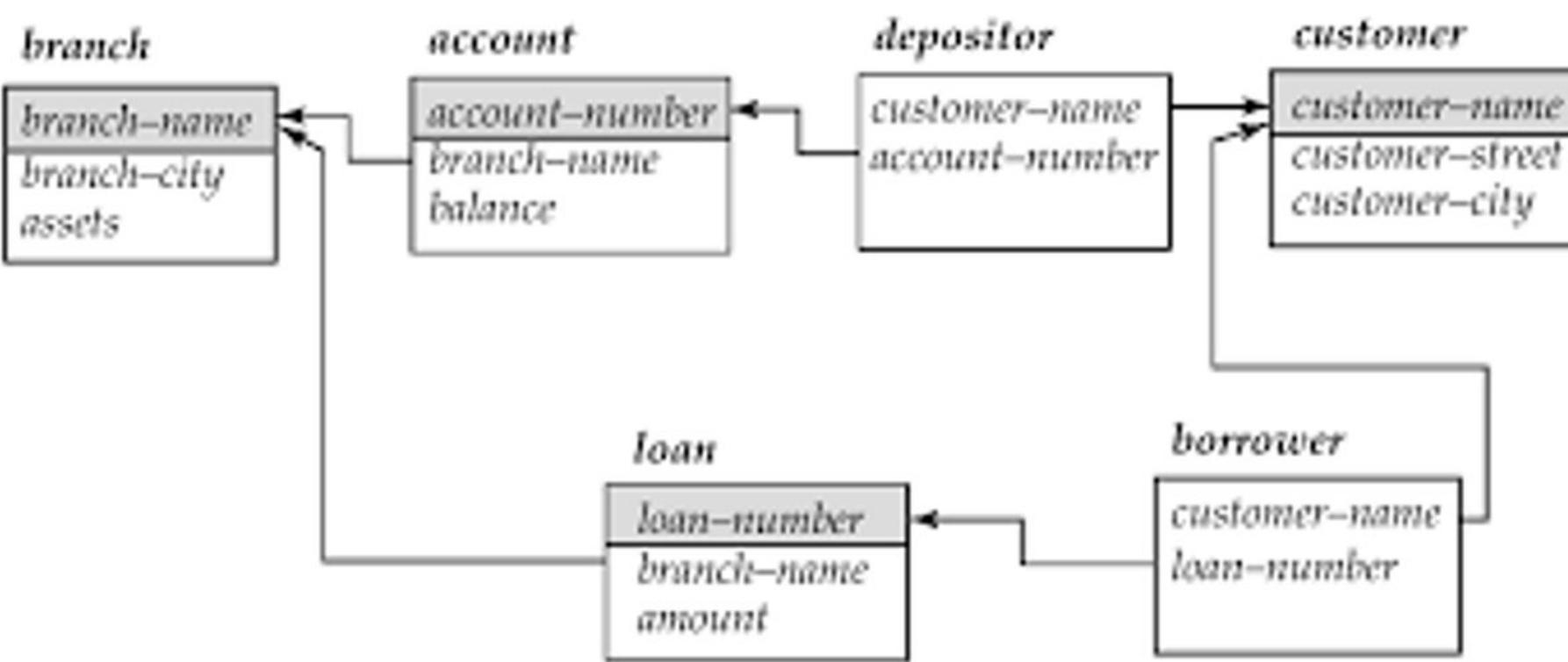
Select Clause

- The function of Select clause in SQL is more or less same as that of ' Π ' projection in the relational algebra. It is used to pick the column required in result of the query out of all the columns in relation/table. (Vertical filtering)
 - **Select A₁, A₂,..., A_n (Column name)**
- The order in which columns are represented in the select clause, will be same the column will appear in the relation.
- It is to be noted that in Relational Algebra ' Π ' projection is not mandatory and should be used only when we require less than all the column available in the table, but in SQL even if we need all the column we must use Select, the argument is it makes SQL query more readable.
- But to make things easy we can use '*' to specify that we need all columns
Select *
- The **select** clause may also contain arithmetic expressions involving the operators +, -, / and * operating on constants or attributes of tuples. however, that it does not result in any change to the relation/table.

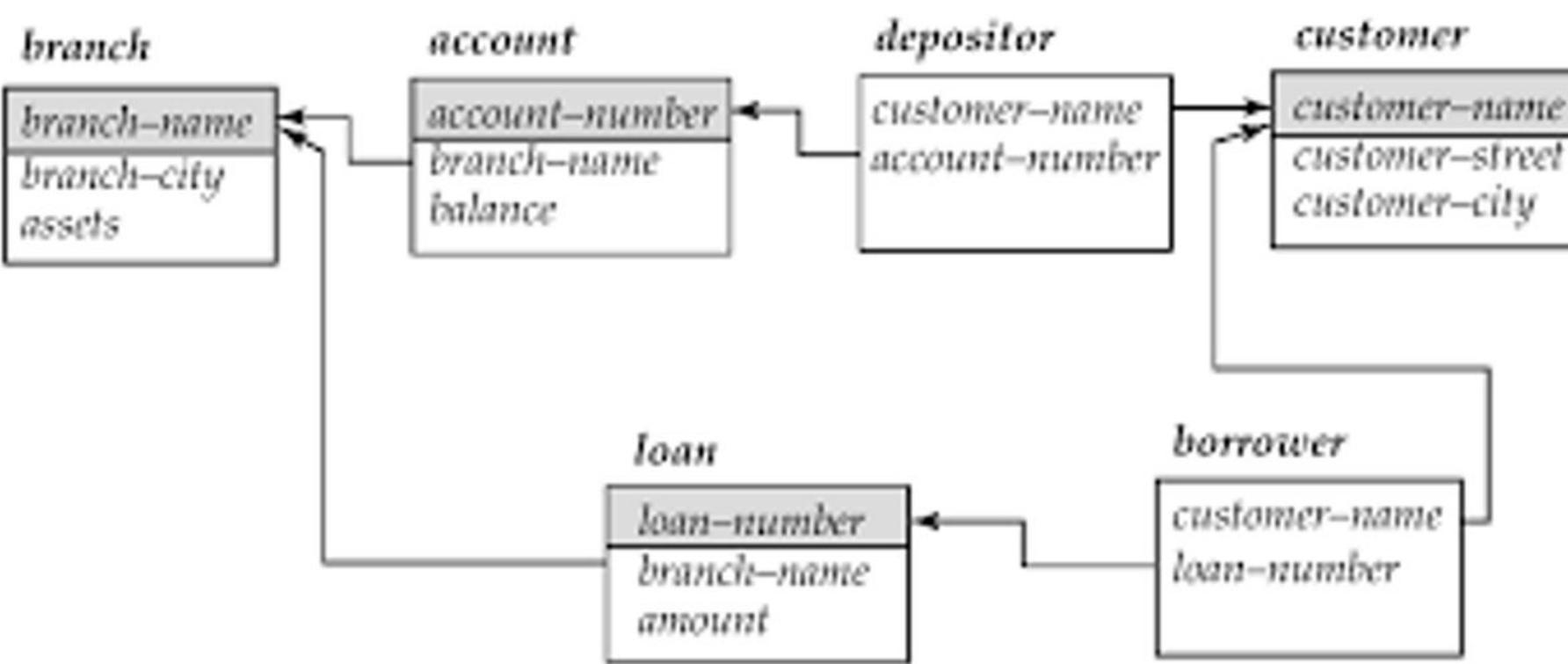
Q Write a SQL query to find all the details of bank branches?



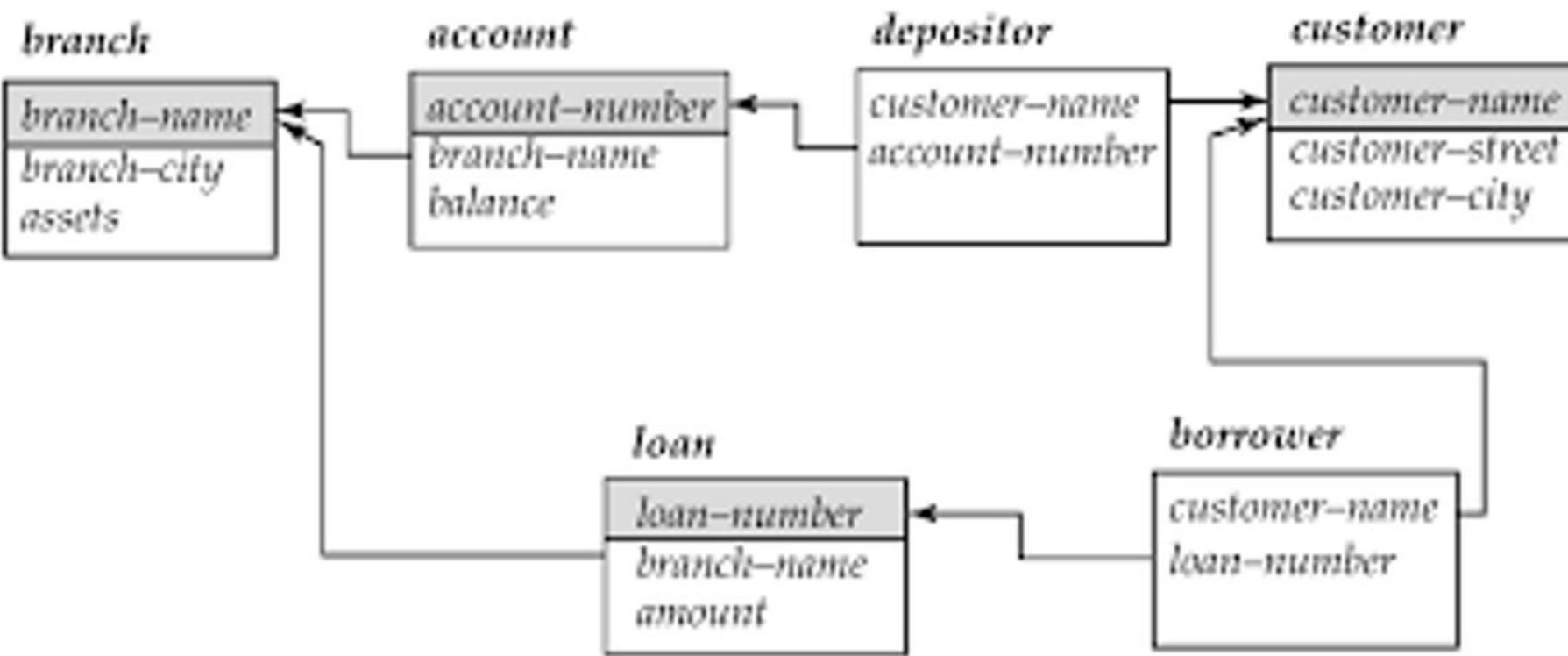
Q Write a SQL query to find each loan number along with loan amount?



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?



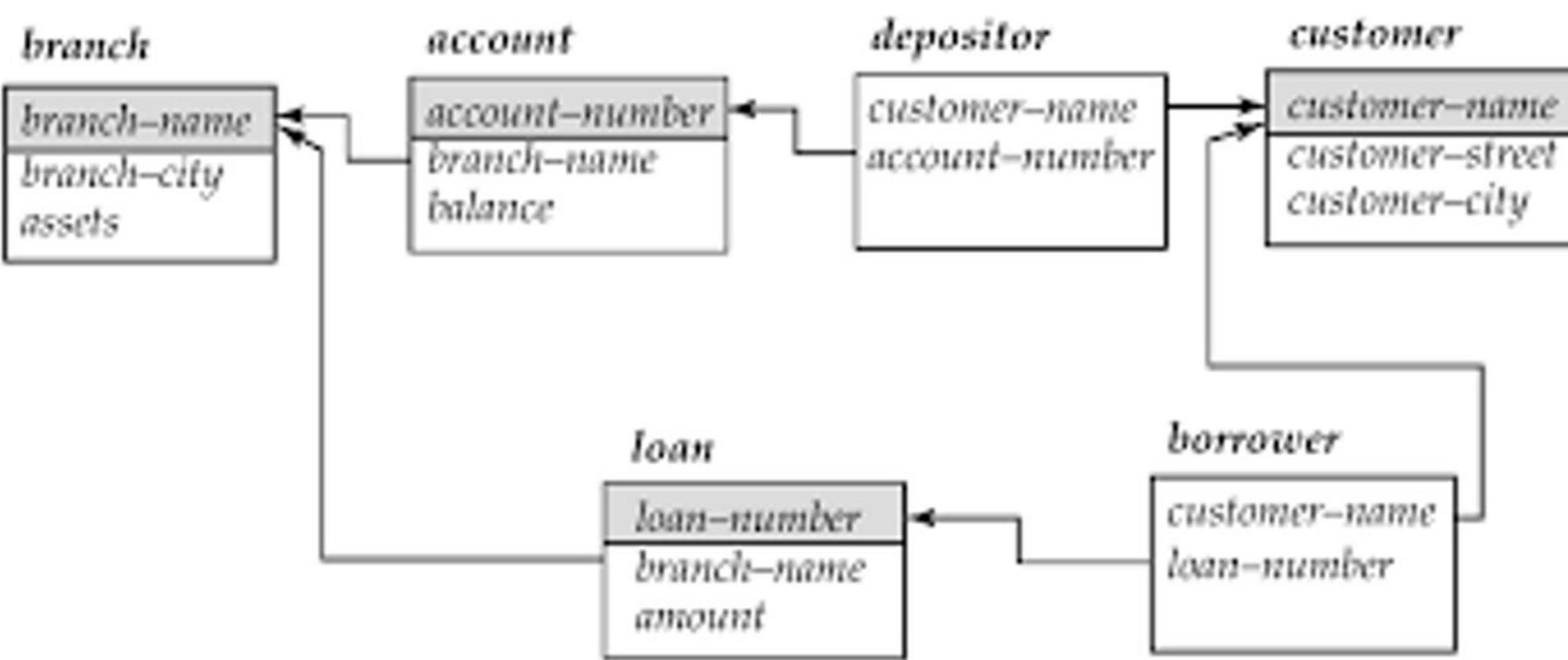
Q Select operation in SQL is equivalent to **(Gate-2015) (1 Marks)**

- (A)** the selection operation in relational algebra
- (B)** the selection operation in relational algebra, except that select in SQL retains duplicates
- (C)** the projection operation in relational algebra, except that select in SQL retains duplicates
- (D)** the projection operation in relational algebra

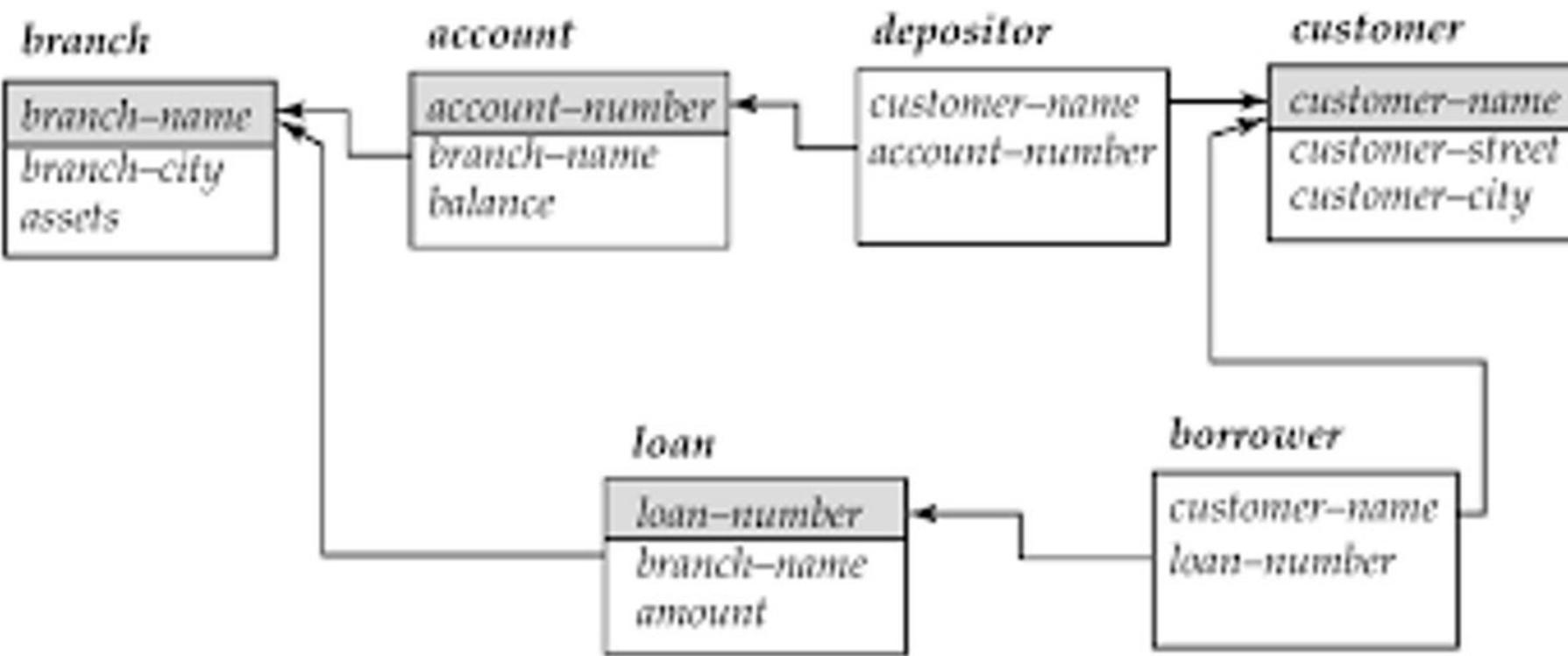
Select Clause with where clause

1. Where clause in SQL is same as ‘ σ ’ sigma of relational algebra where we specify the conditions/Predicate (horizontal filtering)
2. 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.
3. SQL allows the use of the logical connectives **and**, **or**, and **not** in the **where** clause.
4. 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.
5. Similarly, we can use the **not between** comparison operator.

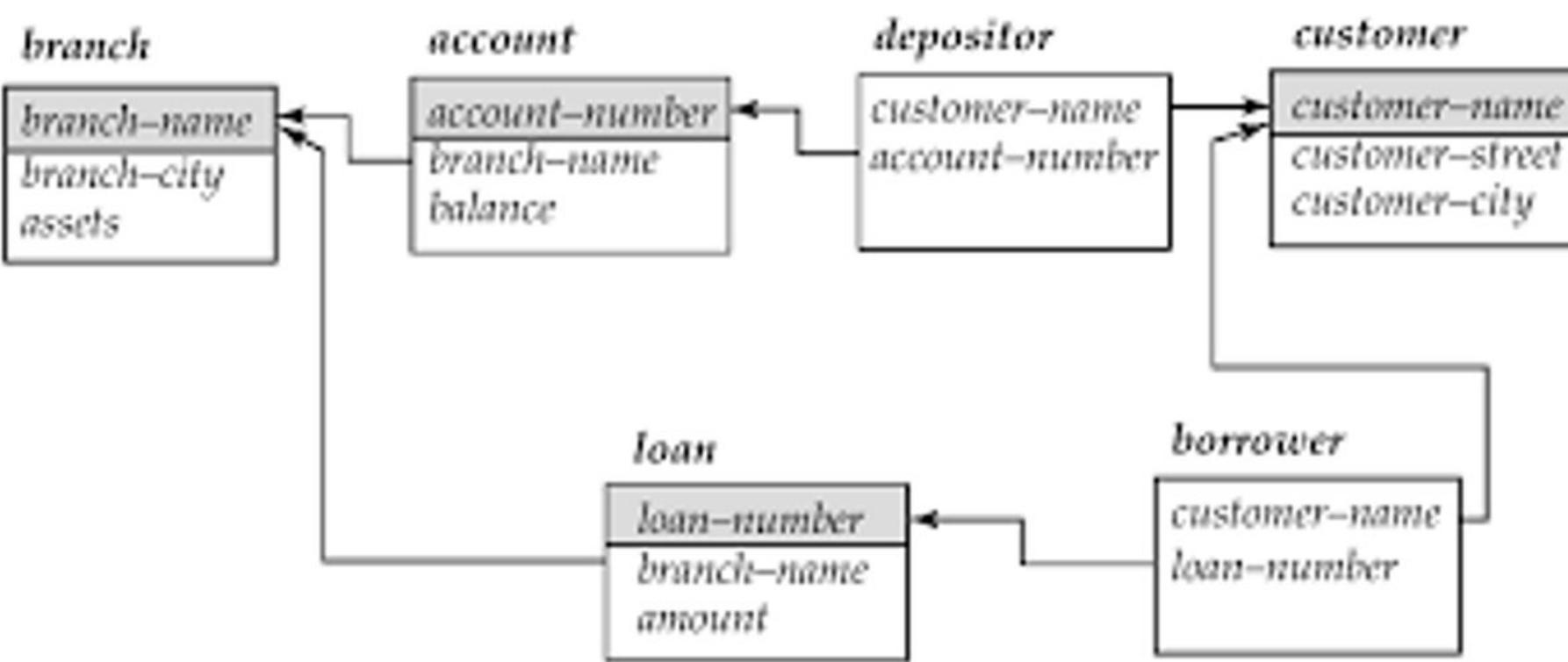
Q Write a SQL query to find all account_no where balance is less than 1000?



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 a database table T containing two columns X and Y each of type integer. After the creation of the table, one record ($X=1$, $Y=1$) is inserted in the table. Let MX and MY denote the respective maximum values of X and Y among all records in the table at any point in time. Using MX and MY , new records are inserted in the table 128 times with X and Y values being $MX+1$, $2*MY+1$ respectively. It may be noted that each time after the insertion, values of MX and MY change. What will be the output of the following SQL query after the steps mentioned above are carried out? **(Gate-2011) (2 Marks)**

SELECT Y FROM T WHERE X=7;

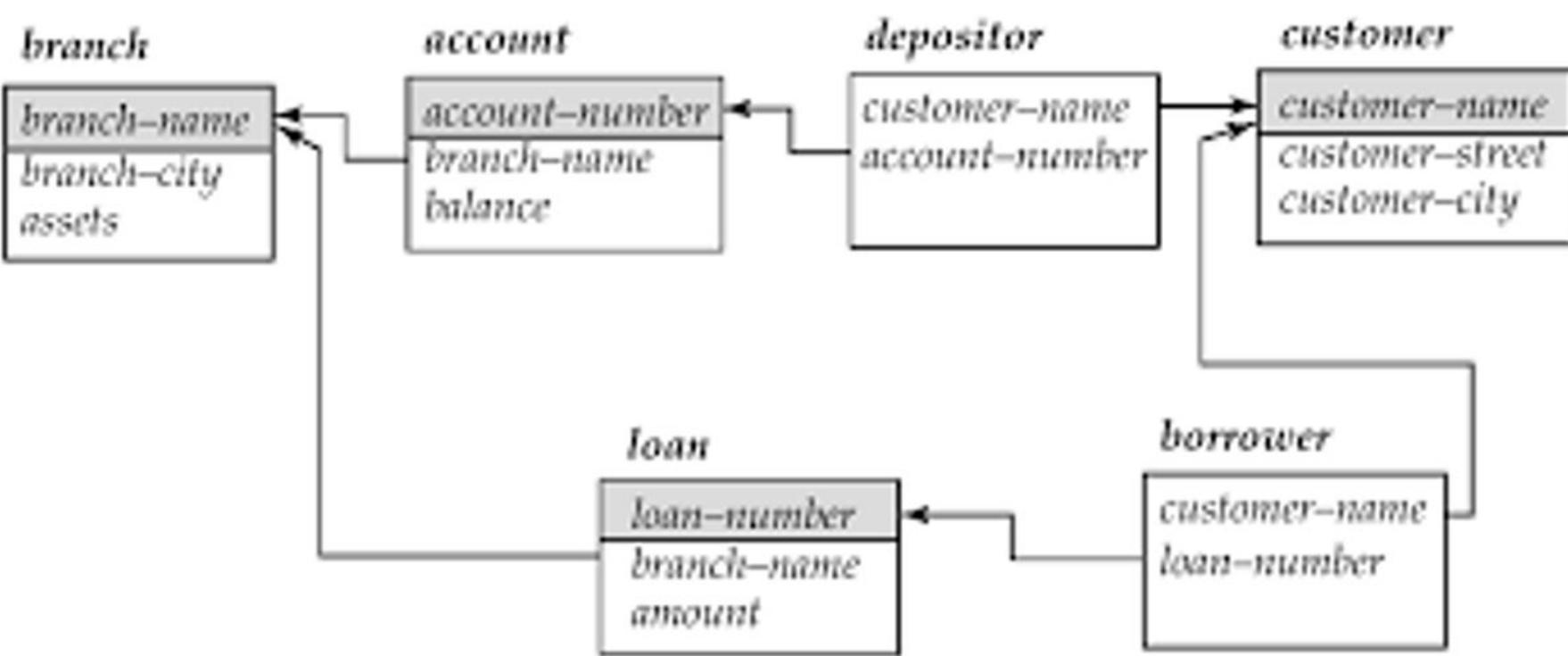
- (A) 127 (B) 255 (C) 129 (D) 257

Set Operation

1. The SQL operations **union**, **intersect**, and **except/minus** operate on relations and corresponds to the mathematical set-theory operations \cup , \cap and $-$ respectively.
2. 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**.
3. The **intersect** operation automatically eliminates duplicates. If we want to retain all duplicates, we must write **intersect all** in place of **intersect**.
4. 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?



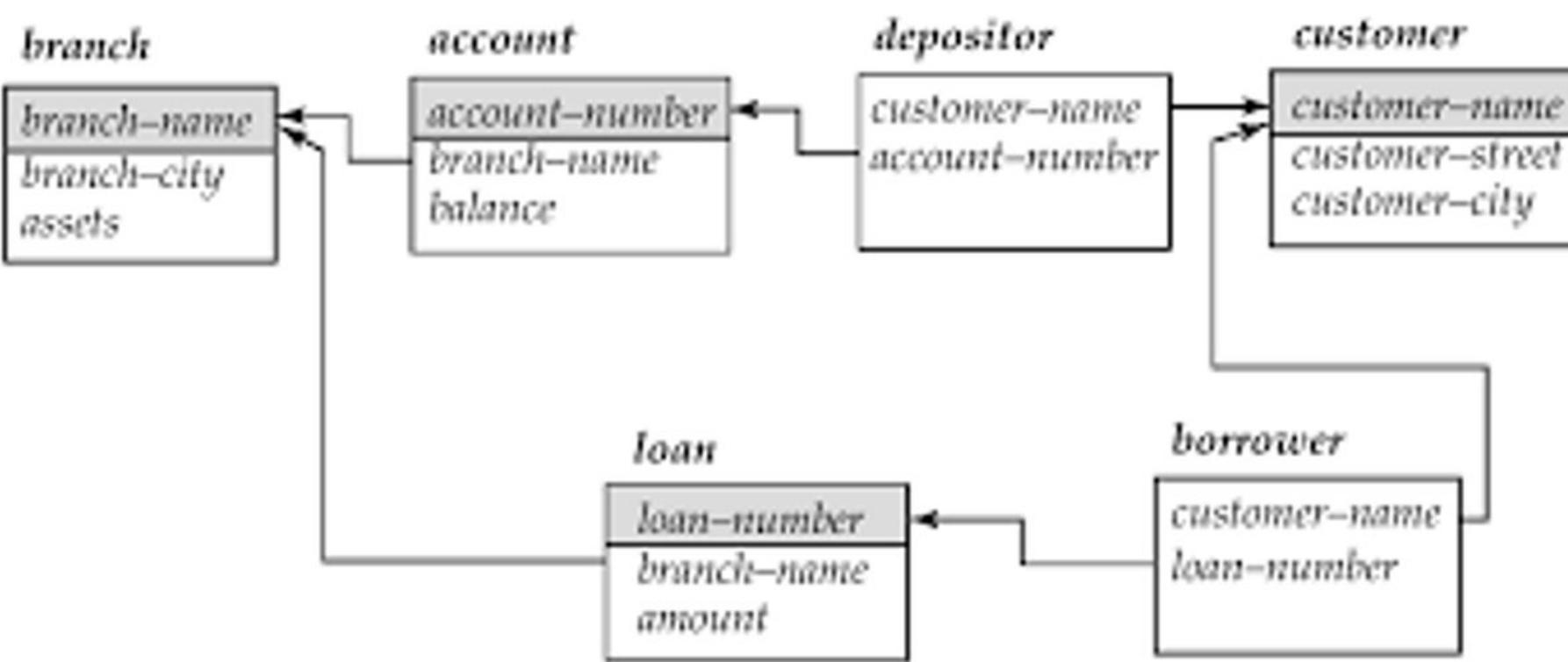
Queries on Multiple Relations

- So far, our example queries were on a single relation. Queries often need to access information from multiple relations.
- The **from** clause by itself defines a Cartesian product of the relations listed in the clause.
- Cartesian product of two relations, which concatenates each tuple of the first relation with every tuple of the second.
- Since the same attribute name may appear in both r_1 and r_2 , we prefix the name of the relation from which the attribute originally came, before the attribute name.
- For those attributes that appear in only one of the two schemas, we shall usually drop the relation-name prefix. This simplification does not lead to any ambiguity.
- Cartesian Product is commutative in nature

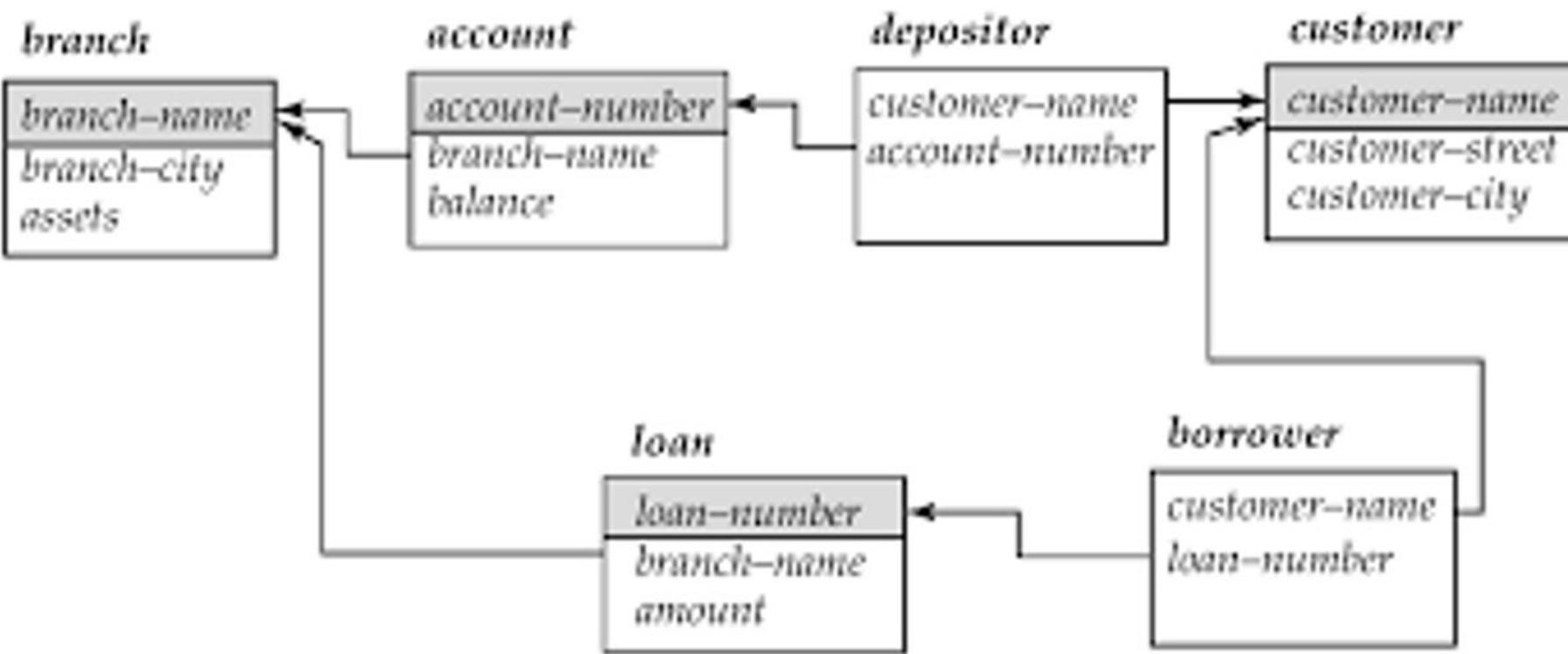
	R
A	B
1	P
2	Q
3	R

	R ₂
B	C
Q	X
R	Y
S	Z

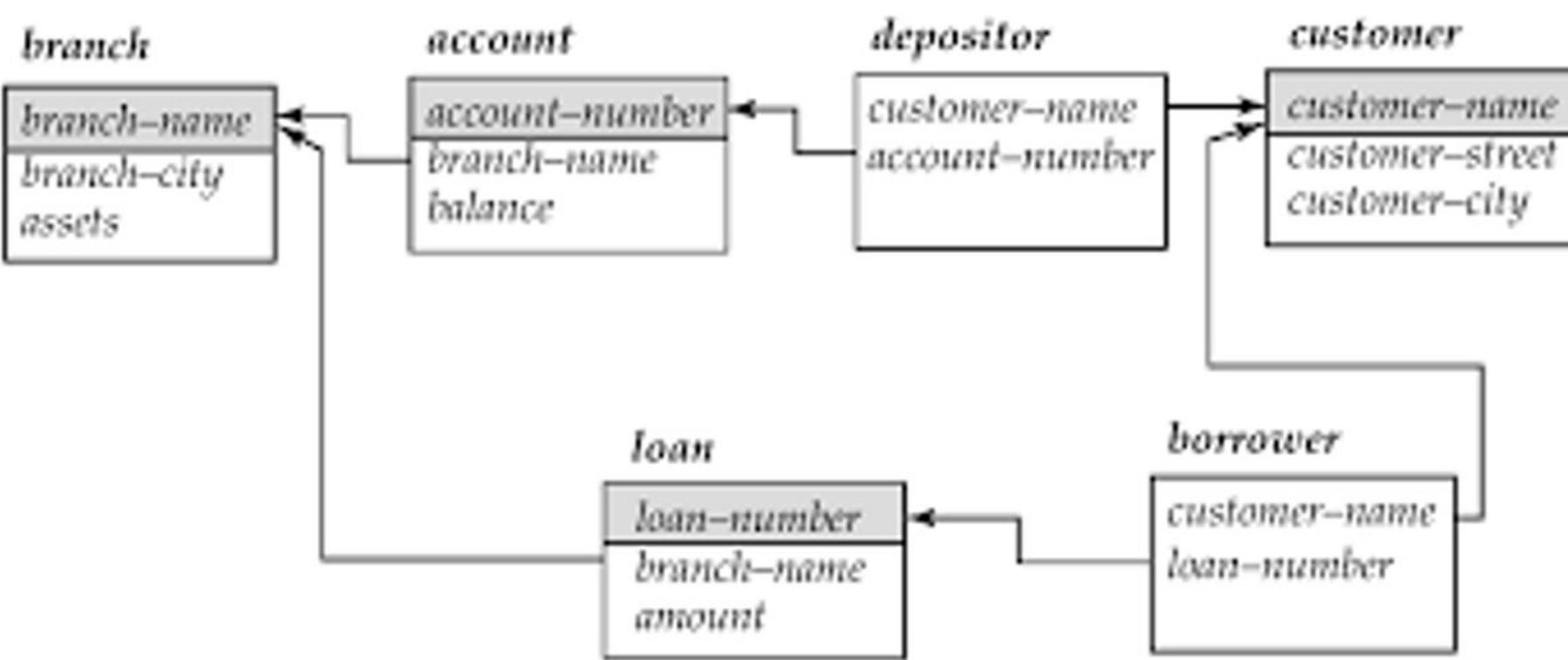
Q Write a SQL query to find the name of all the customers with account balance, who have an account in the bank?



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 Consider the set of relations shown below and the SQL query that follows (Gate-2003) (2 Marks)

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 Given relations $r(w, x)$ and $s(y, z)$, the result of

SELECT DISTINCT w, x FROM r, s

is guaranteed to be same as r , provided **(GATE-2003) (1 Marks)** [Asked in CoCubes 2019]

- (A) r has no duplicates and s is non-empty
- (B) r and s have no duplicates
- (C) s has no duplicates and r is non-empty
- (D) r and s have the same number of tuples

Natural Join

1. To make the life of an SQL programmer easier for this common case, SQL supports an operation called the *natural join*.
2. The **natural join** operation like cartesian product operates on two relations and produces a relation as the result.
3. Natural join considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relations.
4. Notice that we do not repeat those attributes that appear in the schemas of both relations; rather they appear only once.
5. Notice also the order in which the attributes are listed: first the attributes common to the schemas of both relations, second those attributes unique to the schema of the first relation, and finally, those attributes unique to the schema of the second relation.
6. commutative in nature

A **from** clause in an SQL query can have multiple relations combined using natural join, as shown here:

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

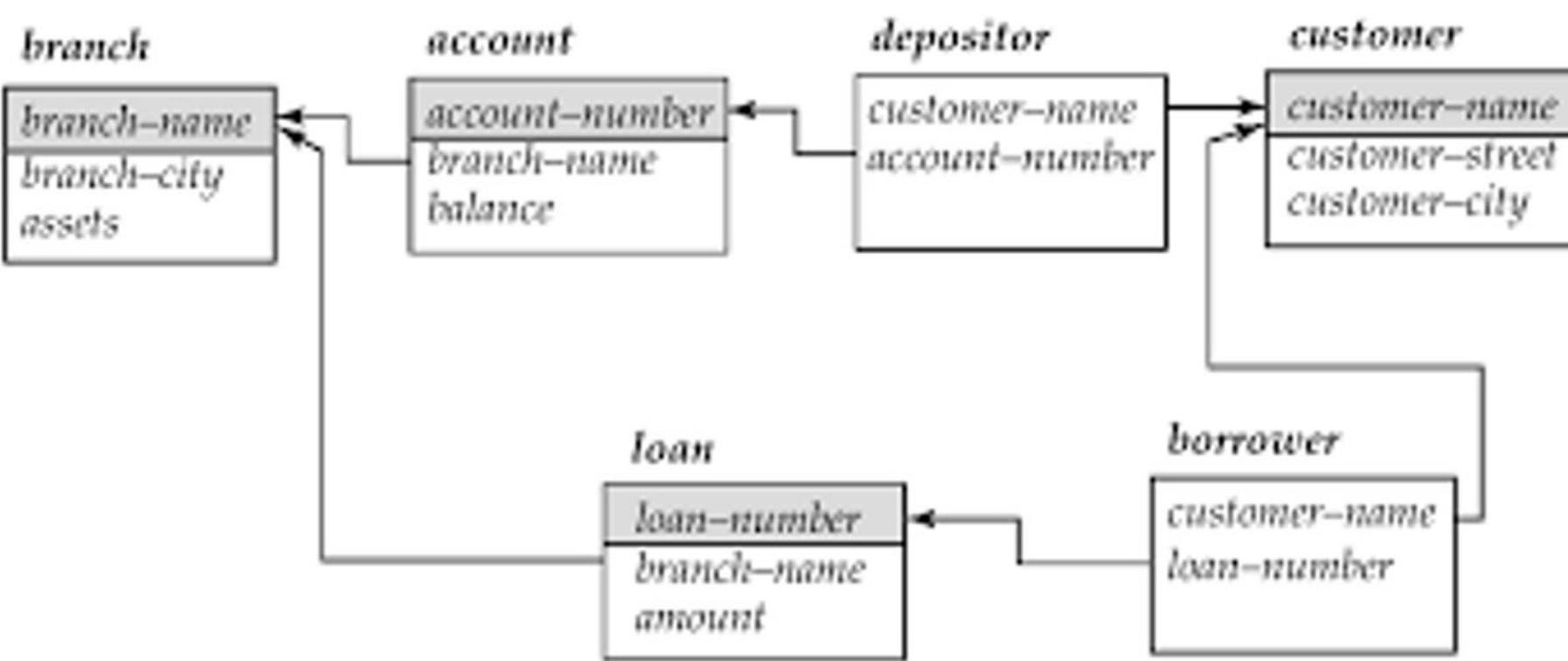
R_1	
A	B
1	P
2	Q
3	R

R_2	
B	C
Q	X
R	Y
S	Z

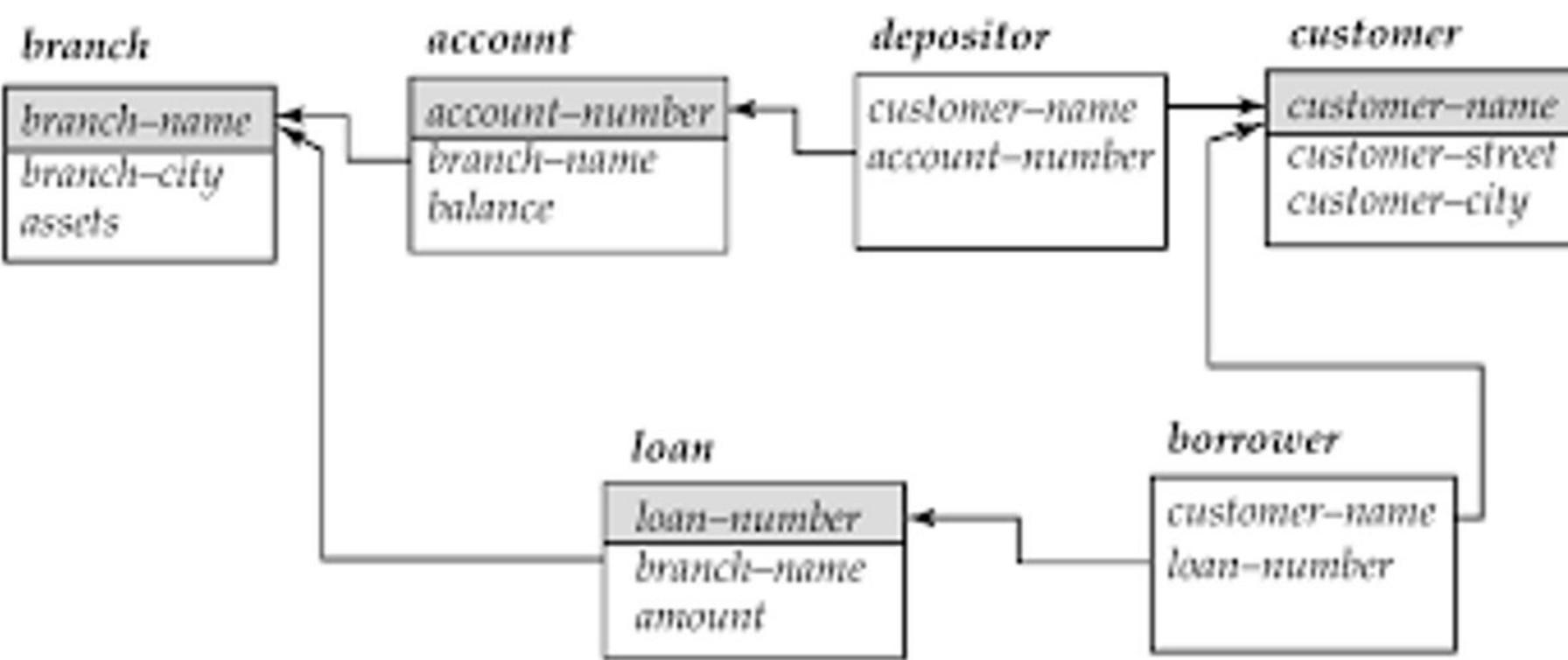
$R_1 * R_2$			
A	R_1 B	R_2 B	C
1	P	Q	X
1	P	R	Y
1	P	S	Z
2	Q	Q	X
2	Q	R	Y
2	Q	S	Z
3	R	Q	X
3	R	R	Y
3	R	S	Z

$R_1 \bowtie R_2$		
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 Database table by name Loan_Records is given below. (Gate - 2011) (2 Marks)

Borrower	Bank_Manager	Loan_Amount
Ramesh	Sunderajan	10000.00
Suresh	Ramgopal	5000.00
Mahesh	Sunderajan	7000.00

What is the output of the following SQL query?

```
SELECT Count(*)  
FROM ( (SELECT Borrower, Bank_Manager FROM Loan_Records) AS S  
NATURAL JOIN  
      (SELECT Bank_Manager, Loan_Amount FROM Loan_Records) AS T );
```

- (A) 3
- (B) 9
- (C) 5
- (D) 6

Q Consider the following relation schema pertaining to a students database:

Student (rollno, name, address)

Enroll (rollno, courseno, coursename)

where the primary keys are shown underlined. The number of tuples in the Student and Enroll tables are 120 and 8 respectively. What are the maximum and minimum number of tuples that can be present in (Student * Enroll), where '*' denotes natural join? **(Gate-2004) (2 Marks)**

(A) 8, 8

(B) 120, 8

(C) 960, 8

(D) 960, 120

- Q** Consider the following SQL query (**GATE-2003**) (**1 Marks**)
- select distinct a₁, a₂,....., a_n
from r₁, r₂,....., r_m
where P
- For an arbitrary predicate P, this query is equivalent to which of the following relational algebra expressions ?
- (A) $\prod_{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n} \sigma_{\rho}(r_1 \times r_2 \times \dots \times r_m)$
- (B) $\prod_{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n} \sigma_{\rho}(r_1 \bowtie r_2 \bowtie \dots \bowtie r_m)$
- (C) $\prod_{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n} \sigma_{\rho}(r_1 \cup r_2 \cup \dots \cup r_m)$
- (D) $\prod_{\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n} \sigma_{\rho}(r_1 \cap r_2 \cap \dots \cap r_m)$

Join /inner join

1. Join or inner join is a operation which works same as cartesian product but when it is used with “using” keyword it provides additional functionality.
2. Join with using – Provides ability to explicitly chose columns which must be used by join for comparison and removal of redundant tuples, i.e. if there are more than one column which are common between two table but we do not want each of them to be considered, then we can use join with using.

1. So join is a form of the natural join construct that allows you to specify exactly which columns should be equated.
 - $R_1(A, B, C)$
 - $R_2(B, C, D)$
2. Consider the operation R_1 join R_2 using (B). The operation is similar to R_1 natural join R_2 , except that a pair of tuples t_1 from R_1 and t_2 from R_2 match if $t_1.B = t_2.B$; even if R_1 and R_2 both have an attribute named C , it is *not* required that $t_1.C = t_2.C$.

Outer Join

1. The join operations we studied earlier that do not preserve nonmatched tuples are called inner join operations, to distinguish them from the outer-join operations.
2. The problem with natural join or join is only those values that appears in both relations will manage to reach final table, but if some value is explicitly in table one or in second table then that information will be lost, and that will be loss of information.
3. The outer join operation works in a manner similar to the join operations we have already studied, but preserve those tuples that would be lost in a join, by creating tuples in the result containing null values.

- There are in fact three forms of outer join:
 1. The **left outer join** (left join) preserves tuples only in the relation named before (to the left of) the left outer join operation.
 2. The **right outer join** (right join) preserves tuples only in the relation named after (to the right of) the right outer join operation.
 3. The **full outer join** preserves tuples in both relations.

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	R_2	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
2	Q	X
3	R	Y

$R_1 \bowtie R_2$		
A	B	C

$R_1 \bowtie R_2$		
A	B	C

$R_1 \bowtie R_2$		
A	B	C

Q Consider the following two tables and four queries in SQL. (GATE - 2018) (1 Marks)

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;

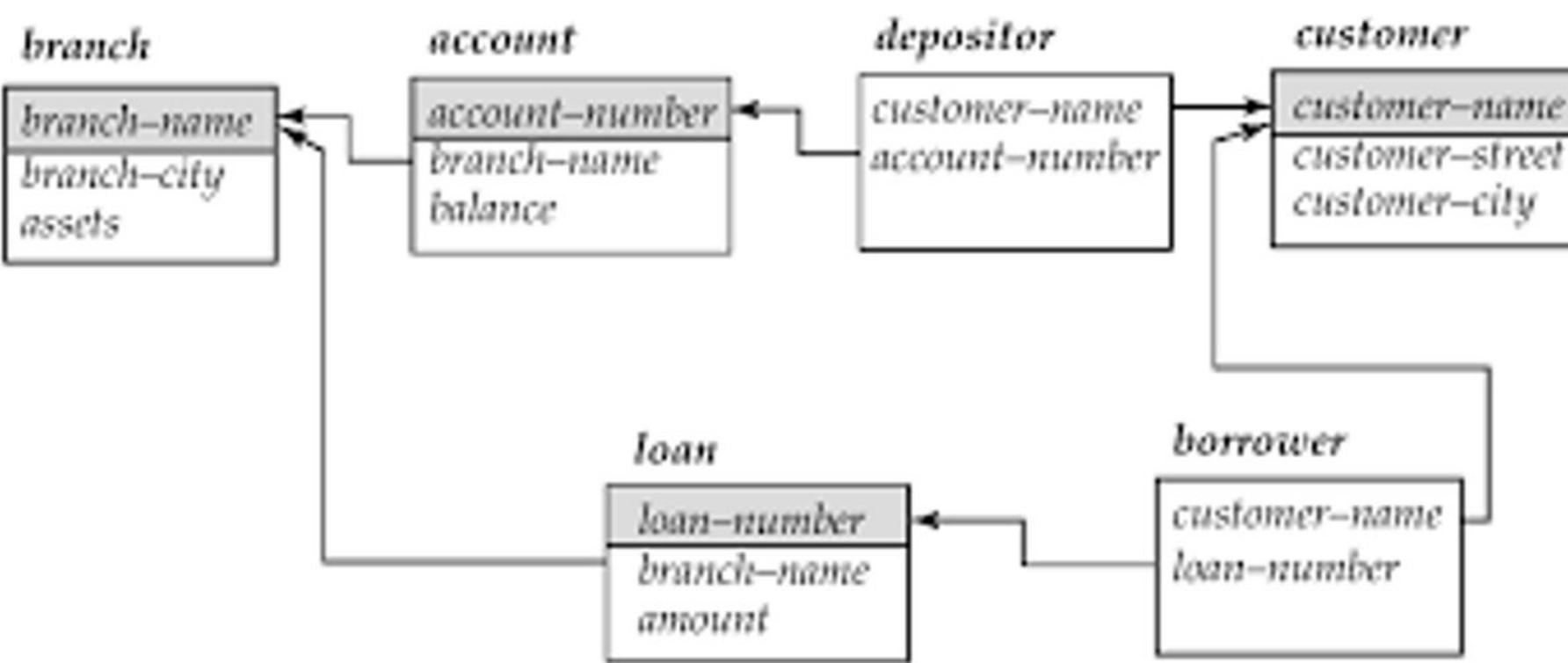
Which one of the queries above is certain to have an output that is a superset of the outputs of the other three queries?

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

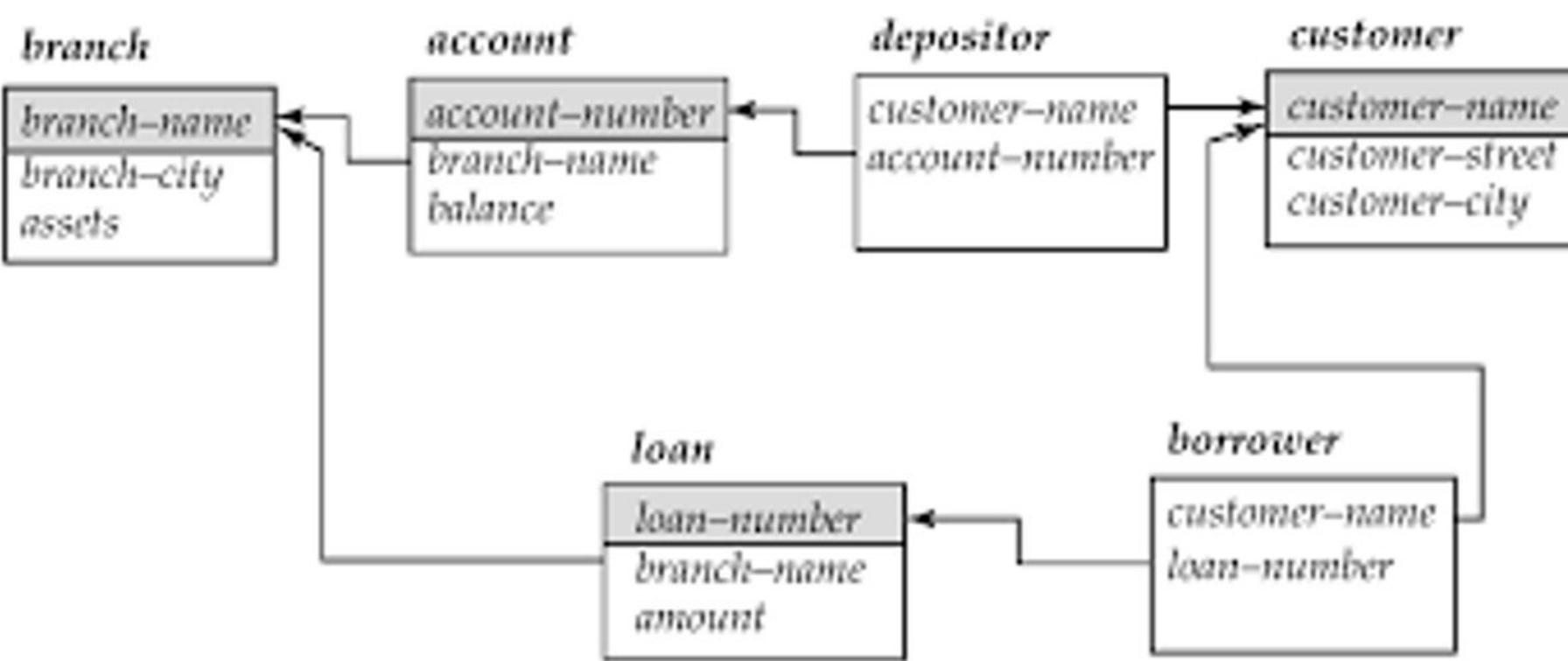
Alias Operation/ rename

1. 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.
2. Aliases are often used to make column names more readable.
3. 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.
4. An alias only exists for the duration of the query.

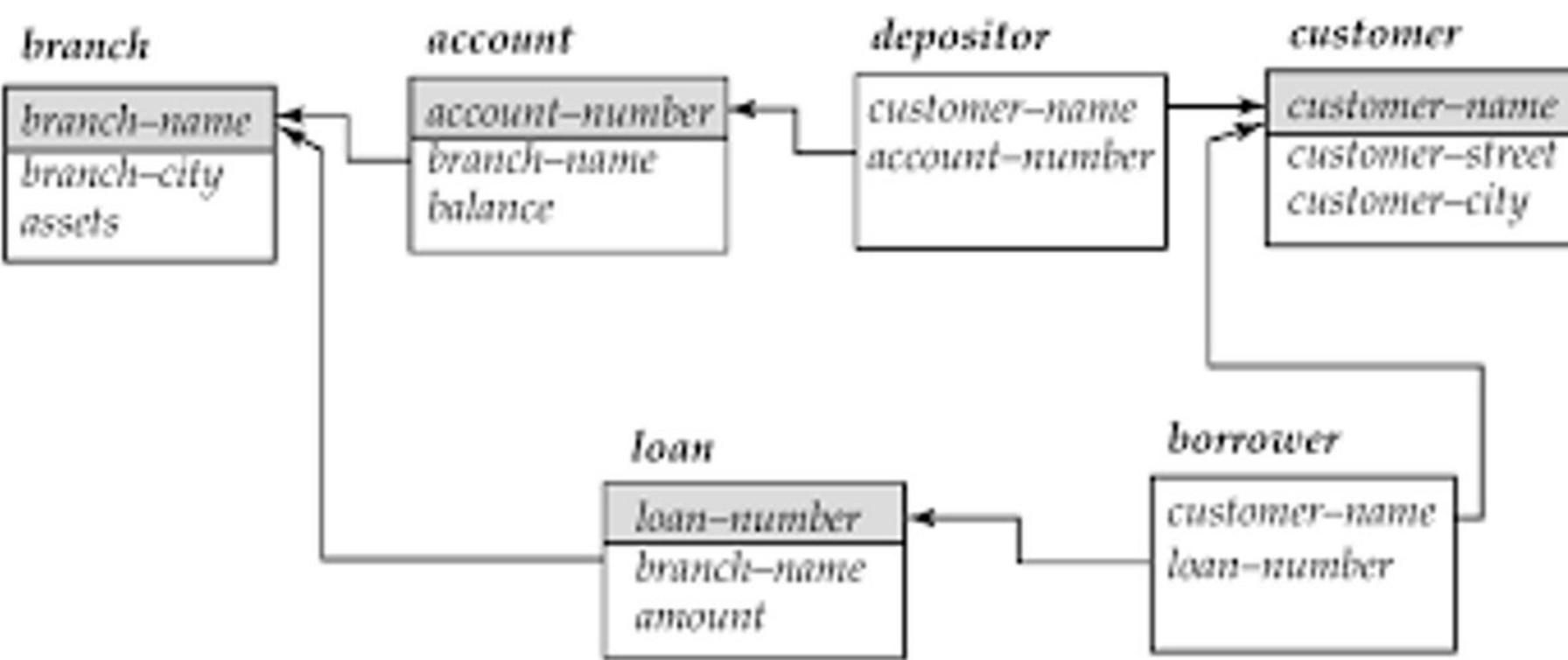
Q Write a SQL query to find the account_no along and balance with 8% interest, as Account, total_balance?



Q Write a SQL query to find all the account number along with branch_name and branch_city?



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

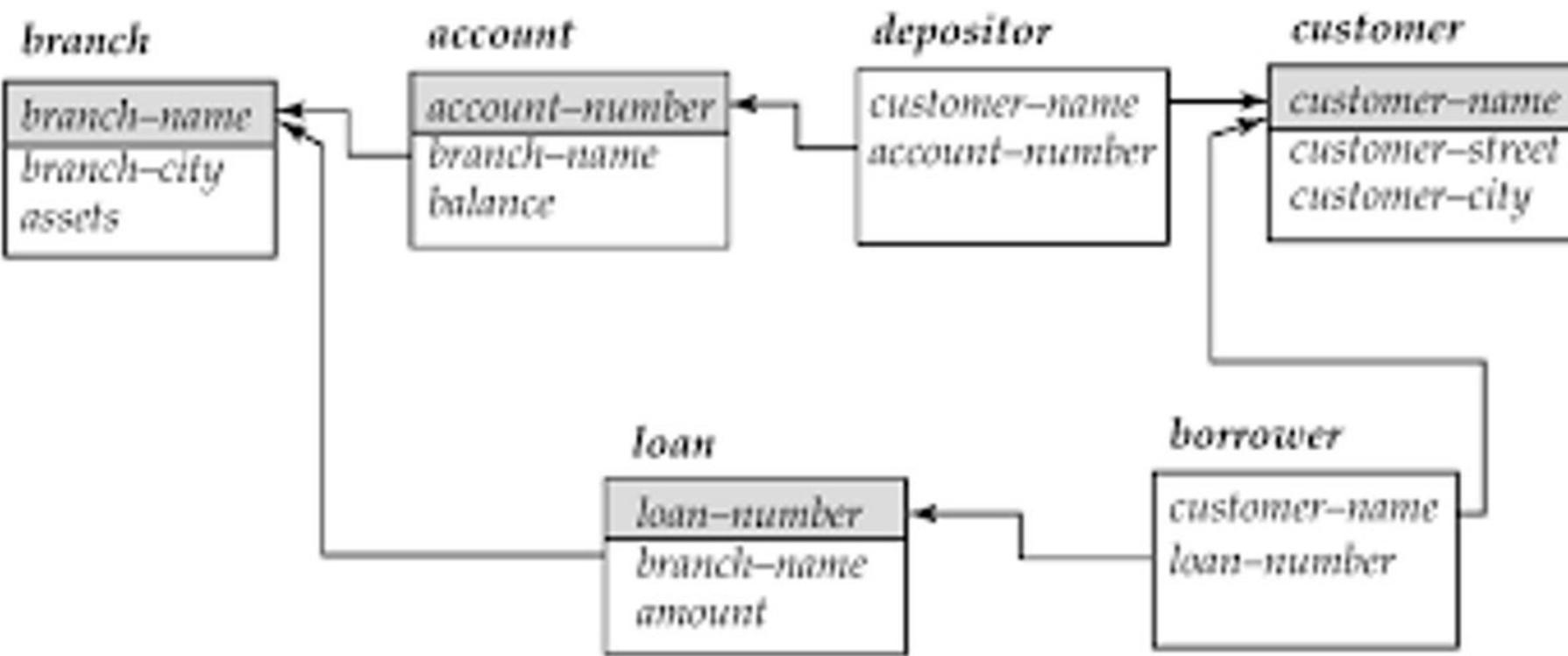


Aggregate Functions

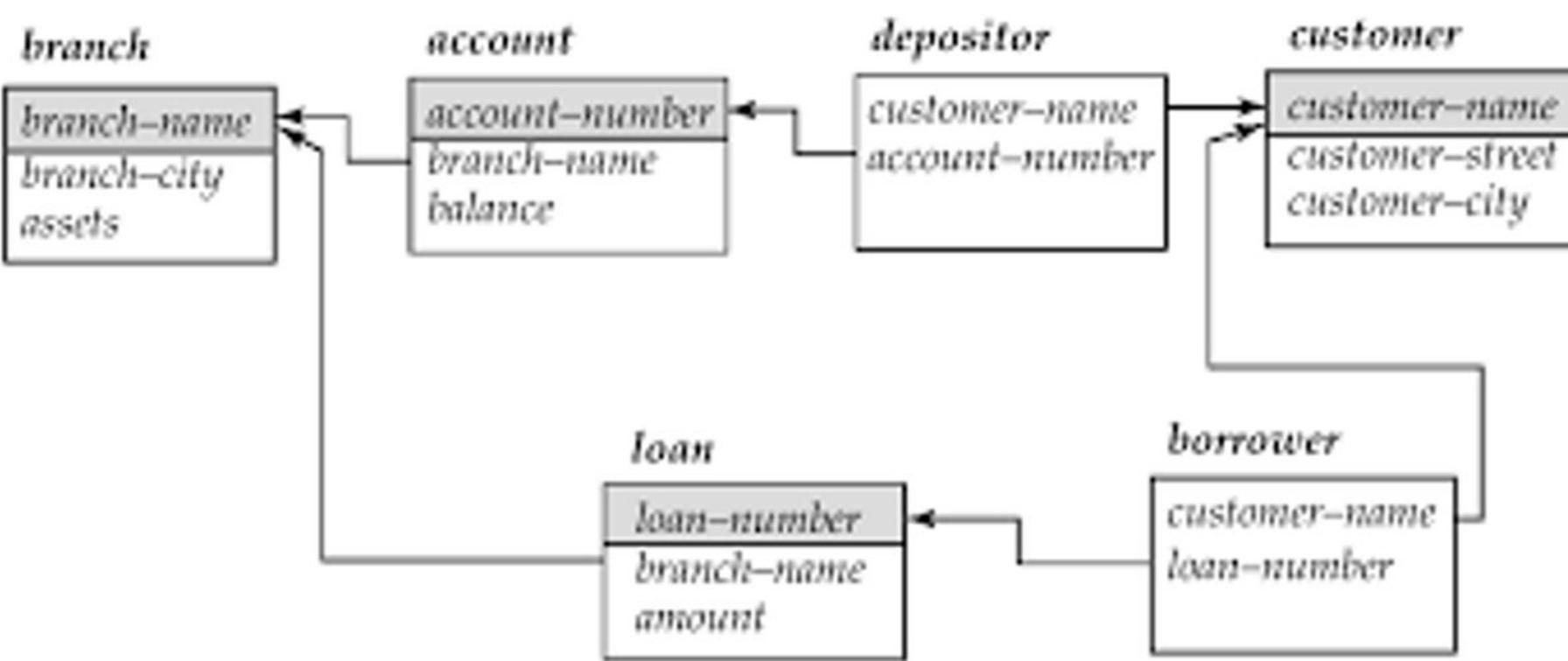
- *Aggregate functions* are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five built-in aggregate functions:
 - Average: **avg**
 - Minimum: **min**
 - Maximum: **max**
 - Total: **sum**
 - Count: **count**

1. 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.
2. 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 (*)**.
3. 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**

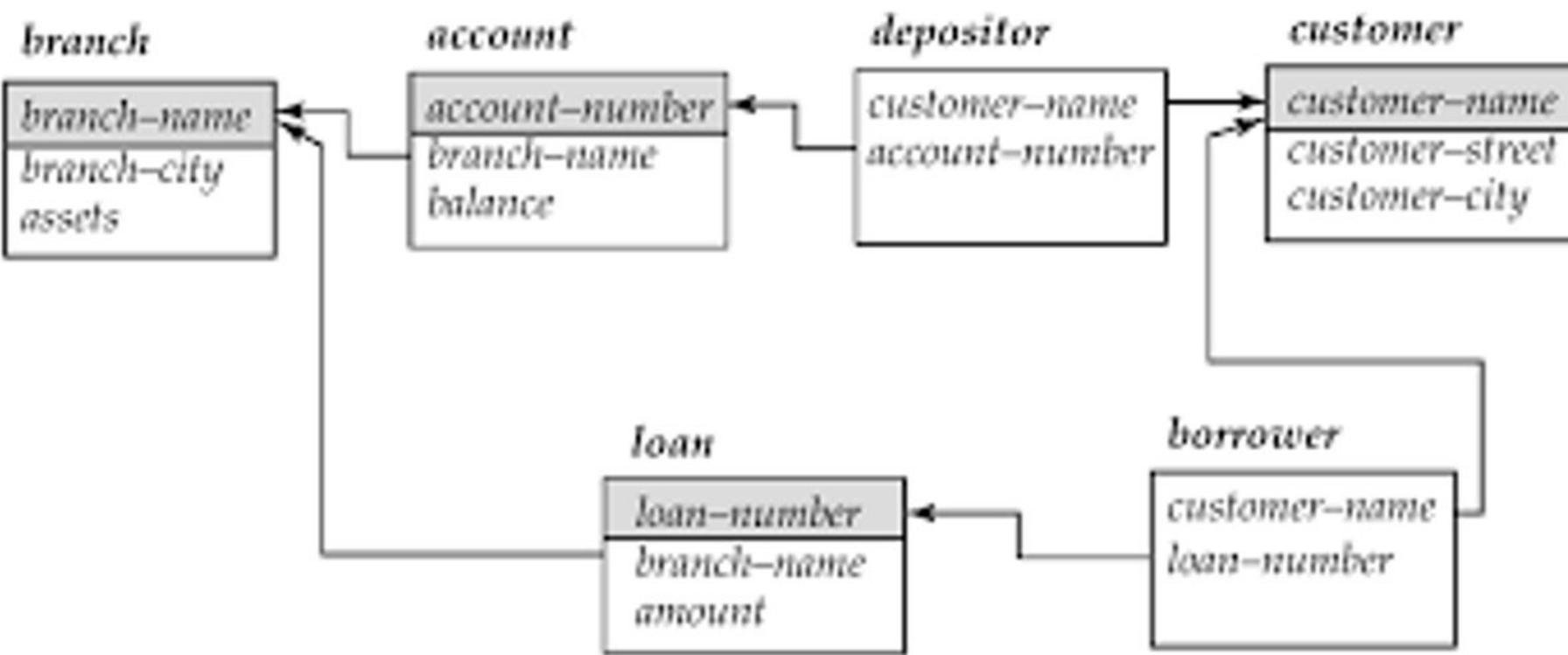
Account_no	balance	Branch_name
Abc123	100	N_delhi
Pqr123	500	S_mumbai
Wyz123	null	S_delhi

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

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?



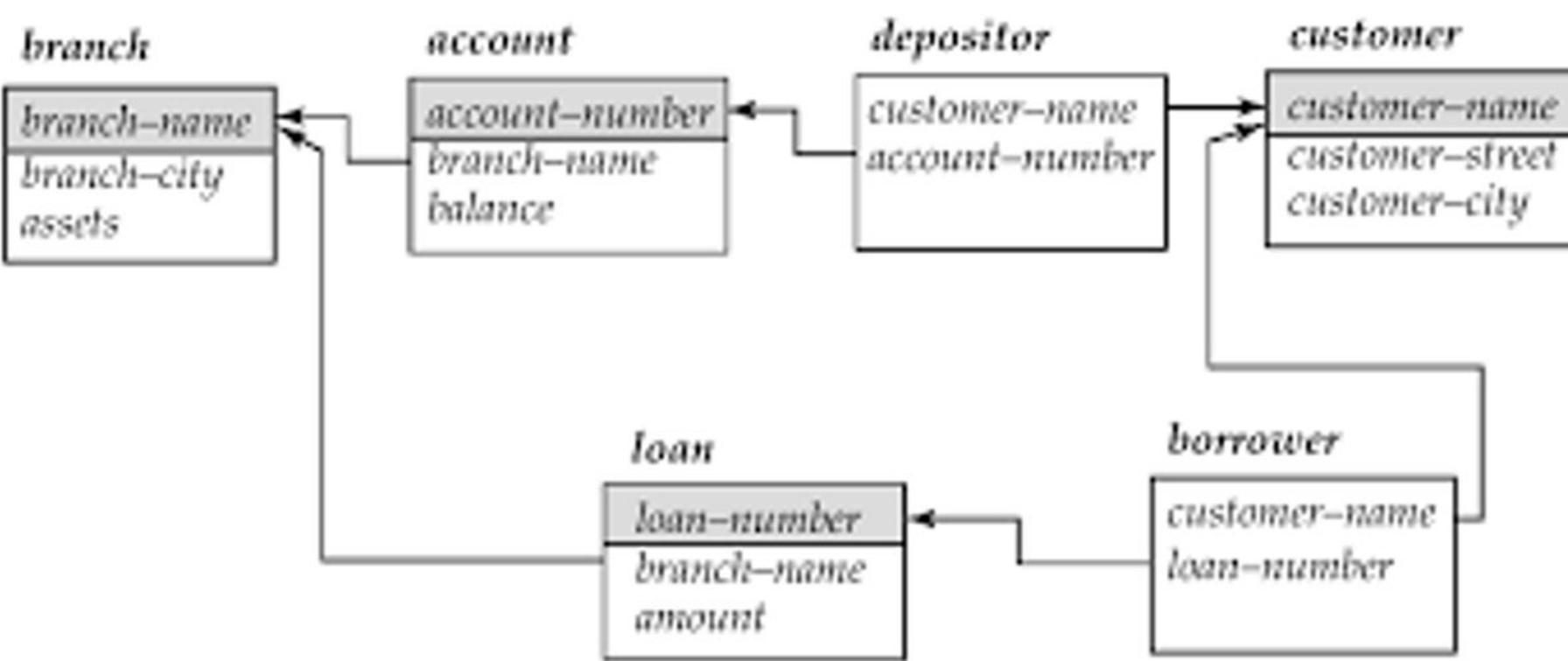
String Operations

1. SQL specifies strings by enclosing them in single quotes, for example, 'Computer'.
2. The SQL standard specifies that the equality operation on strings is case sensitive; as a result the expression 'Computer' = 'computer' evaluates to false.
3. 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.
4. This default behavior can, however, be changed, either at the database level or at the level of specific attributes.

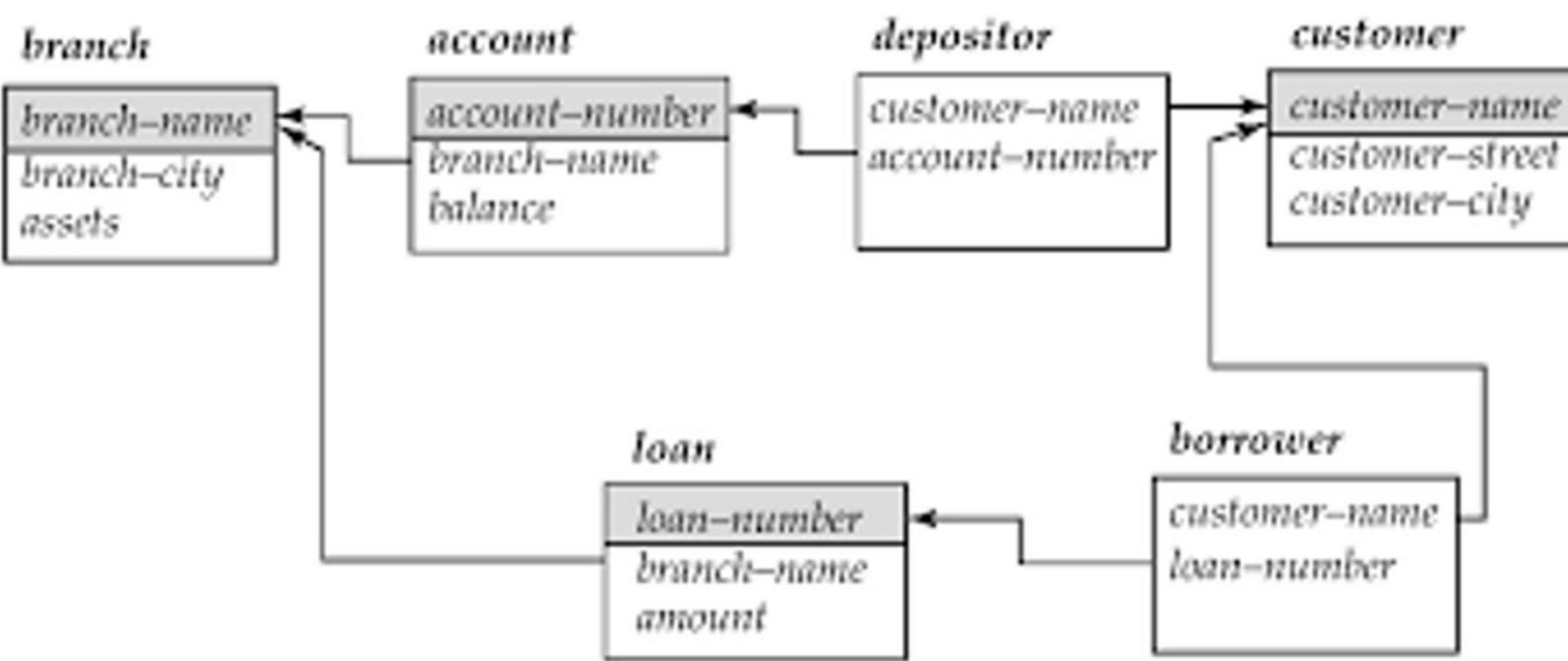
1. SQL also permits a variety of functions on character strings, such as concatenating, extracting substrings, finding the length of strings, converting strings to uppercase and lowercase, removing spaces at the end of the string and so on.
2. There are variations on the exact set of string functions supported by different database systems.
3. 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 ?



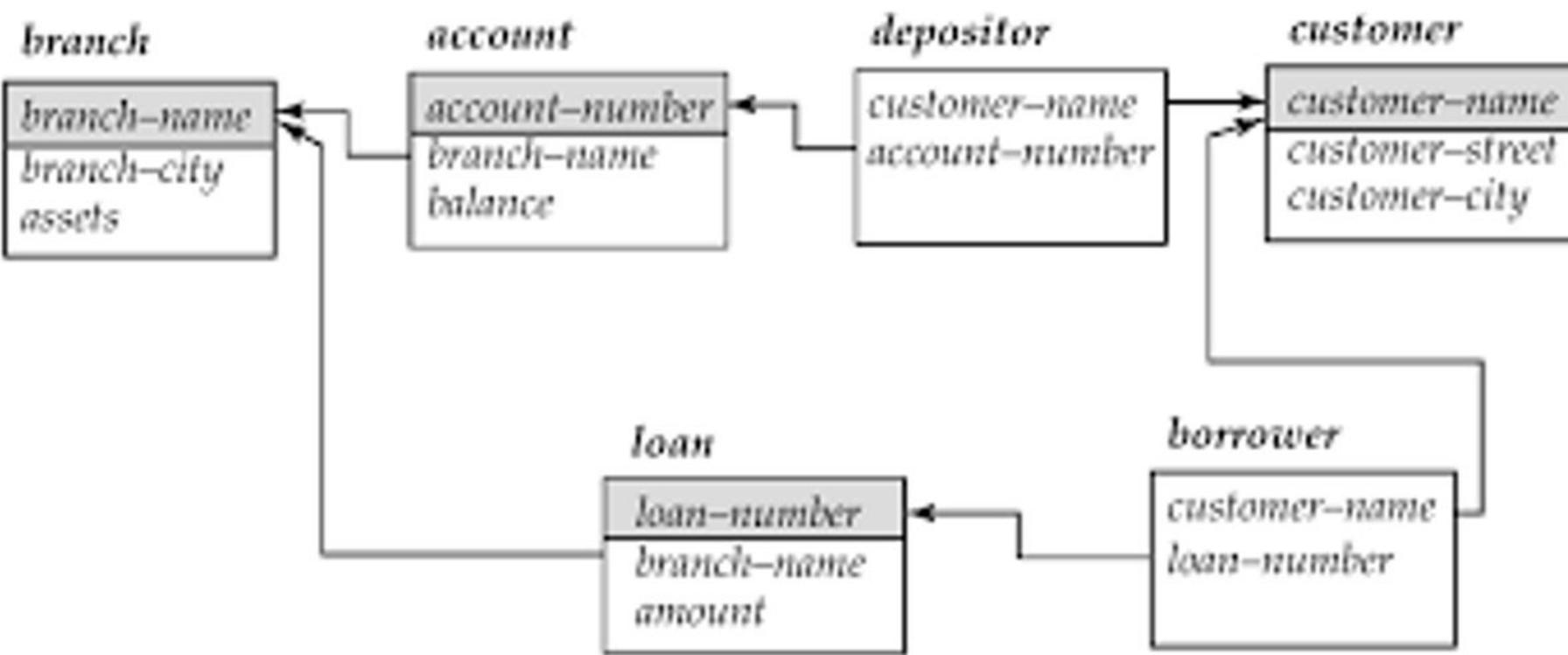
1. For patterns to include the special pattern characters (that is, % and _), SQL allows the specification of an escape character.
2. The escape character is used immediately before a special pattern character to indicate that the special pattern character is to be treated like a normal character.
3. 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:
 4. **like 'ab\%cd%' escape '\'** matches all strings beginning with "ab%cd".
 5. **like 'ab\\cd%' escape '\'** matches all strings beginning with "ab\cd".

1. 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.
2. 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.

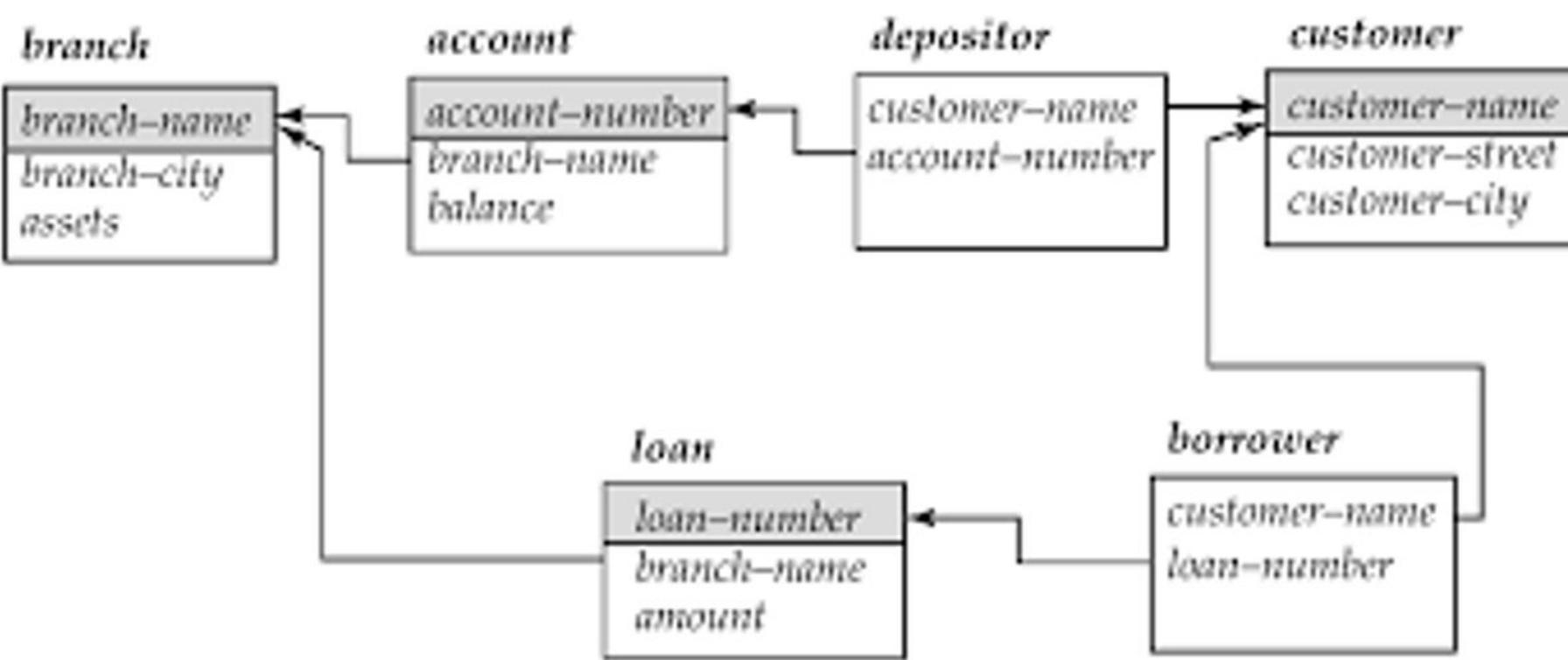
Group by clause

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 A relational database contains two tables Student and Performance as shown below: (GATE - 2019) (2 Marks)

The primary key of the Student table is Roll_no. For the Performance table, the columns Roll_no. and Subject_code together form the primary key. Consider the SQL query given below:

```
SELECT S.Student_name, sum (P.Marks)
FROM Student S, Performance P
WHERE P.Marks > 84
GROUP BY S.Student_name;
```

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

Student	
Roll_no.	Student_name
1	Amit
2	Priya
3	Vinit
4	Rohan
5	Smita

Performance		
Roll_no.	Subject_code	Marks
1	A	86
1	B	95
1	C	90
2	A	89
2	C	92
3	C	80

Q Consider a database that has the relation schema EMP (Empld, EmpName, and DeptName). An instance of the schema EMP and a SQL query on it are given below.
(GATE-2018) (2 Marks)

The output of executing the SQL query is _____

EMP		
Empld	EmpName	DeptName
1	XYA	AA
2	XYB	AA
3	XYC	AA
4	XYD	AA
5	XYE	AB
6	XYF	AB
7	XYG	AB
8	XYH	AC
9	XYI	AC
10	XYJ	AC
11	XYK	AD
12	XYL	AD
13	XYM	AE

```
SELECT AVG(EC.Num)
FROM EC
WHERE (DeptName, Num)IN
    (SELECT DeptName, COUNT(Empld) AS
     EC(DeptName, Num)
    FROM EMP
    GROUP BY DeptName)
```

Q Consider the following relations (Gate-2015) (2 Marks)

```
SELECT S.Student_Name, sum(P.Marks)
FROM Student S, Performance P
WHERE S.Roll_No = P.Roll_No
GROUP BY S.Student_Name
```

The number of rows that will be returned by the SQL query is _____

Students

<u>Roll No</u>	<u>Student Name</u>
1	Raj
2	Rohit
3	Raj

Performance

<u>Roll No</u>	<u>Course</u>	<u>Marks</u>
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Q The employee information in a company is stored in the relation

Employee (name, sex, salary, deptName)

Consider the following SQL query

```
select deptName  
from Employee  
where sex = 'M'  
group by deptName  
having avg (salary) > (select avg (salary) from Employee)
```

It returns the names of the department in which **(Gate-2004) (2 Marks) (NET-JUNE-2013)**

- (A)** the average salary is more than the average salary in the company
- (B)** the average salary of male employees is more than the average salary of all male employees in the company
- (C)** the average salary of male employees is more than the average salary of employees in the same department
- (D)** the average salary of male employees is more than the average salary in the company

- The SQL WITH clause was introduced by Oracle in the Oracle 9i release 2 database. The SQL WITH clause allows you to give a sub-query block a name (a process also called sub-query refactoring), which can be referenced in several places within the main SQL query.
- The clause is used for defining a temporary relation such that the output of this temporary relation is available and is used by the query that is associated with the WITH clause.
- WITH clause is not supported by all database system.

Q Consider the following database table named water_schemes :

The number of tuples returned by the following SQL query is _____. GATE(2015 SET 1)

```
with total(name, capacity) as
  select district_name, sum(capacity)
    from water_schemes
   group by district_name
with total_avg(capacity) as
  select avg(capacity)
    from total
select name
  from total, total_avg
 where total.capacity ≥ total_avg.capacity
```

water - schemes		
scheme_no	district_name	capacity
1	Ajmer	20
1	Bikaner	10
2	Bikaner	10
3	Bikaner	20
1	Churu	10
2	Churu	20
1	Dungargarh	10

1. When an SQL query uses grouping, it is important to ensure that the only attributes that appear in the **select** statement without being aggregated are those that are present in the **group by** clause.
2. In other words, any attribute that is not present in the **group by** clause must appear only inside an aggregate function if it appears in the **select** clause, otherwise the query is treated as erroneous.
3. The Having Clause - At times, it is useful to state a condition that applies to groups rather than to tuples.
4. To express such a query, we use the having clause of SQL. SQL applies predicates in the having clause after groups have been formed.
5. Any attribute that is present in the **having** clause without being aggregated must appear in the **group by** clause, otherwise the query is treated as erroneous.

- The meaning of a query containing aggregation, **group by**, or **having** clauses is defined by the following sequence of operations:
 1. As was the case for queries without aggregation, the **from** clause is first evaluated to get a relation.
 2. If a **where** clause is present, the predicate in the **where** clause is applied on the result relation of the **from** clause.
 3. Tuples satisfying the **where** predicate are then placed into groups by the **group by** clause if it is present. If the **group by** clause is absent, the entire set of tuples satisfying the **where** predicate is treated as being in one group.
 4. The **having** clause, if it is present, is applied to each group; the groups that do not satisfy the **having** clause predicate are removed.
 5. The **select** clause uses the remaining groups to generate tuples of the result of the query, applying the aggregate functions to get a single result tuple for each group.

Q Which of the following statements are TRUE about an SQL query? (Gate-2012) (2 Marks)

P : An SQL query can contain a HAVING clause even if it does not have a GROUP BY clause

Q : An SQL query can contain a HAVING clause only if it has a GROUP BY clause

R : All attributes used in the GROUP BY clause must appear in the SELECT clause

S : Not all attributes used in the GROUP BY clause need to appear in the SELECT clause

- (A)** P and R **(B)** P and S **(C)** Q and R **(D)** Q and S

**Q: Division operation is ideally suited to handle queries of the type:
(NET-DEC-2014)**

- (A) customers who have no account in any of the branches in Delhi.**
- (B) customers who have an account at all branches in Delhi.**
- (C) customers who have an account in at least one branch in Delhi.**
- (D) customers who have only joint account in any one branch in Delhi.**

Nested Subqueries

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

Q Consider the following relations A, B, C. How many tuples does the result of the following relational algebra expression contain? Assume that the schema of $A \cup B$ is the same as that of A.

A		
ID	Name	Age
12	Arun	60
15	Shreya	24
99	Rohit	11

B		
ID	Name	Age
15	Shreya	24
25	Hari	40
98	Rohit	20
99	Rohit	11

C		
ID	Phone	Area
10	2200	02
99	2100	01

```
SELECT A.id  
FROM A  
WHERE A.age > ALL (SELECT B.age
```

```
FROM B  
WHERE B.name = "arun")
```

Q The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list? **(Gate - 2005) (2 Marks)** [Asked in TCS NQT 2020]

```
select title  
from book as B  
where (select count(*)  
      from book as T  
      where T.price > B.price) < 5
```

- (A)** Titles of the four most expensive books
- (B)** Title of the fifth most inexpensive book
- (C)** Title of the fifth most expensive book
- (D)** Titles of the five most expensive books

Q Consider the following relation (GATE-2015) (2 Marks)

Cinema (theater, address, capacity)

Which of the following options will be needed at the end of the SQL query

SELECT P₁.address FROM Cinema P₁

Such that it always finds the addresses of theaters with maximum capacity?

- (A) WHERE P₁. Capacity >= All (select P₂. Capacity from Cinema P₂)
- (B) WHERE P₁. Capacity >= Any (select P₂. Capacity from Cinema P₂)
- (C) WHERE P₁. Capacity > All (select max(P₂. Capacity) from Cinema P₂)
- (D) WHERE P₁. Capacity > Any (select max (P₂. Capacity) from Cinema P₂)

Q Consider the following database table named top_scorer. (Gate-2017) (2 Marks)

```
SELECT ta.player
FROM top_scorer AS ta
WHERE ta.goals > ALL ( SELECT tb.goals
    FROM top_scorer AS tb
    WHERE tb.country = 'Spain' )AND ta.goals > ANY (SELECT tc.goals
    FROM top_scorer AS tc
    WHERE tc.country = 'Germany')
```

The number of tuples returned by the above SQL query is ____.

top_scorer		
player	country	goals
Klose	Germany	16
Ronaldo	Brazil	15
G Müller	Germany	14
Fontaine	France	13
Pelé	Brazil	12
Klinsmann	Germany	11
Kocsis	Hungary	11
Batistuta	Argentina	10
Cubillas	Peru	10
Lato	Poland	10
Lineker	England	10
T Müller	Germany	10
Rahn	Germany	10

Q Given the following schema (GATE-2014) (2 Marks)

employees(emp-id, first-name, last-name, hire-date, dept-id, salary)

departments(dept-id, dept-name, manager-id, location-id)

You want to display the last names and hire dates of all latest hires in their respective departments in the location ID 1700. You issue the following query:

```
SELECT last-name, hire-date  
FROM employees  
WHERE (dept-id, hire-date) IN (SELECT dept-id, MAX(hire-date)  
                               FROM employees JOIN departments USING(dept-id)  
                               WHERE location-id = 1700  
                               GROUP BY dept-id);
```

What is the outcome?

- (A) It executes but does not give the correct result.
- (B) It executes and gives the correct result.
- (C) It generates an error because of pairwise comparison.
- (D) It generates an error because the GROUP BY clause cannot be used with table joins in a subquery

Q SQL allows tuples in relations, and correspondingly defines the multiplicity of tuples in the result of joins. Which one of the following queries always gives the same answer as the nested query shown below (**GATE-2014**) (**1 Marks**)

select * from R where a in (select S.a from S)

(A) select R.* from R, S where R.a=S.a

(B) select distinct R.* from R,S where R.a=S.a

(C) select R.* from R,(select distinct a from S) as S₁ where R.a=S₁.a

(D) select R.* from R,S where R.a=S.a and is unique R

Q Consider the relational database with the following four schemas and their respective instances. (GATE 2022) (2 MARKS)

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 _____.

Q Consider the following relational schema (Gate-2014) (2 Marks)

employee(empld, empName, empDept)

customer(custId, custName, salesRepld, rating)

salesRepld is a foreign key referring to empld of the employee relation. Assume that each employee makes a sale to at least one customer. What does the following query return?

```
SELECT empName  
FROM employee E  
WHERE NOT EXISTS (SELECT custId  
                   FROM customer C  
                   WHERE C.salesRepld = E.Empld AND C.rating <> 'GOOD');
```

- (a) Names of all the employees with at least one of their customers having a 'GOOD' rating.
- (B) Names of all the employees with at most one of their customers having a 'GOOD' rating.
- (C) Names of all the employees with none of their customers having a 'GOOD' rating.
- (D) Names of all the employees with all their customers having a 'GOOD' rating.

Q A relational schema for a train reservation database is given below.

Passenger (pid, pname, age)

Reservation (pid, class, tid)

What pids are returned by the following SQL query for the above instance of the tables? **(Gate-2010) (2 Marks)**

SLECT pid

FROM Reservation

WHERE class 'AC' AND EXISTS (SELECT *

 FROM Passenger

 WHERE age > 65 AND Passenger.pid = Reservation.pid)

(A) 1, 0

(B) 1, 2

(C) 1, 3

(D) 1, 5

Passenger		
pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Reservation		
pid	class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

Q Consider the following Employee table (Gate-2015) (2 Marks)

How many rows are there in the result of following query?

```
SELECT E.ID  
FROM Employee E  
WHERE EXISTS (SELECT E2.salary  
              FROM Employee E2  
              WHERE E2.DeptName
```

ID	salary	DeptName
1	10000	EC
2	40000	EC
3	30000	CS
4	40000	ME
5	50000	ME
6	60000	ME
7	70000	CS

Q Consider the table **employee(empld, name, department, salary)** and the two queries Q_1, Q_2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table? **(Gate-2007) (2 Marks)**

Q_1 : Select e.empld

From employee e

Where not exists (Select *

From employee s

where s.department = "5" and s.salary >=e.salary)

Q_2 : Select e.empld

From employee e

Where e.salary > Any (Select distinct salary

From employee s

Where s.department = "5")

(A) Q_1 is the correct query

(B) Q_2 is the correct query

(C) Both Q_1 and Q_2 produce the same answer.

(D) Neither Q_1 nor Q_2 is the correct query

Q In SQL, relations can contain null values, and comparisons with null values are treated as unknown. Suppose all comparisons with a null value are treated as false. Which of the following pairs is not equivalent? **(Gate-2000) (2 Marks)**

(A) $x = 5 \text{ AND } \text{not}(\text{not}(x = 5))$

(B) $x = 5 \text{ AND } x > 4 \text{ and } x < 6$, where x is an integer

(C) $x < 5 \text{ AND } \text{not } (x = 5)$

(D) None of the above

Q. Consider the following relational schema:

Students (rollno: integer, name: string, age: integer, cgpa: real)

Courses (courseno: integer, cname: string, credits: integer)

Enrolled (rollno: integer, courseno: integer, grade: string)

Which of the following options is/are correct SQL query/queries to retrieve the names of the students enrolled in course number (i.e., courseno) 1470? **(Gate 2025)**

- A) SELECT S.name FROM Students S WHERE EXISTS (SELECT * FROM Enrolled E WHERE E.courseno = 1470 AND E.rollno = S.rollno);
- B) SELECT S.name FROM Students S WHERE SIZEOF (SELECT * FROM Enrolled E WHERE E.courseno = 1470 AND E.rollno = S.rollno) > 0;
- C) SELECT S.name FROM Students S WHERE 0 < (SELECT COUNT(*) FROM Enrolled E WHERE E.courseno = 1470 AND E.rollno = S.rollno);
- D) SELECT S.name FROM Students S NATURAL JOIN Enrolled E WHERE courseno = 1470;

Q. Consider the following database tables of a sports league. (Gate 2025)

player (pid, pname, age)

team(tid, tname, city, cid)

coach (cid, cname)

members (pid,tid)

An instance of the table and an SQL query are given.

player		
pid	pname	age
1	Jasprit	31
2	Atharva	24
3	Ishan	26
4	Axar	30

coach	
cid	cname
101	Ricky
102	Mark
104	Teevor

team			
tid	tname	city	cid
10	MI	Mumbai	102
20	DC	Delhi	101
30	PK	Mohali	103

members	
pid	tid
1	10
2	30
3	10
4	20

SELECT MIN (P.age) FROM player P WHERE P.pid IN (SELECT M.pid FROM team T, coach C, members M WHERE C.cname = 'Mark' AND T.cid = C.cid AND M.tid = T.tid)

The value returned by the given SQL query is _____. (Answer in integer)

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

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

Consider the following SQL query:

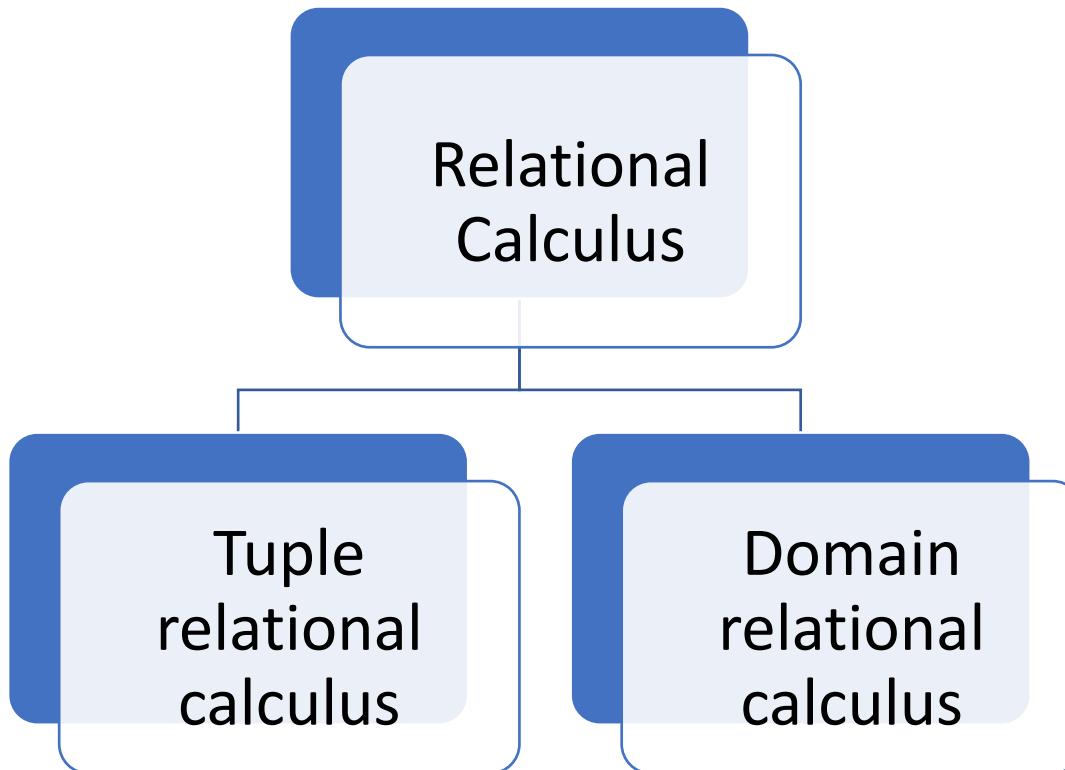
```
select deptId, count(*)  
from emp  
where gender = "female" and salary > (select avg(salary)from emp)  
group by deptId;
```

The above query gives, for each department in the company, the number of female employees whose salary is greater than the average salary of **(GATE 2021) (2 MARKS)**

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

Relational Calculus

- Relational calculus is non-procedural query language, where we have to define what to get and not how to get it



Tuple Relational Calculus

- TRC is based on specifying a number of tuple variables. Each tuple variable usually range over a particular database relation, meaning that the variable may takes as its value from any individual tuple of a relation.
- A simple tuple relational calculus query is of the form.

$$\{t \mid \text{Condition}(t)\}$$

- Where t is a tuple variable and $\text{condition}(t)$ is a conditional expression involving. The result of such a query is the set of all tuple t that satisfy $\text{condition}(t)$.

- . We use $t[A]$ or $t.A$ to denote the value of tuple t on attribute A .
- . we use $t \in r$ or $r(t)$ to denote that tuple t is in relation r .

Student(Roll No, Name, Branch)

Q Find the details of all computer science students?

SQL: select * from student where branch = CSE

RA: $\{\sigma_{\text{branch} = \text{CSE}} (\text{Student})\}$

TRC:

DRC:

Student(Roll No, Name, Branch)

Q Find the details of all computer science students?

SQL: select * from student where branch = CSE

RA: $\{\sigma_{\text{branch} = \text{CSE}} (\text{Student})\}$

TRC: {t | Student(t) \cap t. branch = CSE}

DRC:

Student(Roll No, Name, Branch)

Q Find the Roll No of all computer science students?

SQL: select Roll No from student where branch = CSE

RA: $\{\prod_{\text{sname}} (\sigma_{\text{branch} = \text{CSE}} (\text{Student}))\}$

TRC:

DRC:

Student(Roll No, Name, Branch)

Q Find the Roll No of all computer science students?

SQL: select Roll No from student where branch = CSE

RA: $\{\prod_{\text{roll no}} (\sigma_{\text{branch} = \text{CSE}} (\text{Student}))\}$

TRC: $\{t. \text{ Roll No} \mid \text{Student}(t) \cap t. \text{ branch} = \text{CSE}\}$

DRC:

Formal Definition

- A tuple-relational-calculus expression is of the form: $\{t \mid P(t)\}$ where P is a formula.
- Several tuple variables may appear in a formula.
- A tuple variable is said to be a **free variable** unless it is quantified by a \exists or \forall .
- A tuple variable is said to be a bounded variable if it is quantified by \exists or \forall .

- $P(t)$ may have various conditions logically combined with OR (\vee), AND (\wedge), NOT(\neg)).
- It also uses quantifiers:
 - $\exists t \in r (Q(t))$ = "there exists" a tuple in t in relation r such that predicate $Q(t)$ is true.
 - $\forall t \in r (Q(t))$ = $Q(t)$ is true "for all" tuples in relation r .

- A tuple-relational-calculus formula is built up out of atoms. An atom has one of the following forms:
 - $s \in r$, where s is a tuple variable and r is a relation.
 - $s[x] \text{ op } u[y]$, where s and u are tuple variables, x is an attribute on which s is defined, y is an attribute on which u is defined, and op is a comparison operator ($<$, \leq , $=$, \neq , \geq , $>$); we require that attributes x and y have domains whose members can be compared by .
 - $s[x] \text{ op } c$, where s is a tuple variable, x is an attribute on which s is defined, op is a comparison operator, and c is a constant in the domain of attribute x .

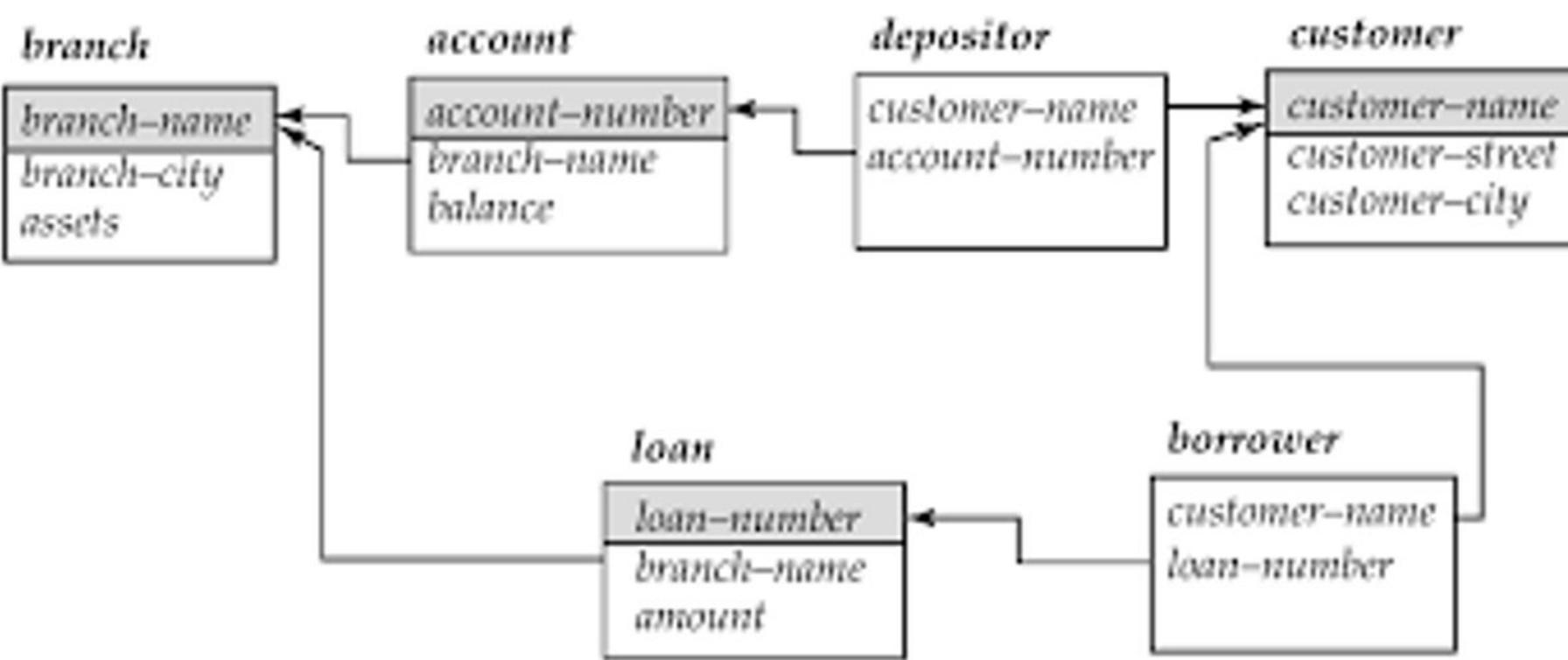
- . We build up formulae from atoms by using the following rules:
 - An atom is a formula.
 - If P_1 is a formula, then so are $\neg P_1$ and (P_1) .
 - If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
 - If $P_1(s)$ is a formula containing a free tuple variable s , and r is a relation,
 - then $\exists s \in r (P_1(s))$ and $\forall s \in r (P_1(s))$

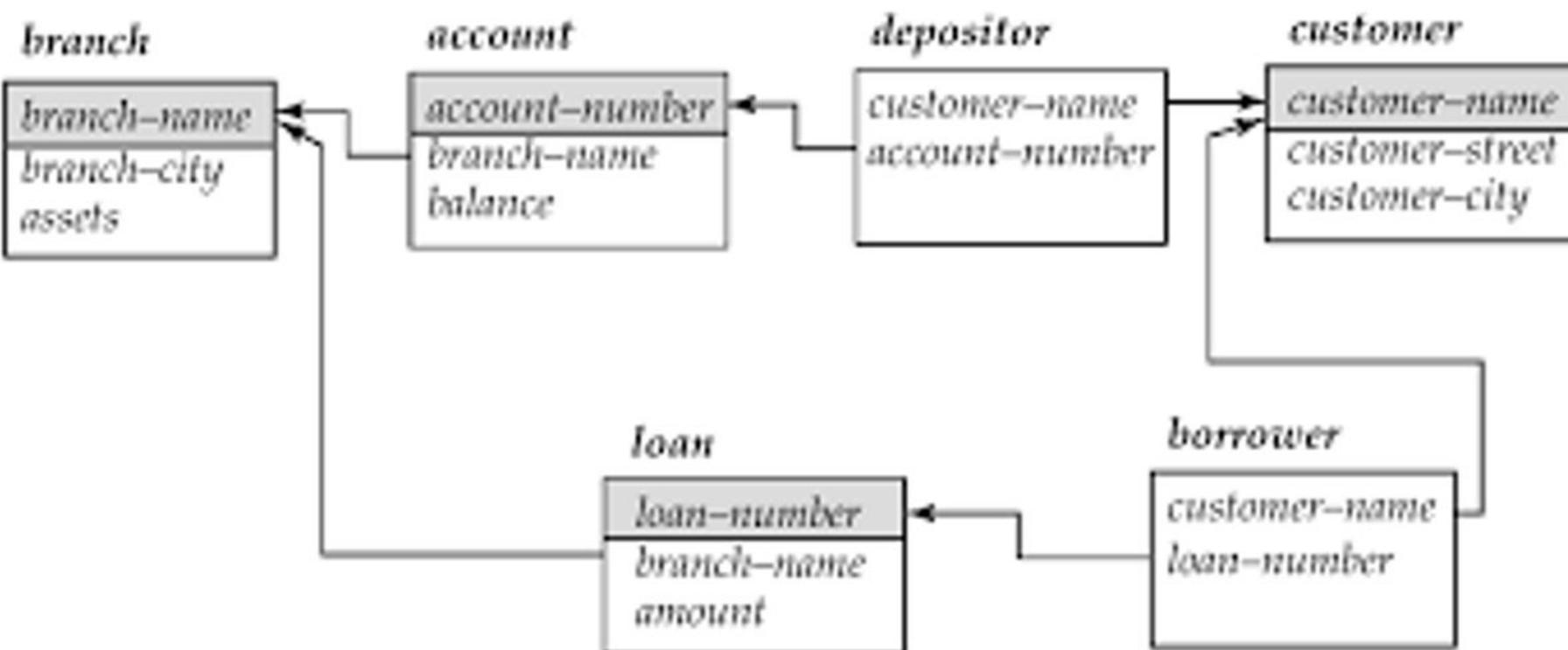
In the tuple relational calculus, equivalences include the following three rules:

1. $P_1 \wedge P_2$ is equivalent to $\neg(\neg(P_1) \vee \neg(P_2))$.
2. $\forall t \in r (P_1(t))$ is equivalent to $\neg \exists t \in r (\neg P_1(t))$.
3. $P_1 \Rightarrow P_2$ is equivalent to $\neg(P_1) \vee P_2$.

Q Find all the details of loan for amount over 1200?

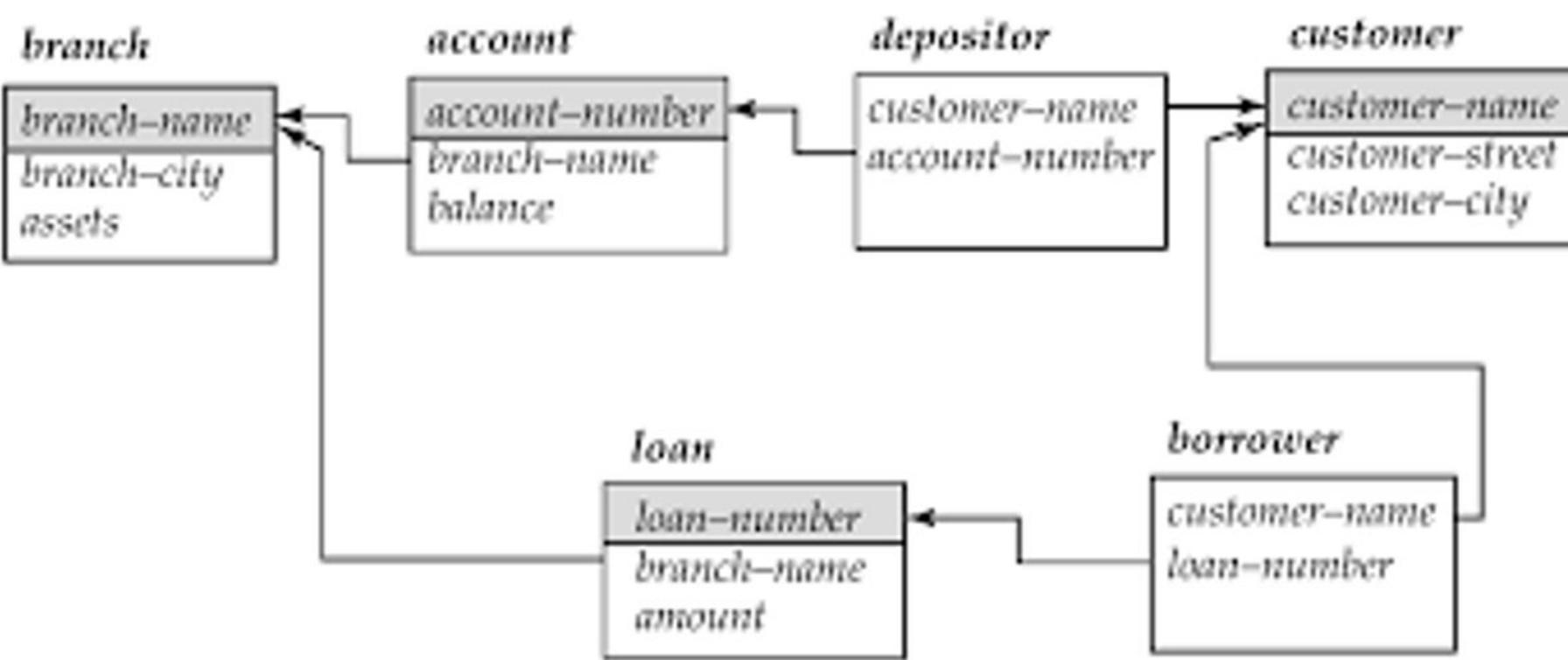
{t| t ∈ loan ∧ t[amount] > 1200 }

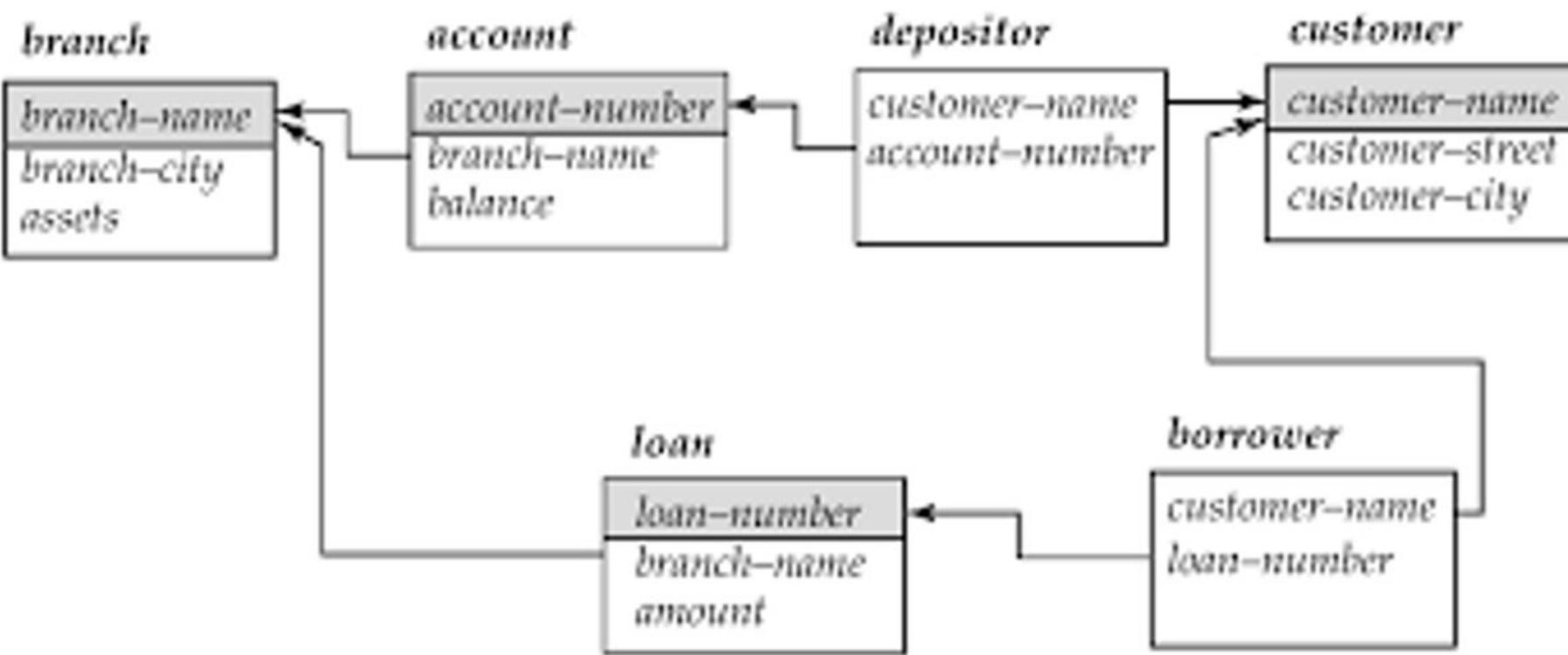


$$\{t \mid t \in \text{loan} \wedge t[\text{amount}] > 1200\}$$


Q Find the loan number for each loan of amount over 1200?

{t| $\exists s \in \text{loan} (t[\text{loan number}] = s[\text{loan number}] \wedge s[\text{amount}] > 1200)$ }



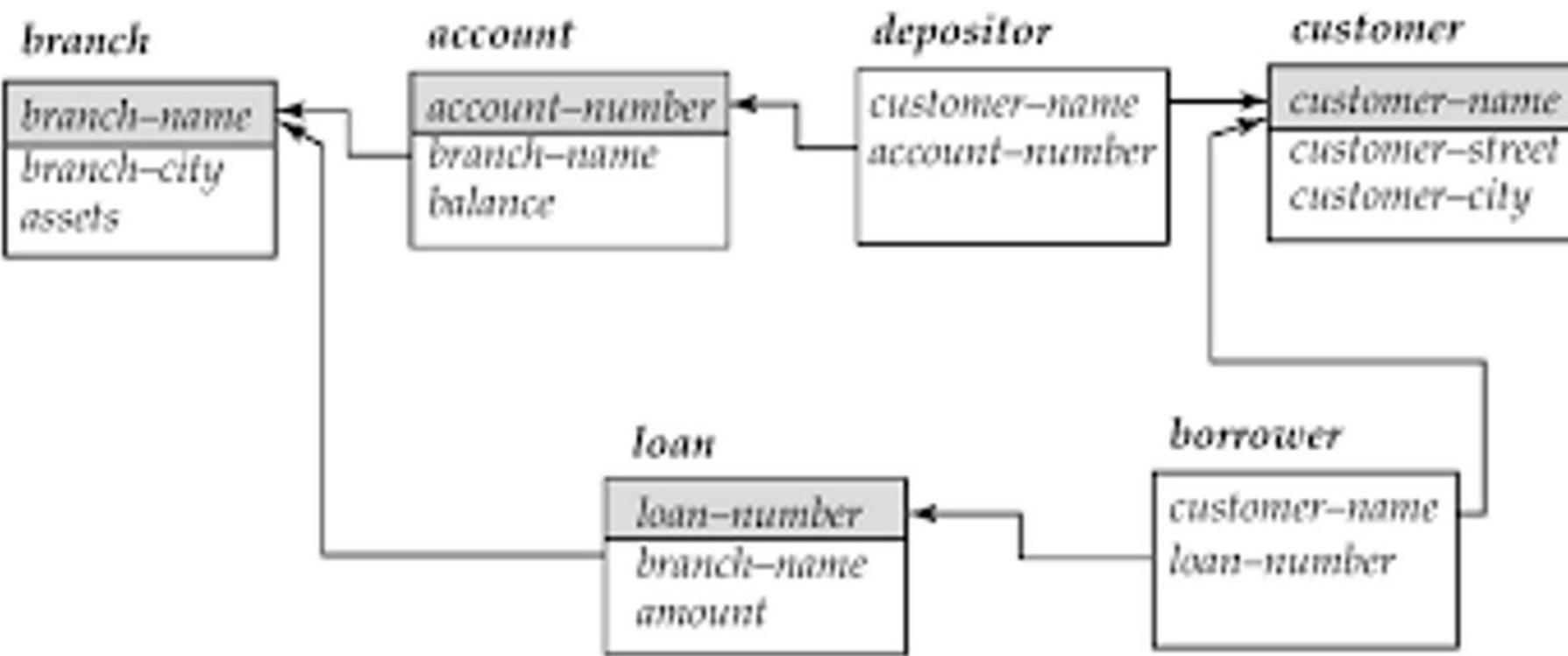
$$\{t \mid \exists s \in \text{loan} (t[\text{loan number}] = s[\text{loan number}] \wedge s[\text{amount}] > 1200) \}$$


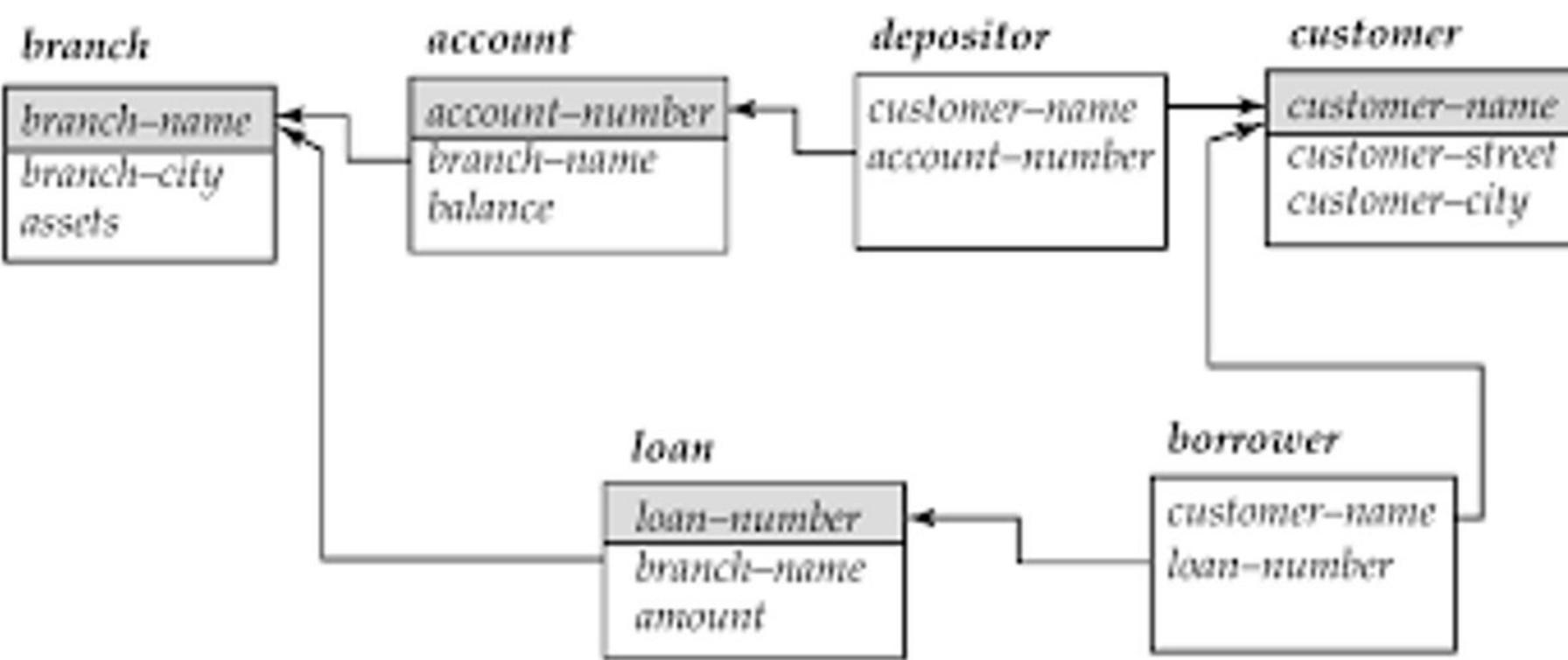
Q Find the name of all the customers who have a loan from Noida branch?

{t| $\exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}])$

$\wedge \exists u \in \text{loan} (u[\text{customer name}] = s[\text{customer name}])$

$\wedge u[\text{branch name}] = \text{'Noida'}$ }

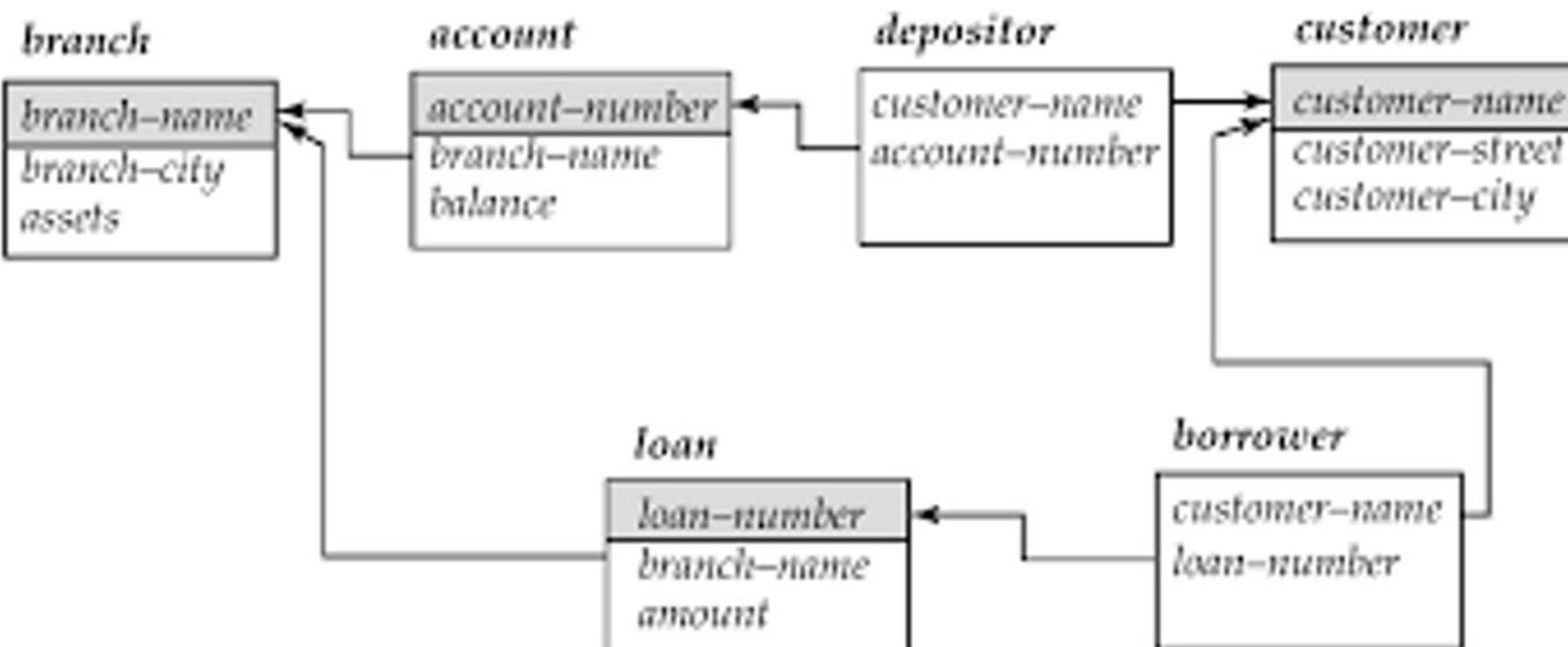


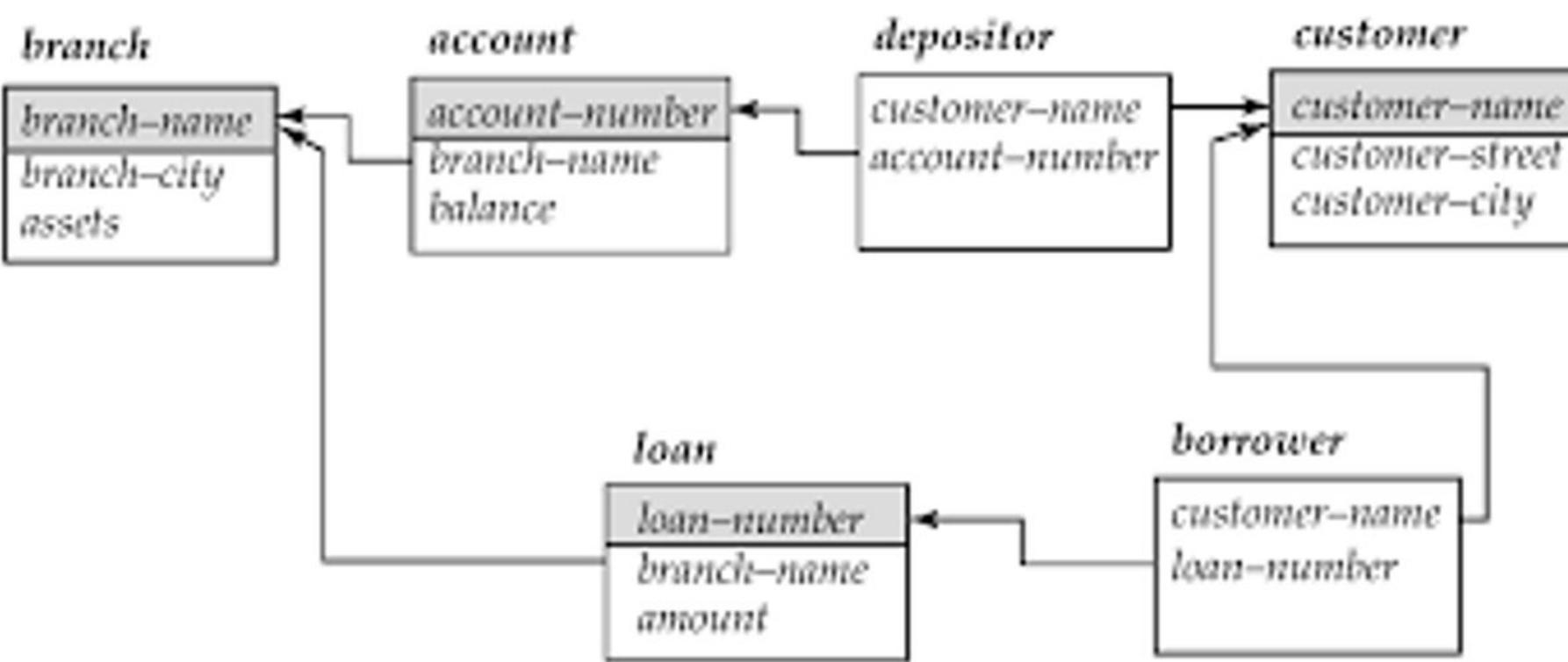
$$\{ t \mid \exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}]) \\ \wedge \exists u \in \text{loan} (u[\text{customer name}] = s[\text{customer name}]) \\ \wedge u[\text{branch name}] = \text{'Noida'} \}$$


Q Find the name of all the customers who have a loan or account or both at the bank?

{ $t \mid \exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}])$ }

$\vee \exists u \in \text{depositor} (t[\text{customer name}] = u[\text{customer name}]) \}$

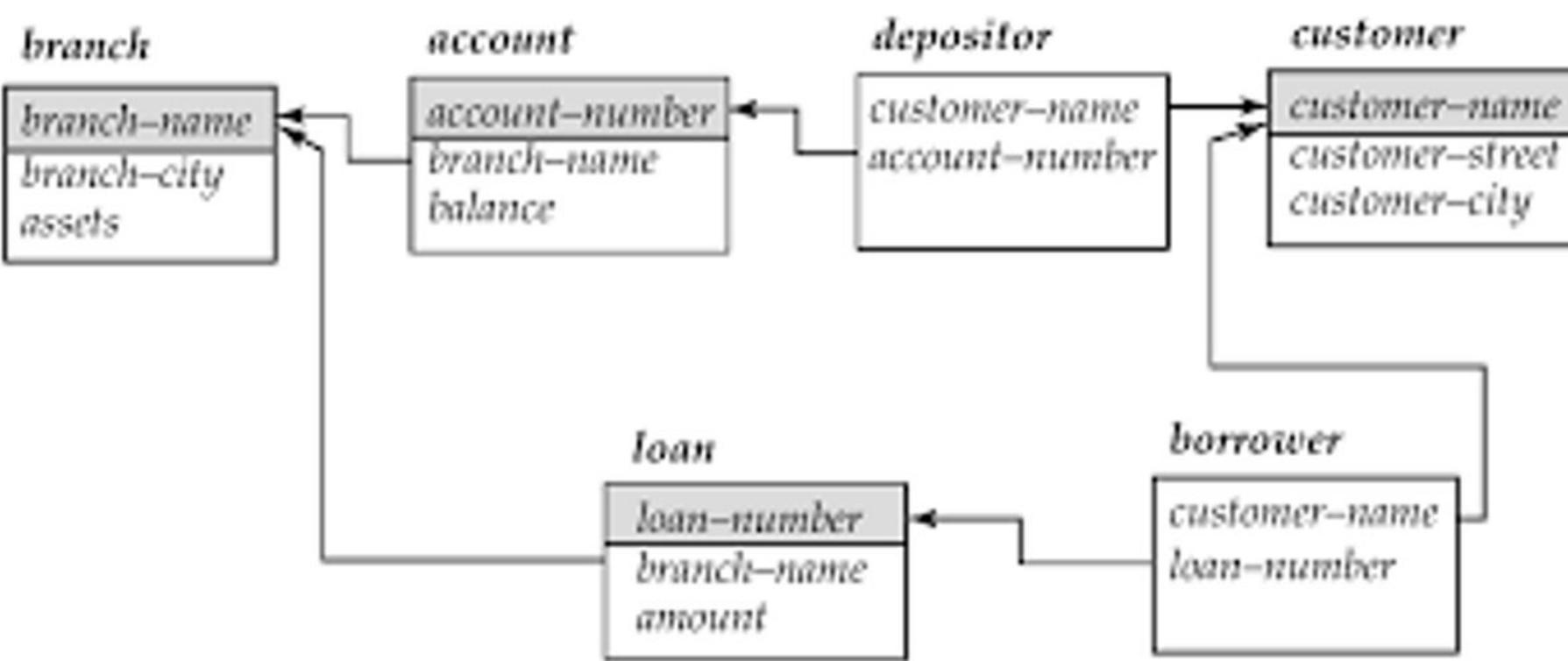


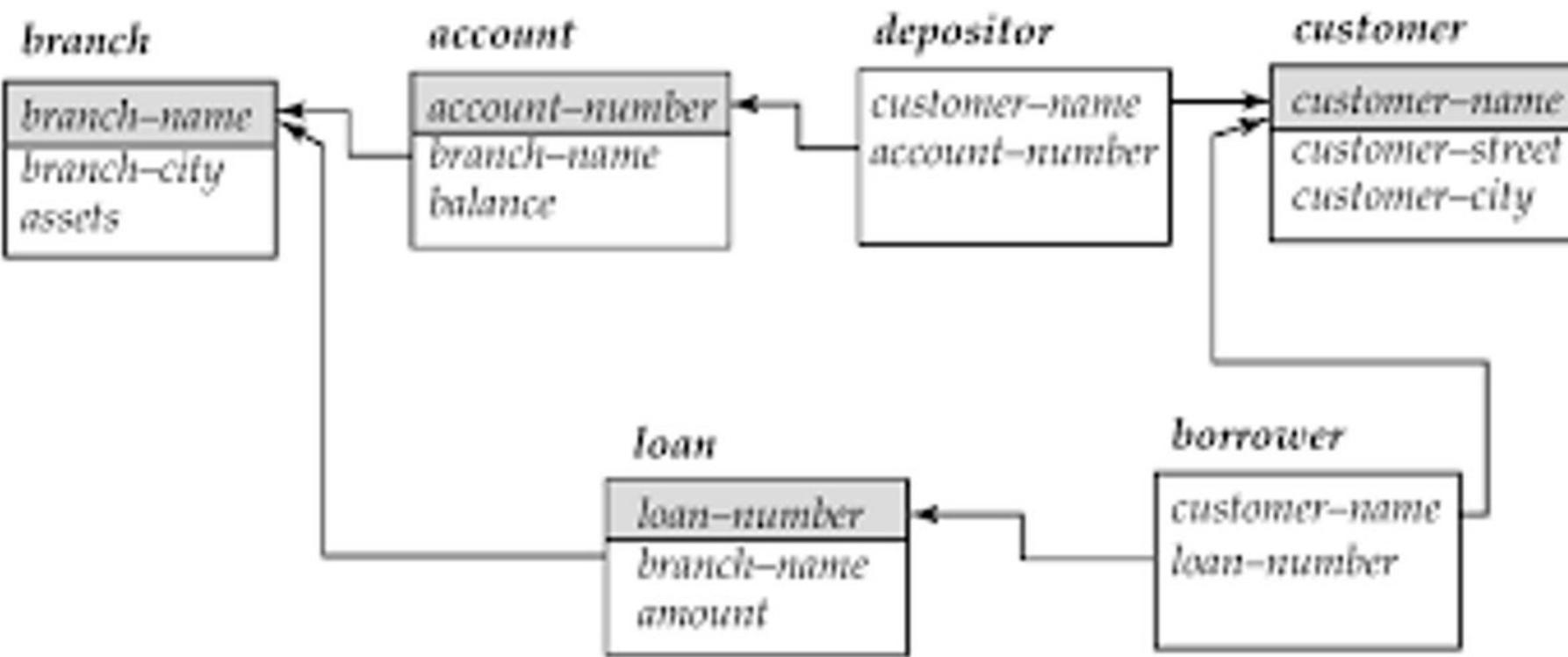
$$\{t \mid \exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}])\}$$
$$\vee \exists u \in \text{depositor} (t[\text{customer name}] = u[\text{customer name}]) \}$$


Q Find the name of all the customers who have a loan and account both at the bank?

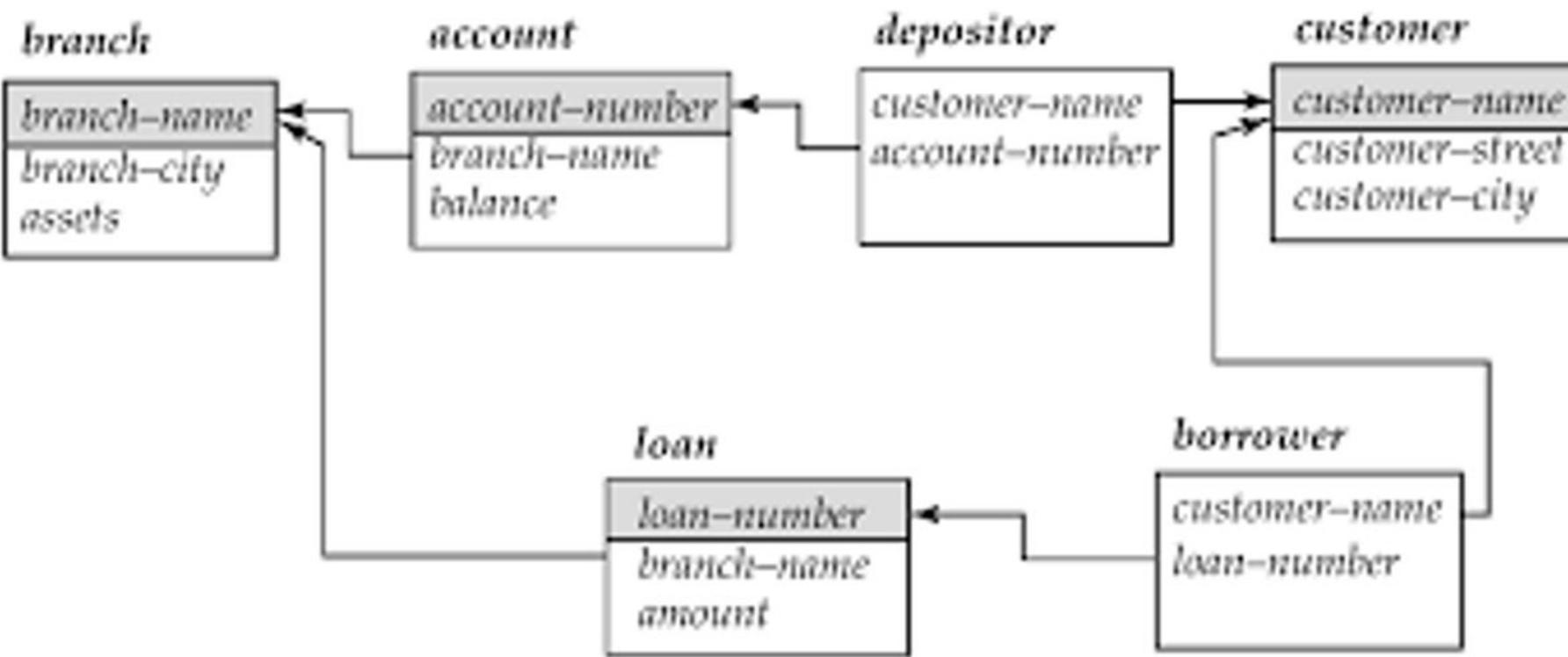
{t| $\exists s \in \text{borrower}$ ($t[\text{customer name}] = s[\text{customer name}]$)}

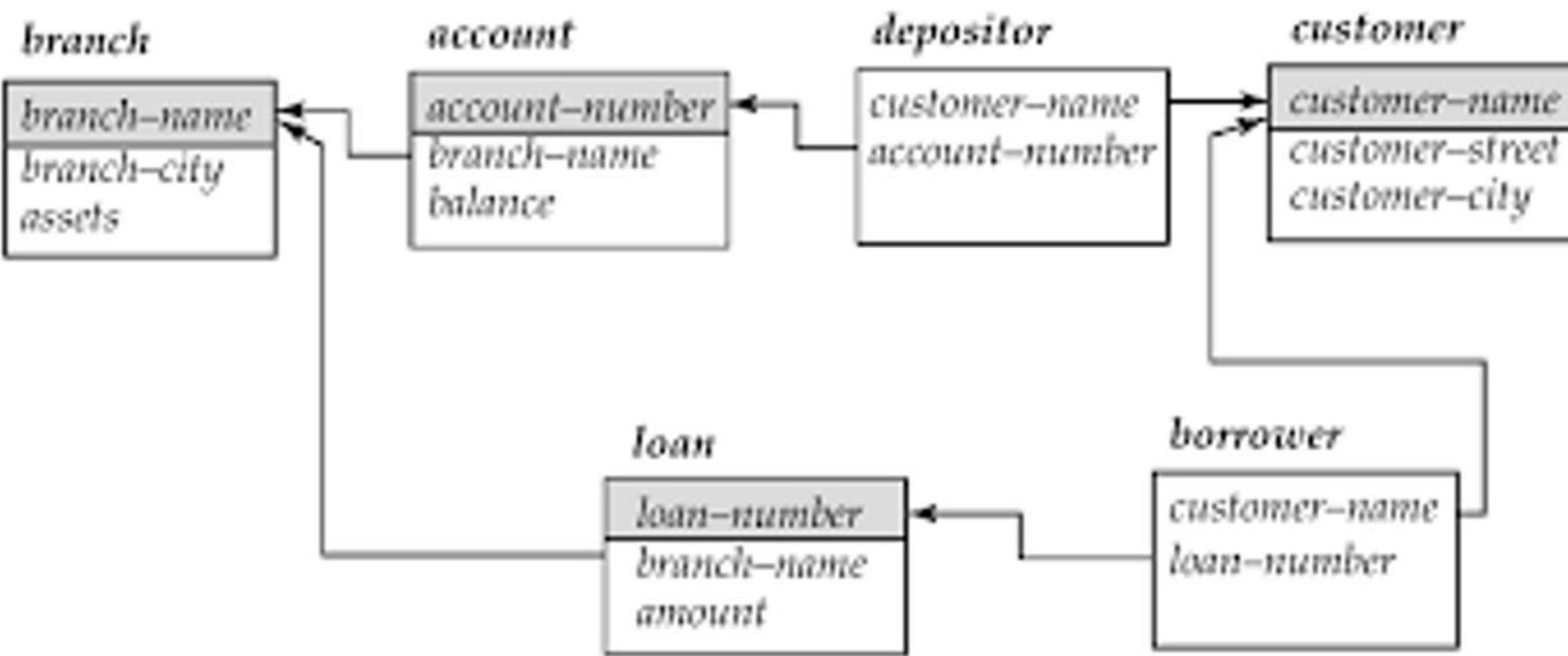
$\wedge \exists u \in \text{depositor}$ ($t[\text{customer name}] = u[\text{customer name}]$) }



$$\{ t \mid \exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}]) \\ \wedge \exists u \in \text{depositor} (t[\text{customer name}] = u[\text{customer name}]) \}$$


Q Find the name of all the customers who have a loan from the bank and do not have an account?

$$\{t \mid \exists u \in \text{depositor/borrower} (t[\text{customer name}] = u[\text{customer name}]) \\ \wedge \neg \exists s \in \text{borrower} (t[\text{customer name}] = s[\text{customer name}])\}$$


$$\{t \mid \exists u \in \text{depositor} \text{ borrower } (t[\text{customer name}] = u[\text{customer name}])$$
$$\wedge \neg \exists s \in \text{borrower} \text{ } (t[\text{customer name}] = s[\text{customer name}])\}$$


Expressive Power of Languages

Important Points to Remember

- The tuple relational calculus restricted to safe expressions is equivalent in expressive power to the basic relational algebra (with the operators \cup , $-$, \times , and $,$, but without the extended relational operations such as generalized projection and aggregation (G)).
- For every relational-algebra expression using only the basic operations, there is an equivalent expression in the tuple relational calculus, and for every tuple-relational-calculus expression, there is an equivalent relational algebra expression.
- Tuple relational calculus does not have any equivalent of the aggregate operation, but it can be extended to support aggregation. Extending the tuple relational calculus to handle arithmetic expressions is straightforward.

Safety of Expressions

- A tuple-relational-calculus expression may generate an infinite relation.

Example: $\{t \mid \neg(t \in \text{instructor})\}$

There are infinitely many tuples that are not in `instructor`. Most of these tuples contain values that do not even appear in the database. We do not want to allow such expressions.

- To help us define a restriction of the tuple relational calculus the concept of the **domain** of a tuple relational formula, P is introduced.
- The domain of P , denoted $\text{dom}(P)$, is the set of all values referenced by P .
- They include values mentioned in P itself, as well as values that appear in a tuple of a relation mentioned in P .

Example: $\text{dom}(t \in \text{instructor} \wedge t[\text{salary}] > 80000)$ is the set containing 80000 as well as the set of all values appearing in any attribute of any tuple in the instructor relation.

- . An expression $\{t \mid P(t)\}$ is safe if all values that appear in the result are values from $\text{dom}(P)$.
- . The expression $\{t \mid \neg(t \in \text{instructor})\}$ is not safe.
- . Safe expressions are guaranteed to have finite results.

Q Which of the rational calculus expression is not safe? (Gate-2001) (2 Marks)

- a) $\{t \mid \exists u \in R_1 (t[A] = u[A]) \wedge \neg \exists s \in R_2 (t[A] = s[A])\}$
- b) $\{t \mid \forall u \in R_1 (u[A] = "x" \Rightarrow \exists s \in R_2 (t[A] = s[A] \wedge s[A] = u[A]))\}$
- c) $\{t \mid \neg(t \in R_1)\}$
- d) $\{t \mid \exists u \in R_1 (t[A] = u[A]) \wedge \exists s \in R_2 (t[A] = s[A])\}$

Q Consider a database that has the relation schemas EMP (Empld, EmpName, DeptId), and DEPT (DeptName, DeptId). Note that the DeptId can be permitted to be NULL in the relation EMP. Consider the following queries on the database expressed in tuple relational calculus.

(Gate-2017) (1 Marks)

- I. $\{t \mid \exists u \in \text{EMP} (t[\text{EMPName}] = u[\text{EmpName}] \wedge \forall v \in \text{DEPT} (t[\text{DeptId}] \neq \text{DeptId}))\}$
- II. $\{t \mid \exists u \in \text{EMP} (t[\text{EMPName}] = u[\text{EmpName}] \wedge \exists v \in \text{DEPT} (t[\text{DeptId}] \neq \text{DeptId}))\}$
- III. $\{t \mid \exists u \in \text{EMP} (t[\text{EMPName}] = u[\text{EmpName}] \wedge \exists v \in \text{DEPT} (t[\text{DeptId}] = \text{DeptId}))\}$

Which of the above queries are safe?

- (A) I and II only
- (B) I and III only
- (C) II and III only
- (D) I, II, and III

Domain Relational Calculus

- Domain calculus differs from the tuple calculus in the type of variables used in formulas: rather than having variables range over tuples.
- The variable range over single values from domains of attributes. To form a relation of degree n for a query result, we must have n of these domain variables. One for each attribute.
- An expression of the domain calculus is of the form
- $(x_1, x_2, \dots, x_n \mid \text{COND}(x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2}, \dots, x_{n+m}))$
- Where x_1, x_2, \dots, x_n are domain variables that range over domains and COND is a condition of the domain relational calculus.

Student(Roll No, Name, Branch)

Q Find the details of all computer science students?

SQL: select * from student where branch = CSE

RA: $\{\sigma_{\text{branch} = \text{CSE}} (\text{Student})\}$

TRC: $\{t \mid \text{Student}(t) \cap t.\text{branch} = \text{CSE}\}$

DRC:

Student(Roll No, Name, Branch)

Q Find the details of all computer science students?

SQL: select * from student where branch = CSE

RA: $\{\sigma_{\text{branch} = \text{CSE}}(\text{Student})\}$

TRC: {t | Student(t) \cap t. branch = CSE}

DRC: {(Roll No, Name, Branch) | Student (Roll no, Name, Branch) \cap branch = CSE}

Student(Roll No, Name, Branch)

Q Find the Roll No of all computer science students?

SQL: select Roll No from student where branch = CSE

RA: $\{\prod_{\text{sname}} (\sigma_{\text{branch} = \text{CSE}} (\text{Student}))\}$

TRC: {t. Roll No | Student(t) \cap t. branch = CSE}

DRC:

Student(Roll No, Name, Branch)

Q Find the Roll No of all computer science students?

SQL: select Roll No from student where branch = CSE

RA: $\{\prod_{\text{sname}} (\sigma_{\text{branch} = \text{CSE}} (\text{Student}))\}$

TRC: {t. Roll No | Student(t) \cap t. branch = CSE}

DRC: {(Roll No) | Student (Roll no, Name, Branch) (Name, Branch) \cap branch = CSE}

Instructor(instructorID, name, dept name, and salary)

{ i, n, d, s | $i, n, d, s \in \text{instructor} \wedge s > 80000$ }

Instructor(instructorID, name, dept name, and salary)

Example: Find all details of instructors whose salary is greater than \$80,000

$$\{< i, n, d, s > \mid < i, n, d, s > \in \text{instructor} \wedge s > 80000\}$$

The Domain Relational Calculus

- . Uses domain variables that take on values from an attributes domain, rather than values for an entire tuple.
- . Closely related to the tuple relational calculus.
- . Domain relational calculus serves as the theoretical basis of the widely used QBE language

Formal Definition

- . An expression in the domain relational calculus is of the form $\{< x_1, x_2, \dots, x_n > \mid P(x_1, x_2, \dots, x_n)\}$ where x_1, x_2, \dots, x_n represent domain variables. P represents a formula composed of atoms.

- . An atom in the domain relational calculus has one of the following forms:
 - $\langle x_1, x_2, \dots, x_n \rangle \in r$, where r is a relation on n attributes and x_1, x_2, \dots, x_n are domain variables or domain constants.
 - $x \text{ op } y$, where x and y are domain variables and op is a comparison operator ($<$, $>$, \geq). We require that attributes x and y have domains that can be compared by .
 - $x \text{ op } c$, where x is a domain variable, op is a comparison operator, and c is a constant in the domain of the attribute for which x is a domain variable.

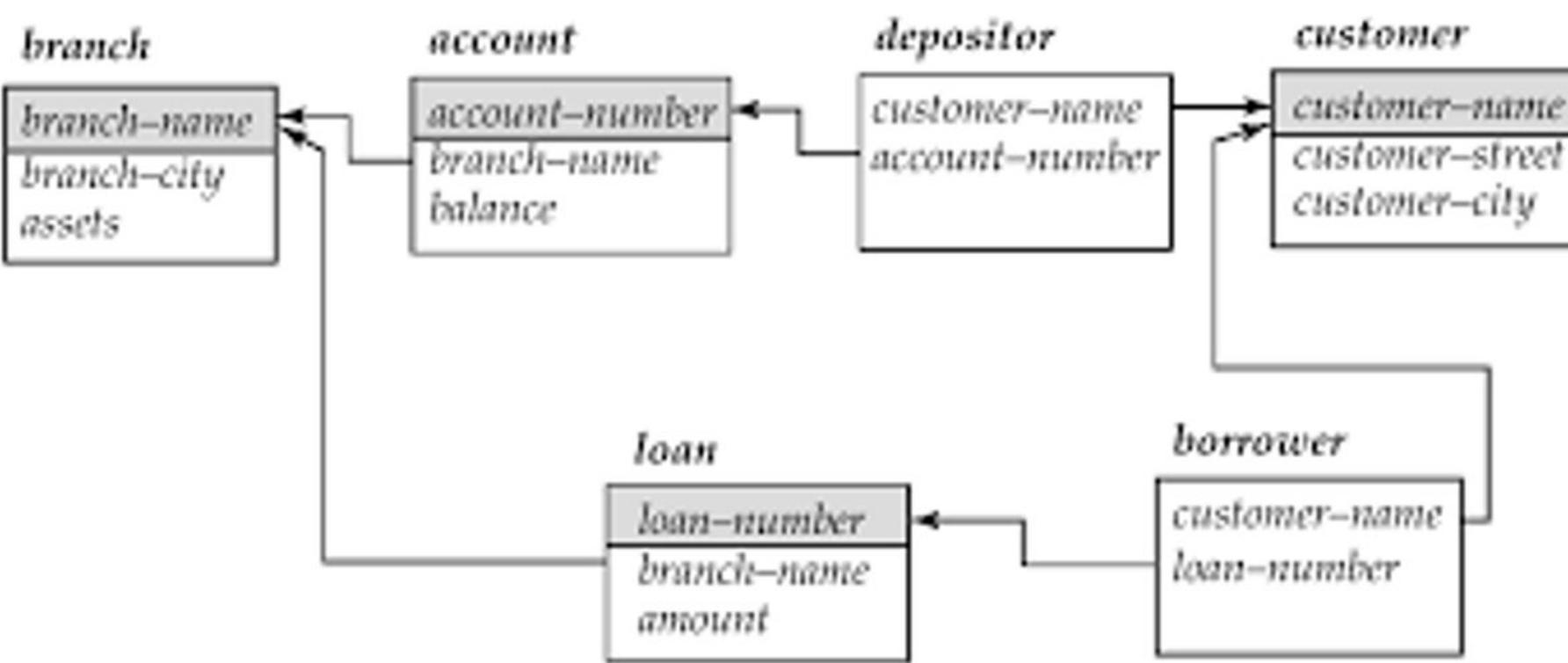
We build up formulae from atoms by using the following rules:

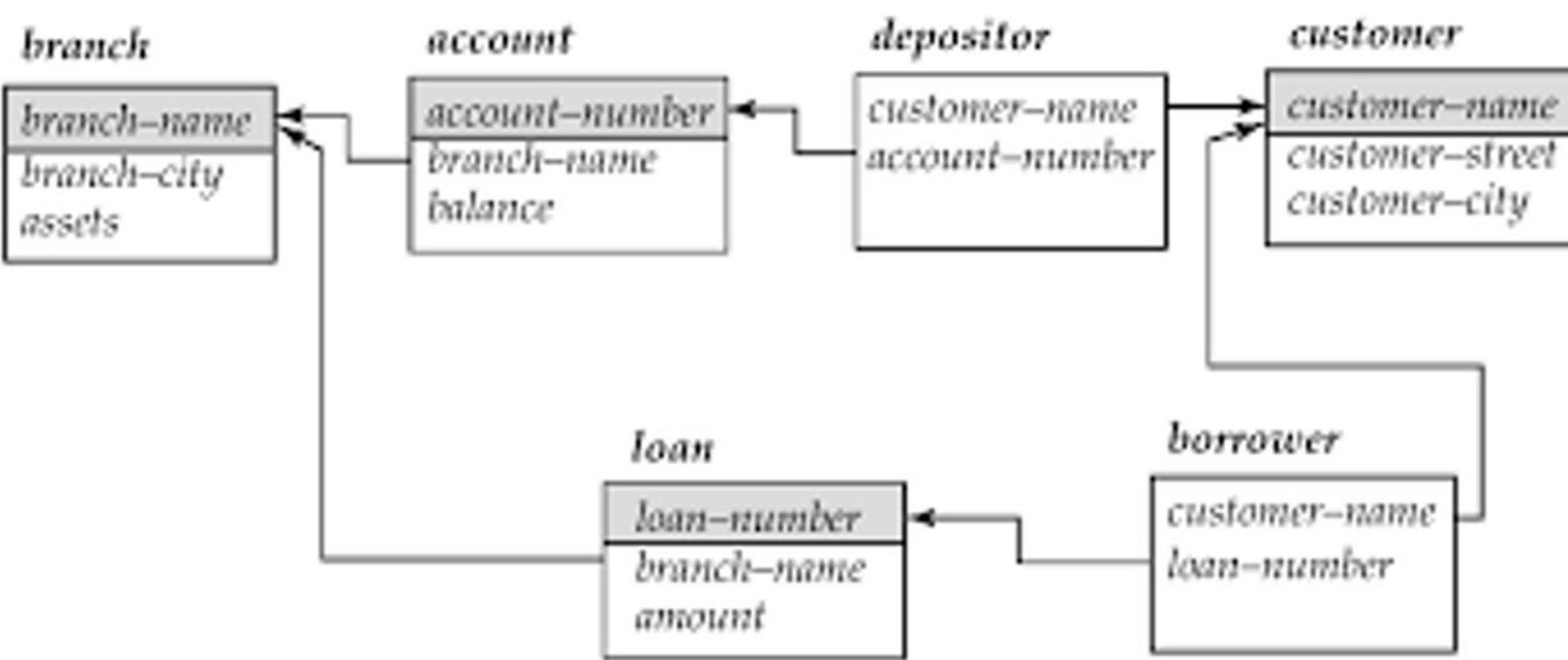
- An atom is a formula.
- If P_1 is a formula, then so are $\neg P_1$ and (P_1) .
- If P_1 and P_2 are formulae, then so are $P_1 \vee P_2$, $P_1 \wedge P_2$, and $P_1 \Rightarrow P_2$.
- If $P_1(x)$ is a formula in x , where x is a free domain variable, then $\exists x (P_1(x))$ and $\forall x (P_1(x))$ are also formulae.

As a notational shorthand, we write $\exists a, b, c (P(a, b, c))$ for $\exists a (\exists b (\exists c (P(a, b, c))))$.

Q Find the branch name, loan number and amount for loan of amount over 1200?

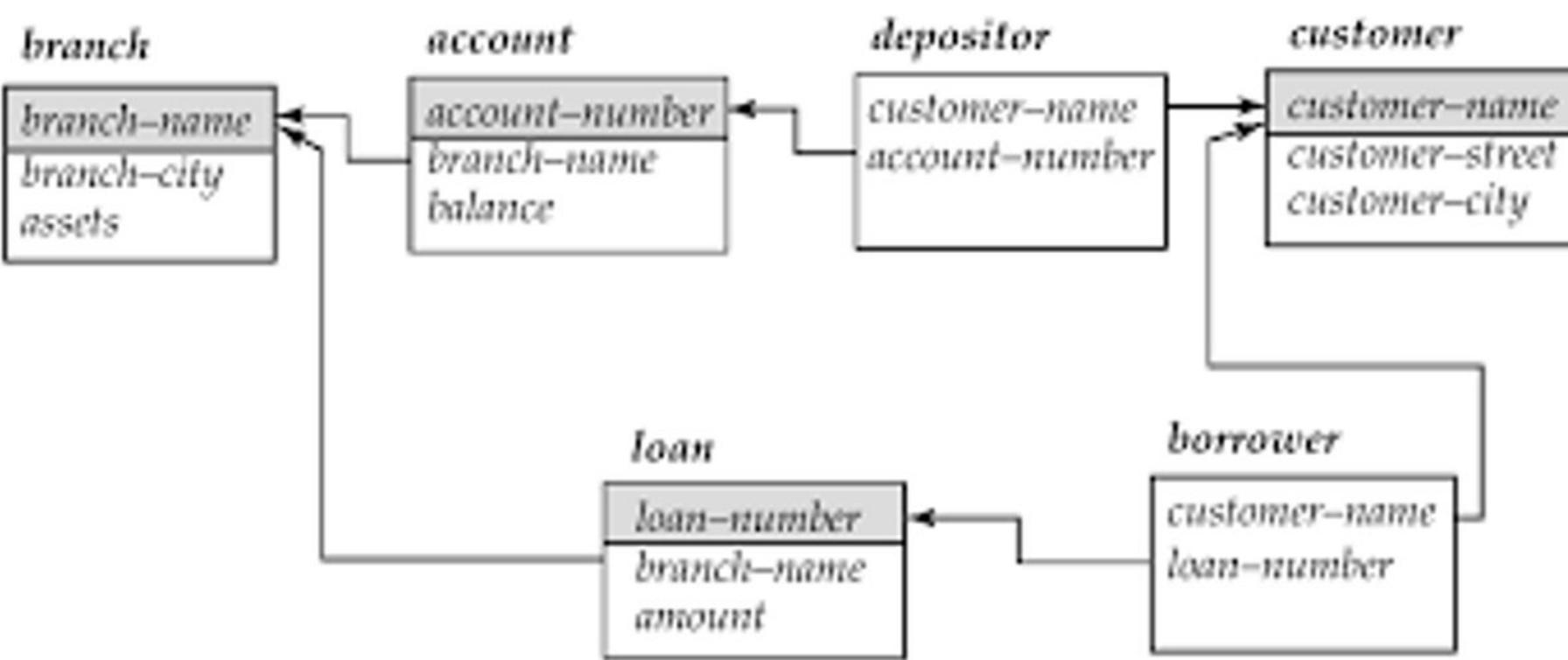
{ $\langle l, b, a \rangle \mid \langle l, b, a \rangle \in \text{loan} \wedge a > 1200$ }

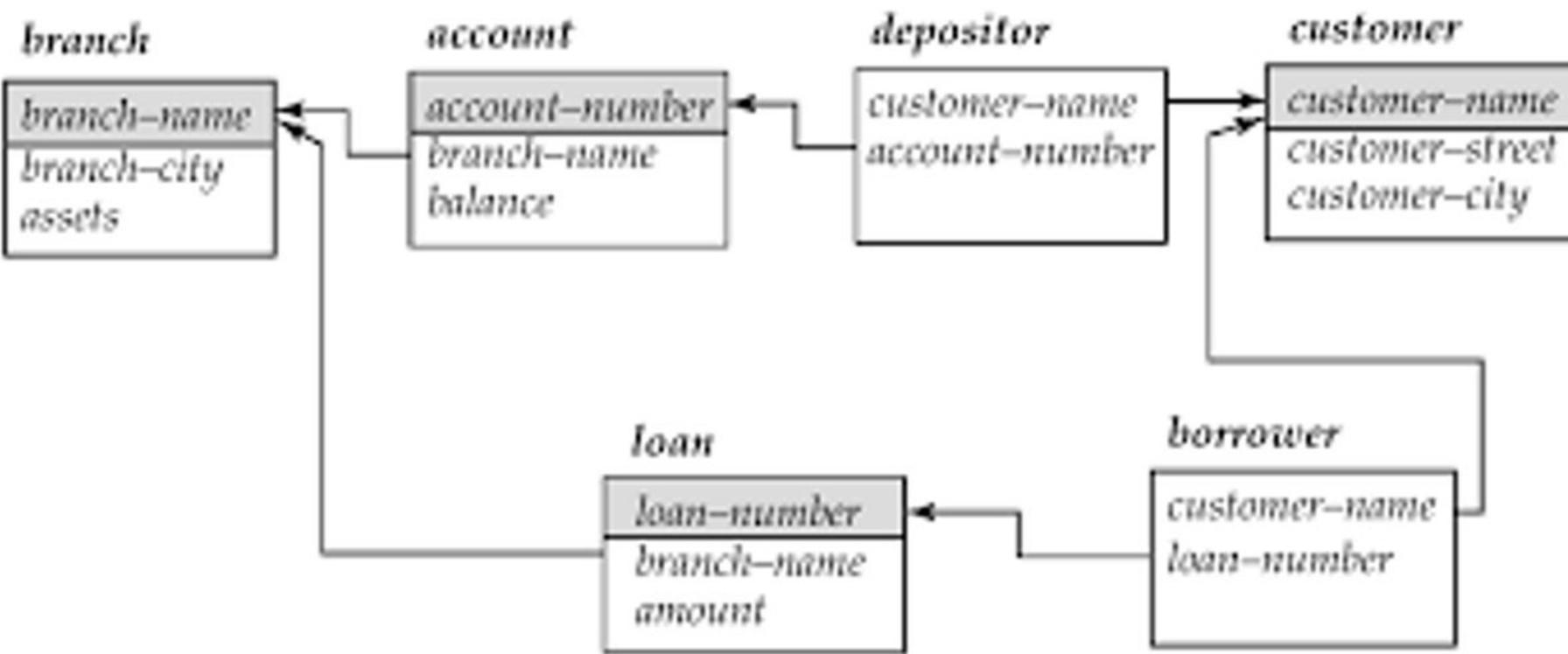


$$\{ \langle l, b, a \rangle \mid \langle l, b, a \rangle \in \text{loan} \wedge a > 1200 \}$$


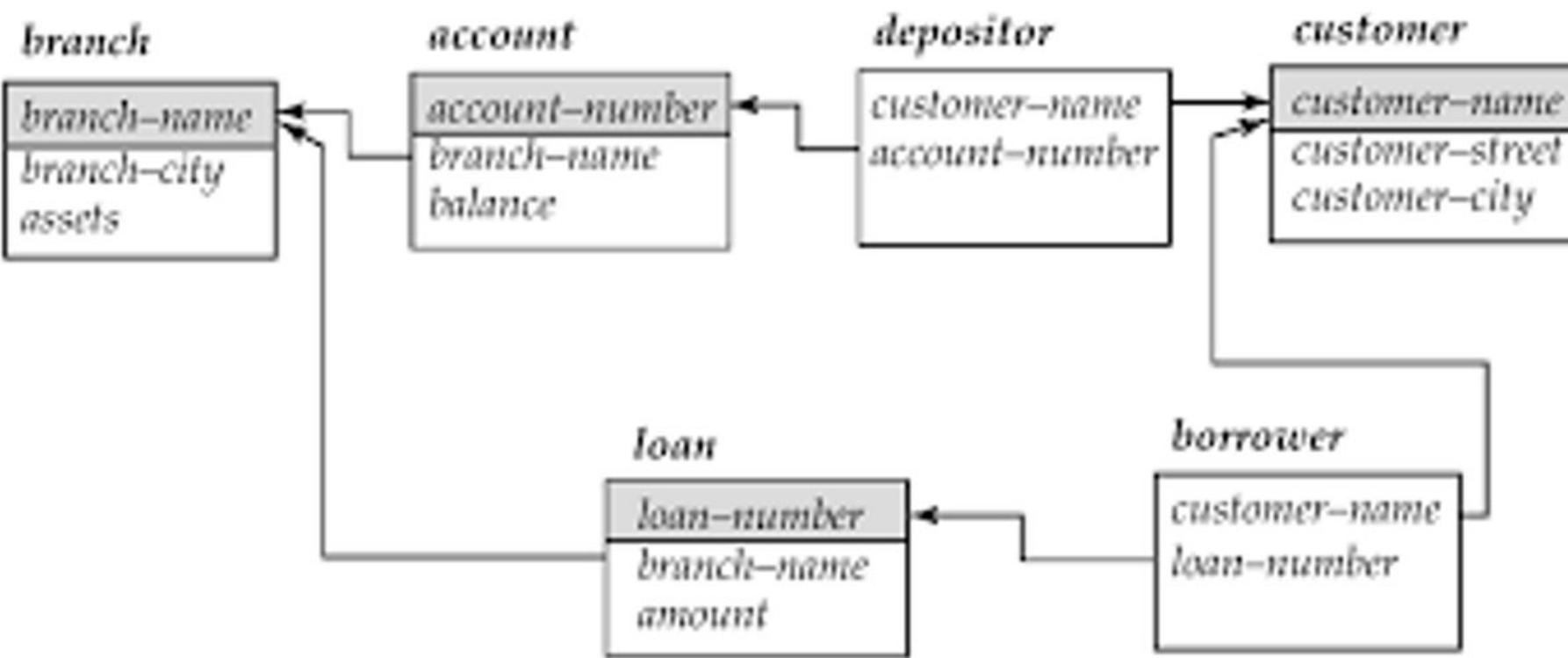
Q Find the loan number for each loan of amount over 1200?

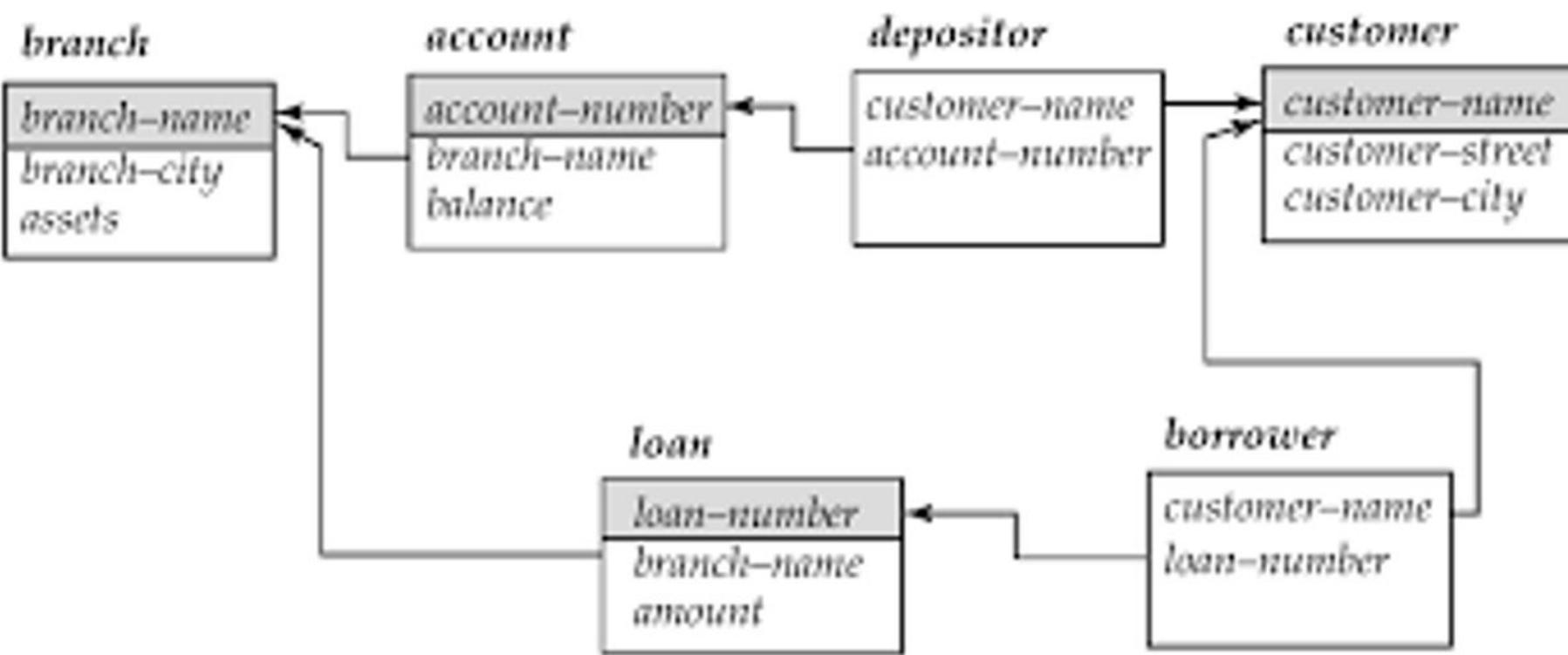
{ $\langle l \rangle \mid \exists_{b,a} \langle l, b, a \rangle \in \text{loan} \wedge a > 1200$ }



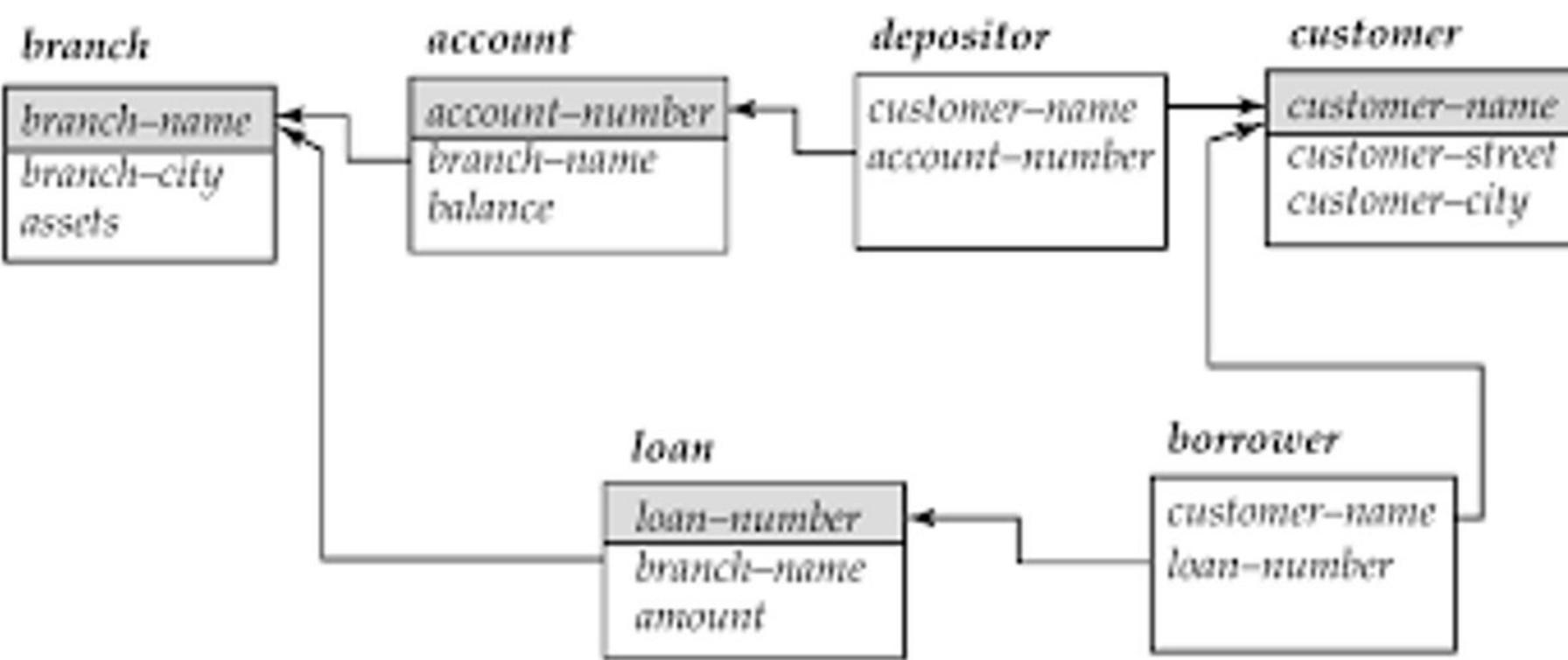
$$\{ \langle l \rangle \mid \exists_{b,a} \langle l, b, a \rangle \in \text{loan} \wedge a > 1200 \} \}$$


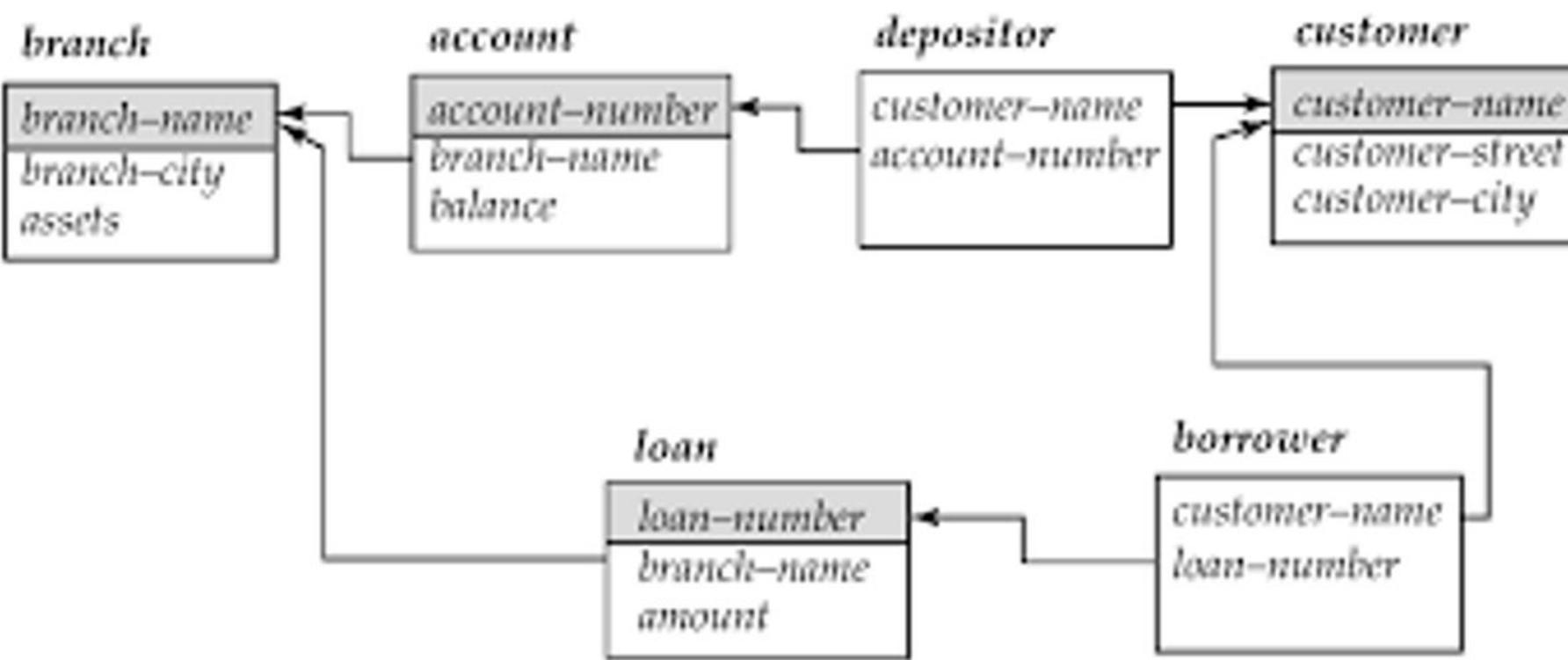
Q Find the name of all the customers who have a loan from Noida branch with loan amount?

$$\{ \langle c, a \rangle \mid \exists_l (\langle c, l \rangle \in \text{borrower} \wedge \exists_b (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{'Noida'})) \}$$


$$\{ \langle c, a \rangle \mid \exists_l (\langle c, l \rangle \in \text{borrower} \wedge \exists_b (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{'Noida'})) \}$$


Q Find the name of all the customers who have a loan or account or both at the bank?

$$\{ \langle c \rangle \mid \exists_l (\langle c, l \rangle \in \text{borrower} \wedge \exists_{b,a} (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{'Noida'})) \}$$
$$\vee \exists_a (\langle c, a \rangle \in \text{depositor} \wedge \exists_{b,a} (\langle a, bn, bal \rangle \in \text{account} \wedge bn = \text{'Noida'})) \}$$


$$\{ \langle c \rangle \mid \exists_l (\langle c, l \rangle \in \text{borrower} \wedge \exists_{b,a} (\langle l, b, a \rangle \in \text{loan} \wedge b = \text{'Noida'})) \\ \vee \exists_a (\langle c, a \rangle \in \text{depositor} \wedge \exists_{bn, bal} (\langle a, bn, bal \rangle \in \text{account} \wedge bn = \text{'Noida'})) \}$$


Safety of Expressions

- Similar to TRC we can have an unsafe expression that may generate infinite relations.
- An expression such as

$$\{ \langle i, n, d, s \rangle \mid \neg (\langle i, n, d, s \rangle \in \text{instructor}) \}$$

is unsafe, because it allows values in the result that are not in the domain of the expression.

- . We say that an expression $\{<x_1, x_2, \dots, x_n> \mid P(x_1, x_2, \dots, x_n)\}$ is safe if all of the following hold:
 1. All values that appear in tuples of the expression are values from $\text{dom}(P)$.
 2. For every “there exists” sub formula of the form $\exists x (P_1(x))$, the sub formula is true if and only if there is a value x in $\text{dom}(P_1)$ such that $P_1(x)$ is true.
 3. For every “for all” sub formula of the form $\forall x (P_1(x))$, the sub formula is true if and only if $P_1(x)$ is true for all values x from $\text{dom}(P_1)$.

Expressive Power of Languages

- When the domain relational calculus is restricted to safe expressions, it is equivalent in expressive power to the tuple relational calculus restricted to safe expressions.
- All three of the following are equivalent:
 - The basic relational algebra (without the extended relational-algebra operations)
 - The tuple relational calculus restricted to safe expressions
 - The domain relational calculus restricted to safe expressions
- Domain relational calculus also does not have any equivalent of the aggregate operation, but it can be extended to support aggregation, and extending it to handle arithmetic expressions is straightforward.

Important difference between Tuple Relational Calculus and Domain Relational Calculus.

- In the tuple calculus, when we write $\exists s$ for some tuple variable s , we bind it immediately to a relation by writing $\exists s \in r$.
- However, when we write $\exists n$ in the domain calculus, n refers not to a tuple, but rather to a domain value. Thus, the domain of variable n is unconstrained until the sub formula $< i, n, d, s > \in \text{instructor}$ constrains n to instructor names that appear in the instructor relation.

Q Consider the following relational schema. (Gate-2013) (2 Marks)

Students (rollno: integer, sname: string)

Courses (courseno: integer, cname: string)

Registration (rollno: integer, courseno: integer, percent: real)

Which of the following queries are equivalent to this query in English?

“Find the distinct names of all students who score more than 90% in the course numbered 107”

i) $\text{SELECT DISTINCT } S.\text{sname} \text{ FROM Students as } S, \text{Registration as } R \text{ WHERE}$

$R.\text{rollno} = S.\text{rollno} \text{ AND } R.\text{courseno} = 107 \text{ AND } R.\text{percent} > 90$

ii) $\prod_{\text{sname}} (\sigma_{\text{courseno} = 107} \wedge \text{percent} > 90 \text{ (Registration} \bowtie \text{Students)})$

iii) $\{T | \exists S \in \text{Students}, \exists R \in \text{Registration} (S.\text{rollno} = R.\text{rollno})$

$\wedge R.\text{courseno} = 107 \wedge R.\text{percent} > 90 \wedge T.\text{sname} = S.\text{sname}\}$

iv) $\{\langle SN \rangle | \exists SR \exists RP (\langle SR, SN \rangle \in \text{Students} \wedge \langle SR, 107, RP \rangle \in \text{Registration} \wedge RP > 90)\}$

a) I, II, III and IV

b) I, II and III only

c) I, II and IV only

d) II, III and IV only

(I) $\text{SELECT DISTINCT } S.\text{sname}$
FROM Students as S, Registration as R
WHERE R.rollno=S.rollno AND R.courseno=107 AND R.percent >90

(II) $\prod_{\text{sname}} (\sigma_{\text{courseno}=107 \wedge \text{percent}>90} (\text{Registration} \bowtie \text{Students}))$

(III) $\{T \mid \exists S \in \text{Students}, \exists R \in \text{Registration} \ (S.\text{rollno}=R.\text{rollno} \wedge$
 $R.\text{courseno}=107 \wedge R.\text{percent}>90 \wedge T.\text{sname}=S.\text{sname})\}$

(IV) $\{\langle S_N \rangle \mid \exists S_R \exists R_P \ (\langle S_R, S_N \rangle \in \text{Students} \wedge \langle S_R, 107, R_P \rangle \in \text{Registration} \wedge R_P > 90)\}$

GATE (2008)

Which of the following tuple relational calculus expression(s) is/are equivalent to $\forall t \in r(P(t))$?

- I. $\neg \exists t \in r(P(t))$
- II. $\exists t \notin r(P(t))$
- III. $\neg \exists t \in r(\neg P(t))$
- IV. $\exists t \notin r(\neg P(t))$

- (A) I only (B) II only**
(C) III only (D) III and IV only

Q Consider the relation employee (name, sex, supervisor Name) with name as the key. supervisor Name gives the name of the supervisor of the employee under consideration. What does the following Tuple Relational Calculus query produce? **(Gate-2007) (2 Marks)**

{e.name | employee (e) \wedge
 $(\forall x)[\neg \text{employee}(x) \vee x.\text{supervisorName} \neq e.\text{name} \vee x.\text{sex} = \text{"male"}]$ }

- (A) Names of employees with a male supervisor.
- (B) Names of employees with no immediate male subordinates.
- (C) Names of employees with no immediate female subordinates.
- (D) Names of employees with a female supervisor.

Q With regard to the expressive power of the formal relational query languages, which of the following statements is true? **(CS-2002)**

- (A)** Relational algebra is more powerful than relational calculus
- (B)** Relational algebra has the same power as relational calculus
- (C)** Relational algebra has the same power as safe relational calculus
- (D)** None of the above

Q. Consider two relations describing teams and players in a sports league:

$\text{teams}(\text{tid}, \text{tname})$: tid , tname are team-id and team-name, respectively

$\text{players}(\text{pid}, \text{pname}, \text{tid})$: pid , pname , and tid denote player-id, playername and the team-id of the player, respectively.

Which ONE of the following tuple relational calculus queries returns the name of the players who play for the team having tname as 'MI'? **(Gate 2025)**

- A) $\{p.\text{pname} \mid p \in \text{players} \wedge \exists t(t \in \text{teams} \wedge p.\text{tid} = t.\text{tid} \wedge t.\text{tname} = 'MI')\}$
- B) $\{p.\text{pname} \mid p \in \text{teams} \wedge \exists t(t \in \text{players} \wedge p.\text{tid} = t.\text{tid} \wedge t.\text{tname} = 'MI')\}$
- C) $\{p.\text{pname} \mid p \in \text{players} \wedge \exists t(t \in \text{teams} \wedge t.\text{tname} = 'MI')\}$
- D) $\{p.\text{pname} \mid p \in \text{teams} \wedge \exists t(t \in \text{players} \wedge t.\text{tname} = 'MI')\}$

TRANSACTION

- Why we study 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

T_1
Read(A)
$A = A - 100$
Write(A)
Read(B)
$B = B + 100$
Write(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 $(A + B) == \text{after } (A + B)$ ”

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



What is transaction

- 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’.

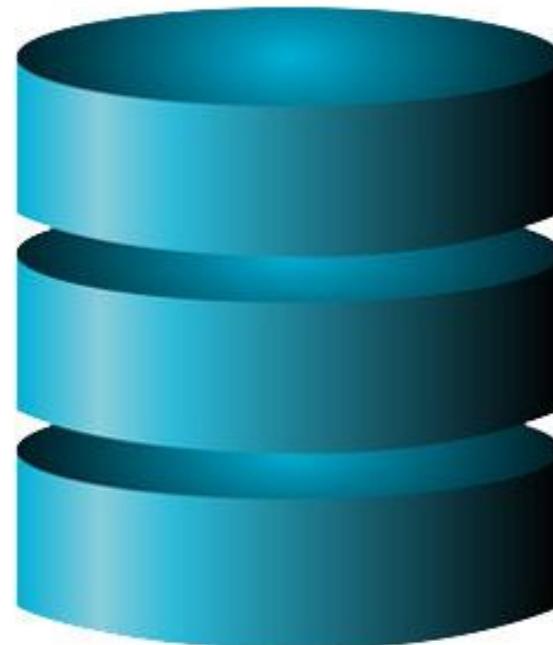
T ₁
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)

- 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.

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.

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



- 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:

A = Atomicity

C = Consistency

I = Isolation

D = Durability

- **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

- **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*.

- **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.

Q NOT a part of the ACID properties of database transactions? (GATE- 2016) (1 Marks)

(a) Atomicity

(b) Consistency

(c) Isolation

(d) Deadlock-freedom

**Q Consider the following transaction involving two bank accounts x and y. (GATE - 2015)
(1 Marks)**

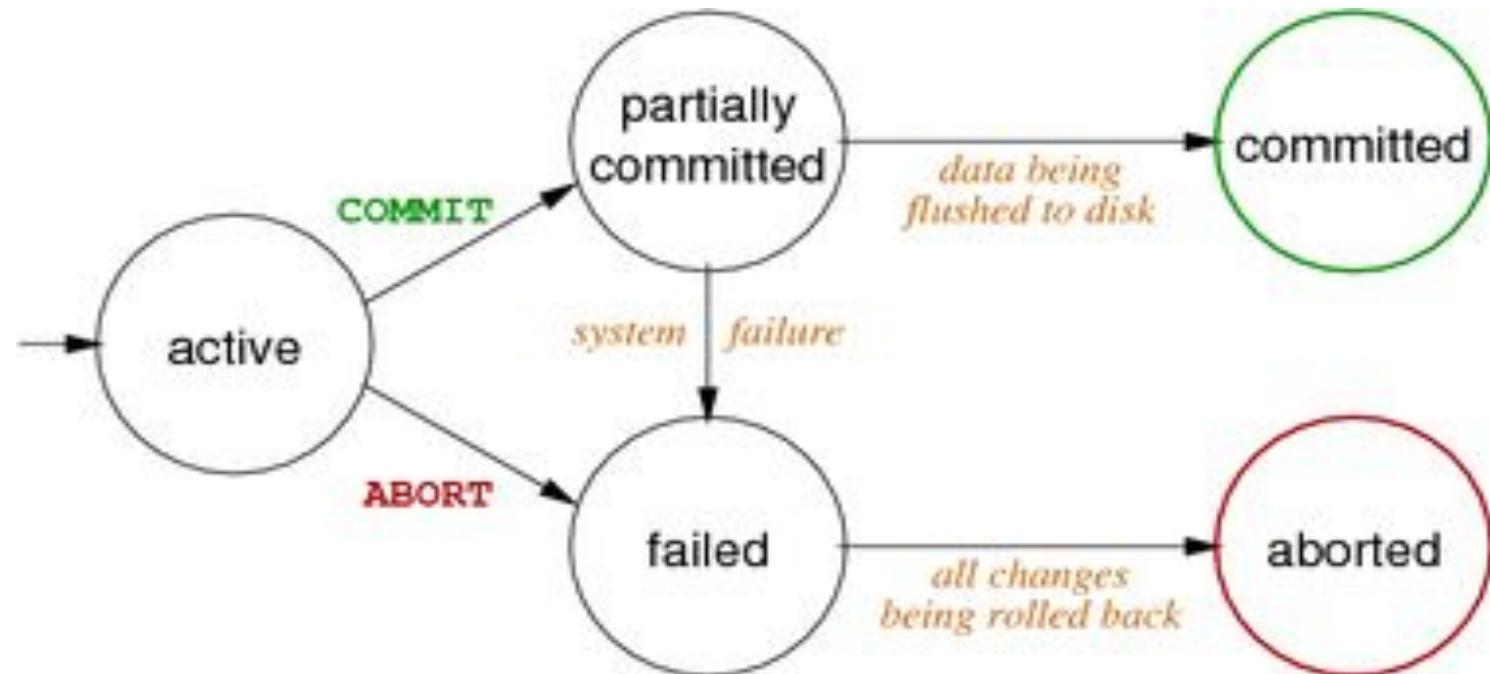
```
read(x);  
x: = x - 50;  
write(x);  
read(y);  
y: = y + 50;  
write(y)
```

The constraint that the sum of the accounts x and y should remain constant is that of
(A) Atomicity (B) Consistency

(C) Isolation (D) Durability

Q. Once the DBMS informs the user that a transaction has been successfully completed, its effect should persist even if the system crashes before all its changes are reflected on disk. This property is called (Gate 2024 CS) (1 Marks)(MCQ)

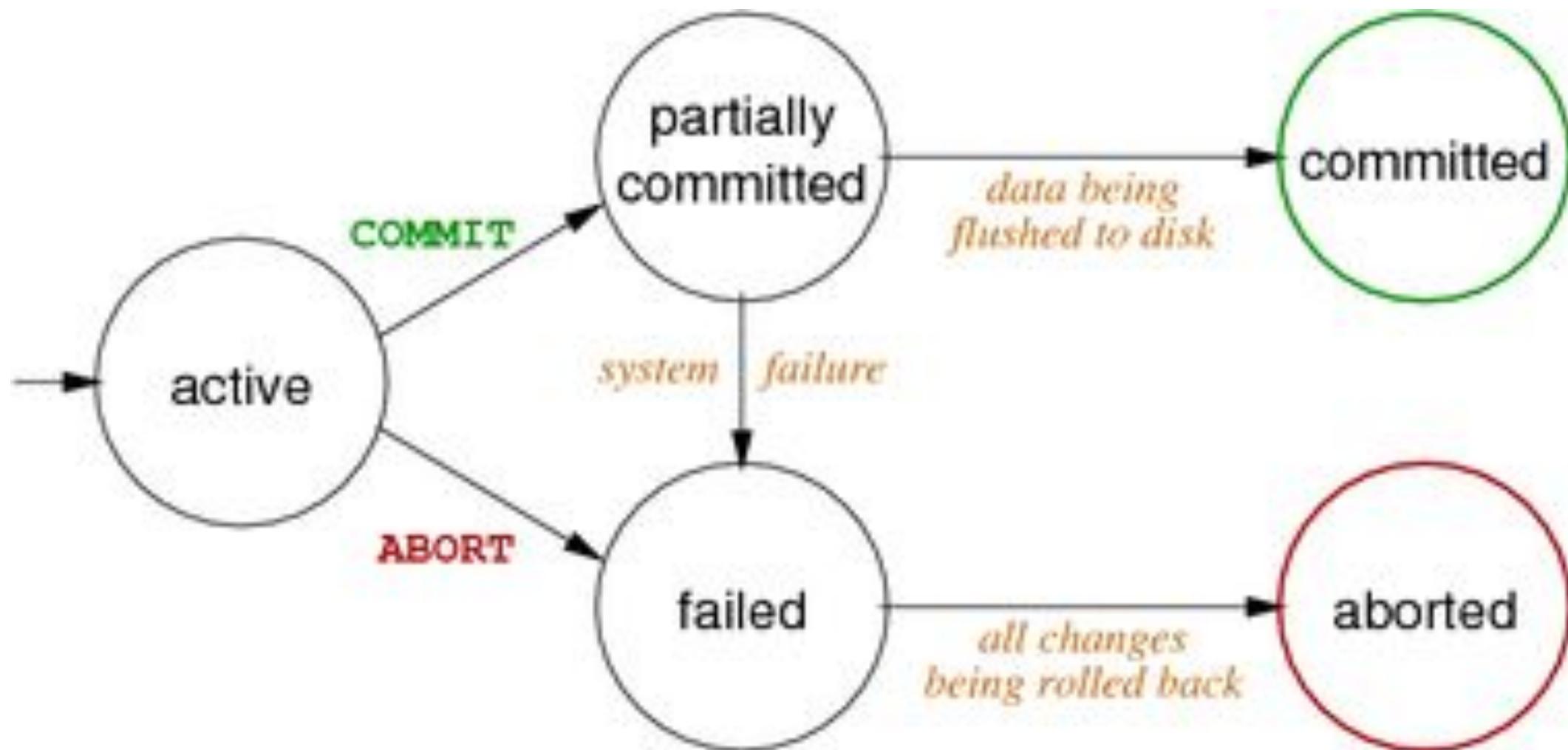
- (a) durability
- (b) atomicity
- (c) consistency
- (d) Isolation



Q. An audit of a banking transactions system has found that on an earlier occasion, two joint holders of account A attempted simultaneous transfers of Rs. 10000 each from account A to account B. Both transactions read the same value, Rs. 11000, as the initial balance in A and were allowed to go through. B was credited Rs. 10000 twice. A was debited only once and ended up with a balance of Rs. 1000. Which of the following properties is/are certain to have been violated by the system? **(Gate 2025)**

- A)** Atomicity
- B)** Consistency
- C)** Isolation
- D)** Durability

Transaction states



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 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.

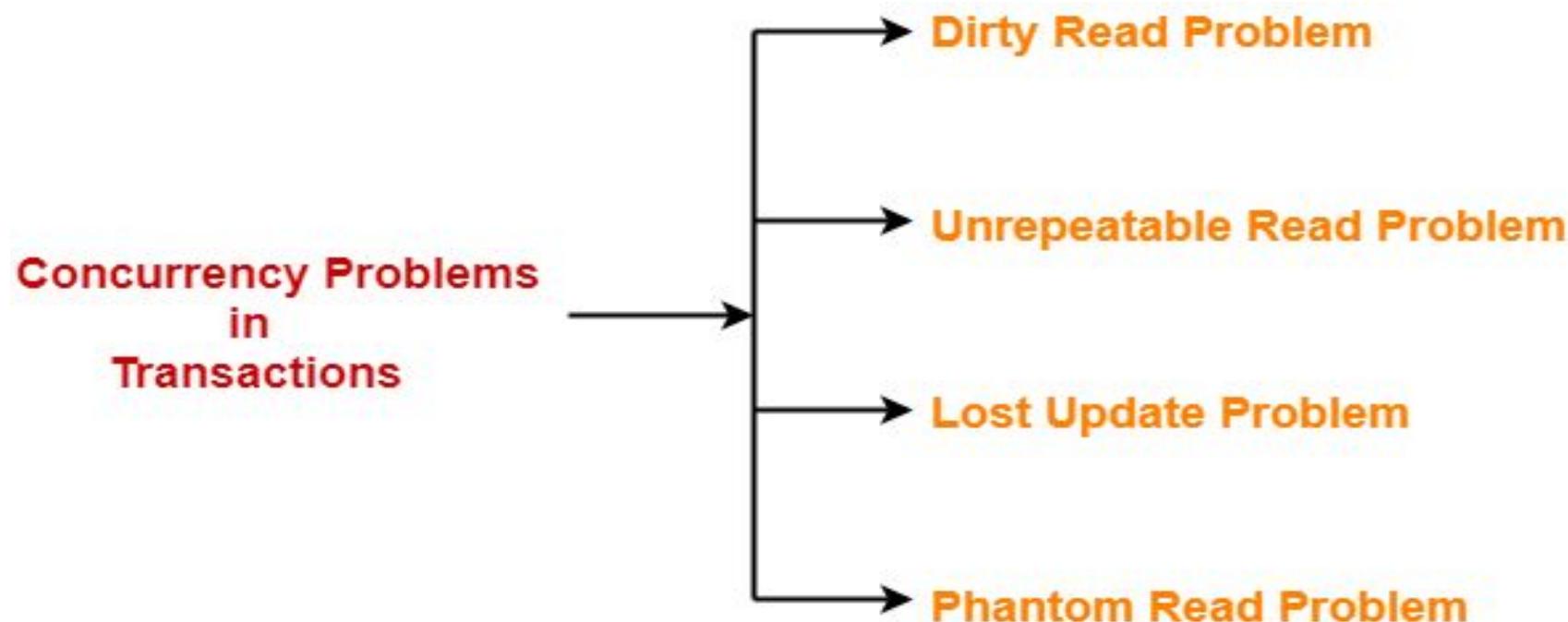
Why we need concurrent execution

- 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 statutes of the system will be inconsistent.



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? (GATE – 2008) (1 Marks)

- (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(B) write(B)
read(A)	read(A) write(A)

- 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

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)

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
	R(A)
W(A)	

T_1	T_2
R(A)	
	R(A)

T_1	T_2
	R(A)
R(A)	

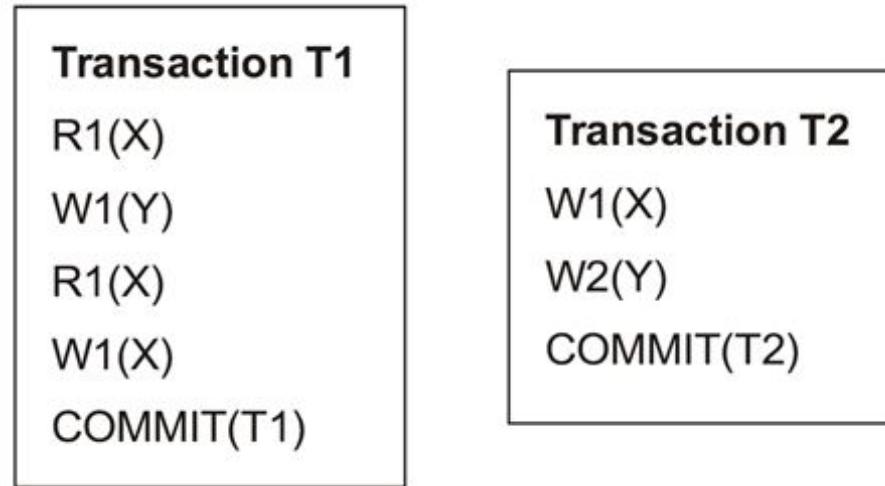
T_1	T_2
	W(A)
W(A)	

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. Consider the database transactions T1 and T2, and data items X and Y. Which of the schedule(s) is/are conflict serializable?(Gate 2025)



- A) R1(X), W2(X), W1(Y), W2(Y), R1(X), W1(X), COMMIT(T2), COMMIT(T1)
- B) W2(X), R1(X), W2(Y), W1(Y), R1(X), COMMIT(T2), W1(X), COMMIT(T1)
- C) R1(X), W1(Y), W2(X), W2(Y), R1(X), W1(X), COMMIT(T1), COMMIT(T2)
- D) W2(X), R1(X), W1(Y), W2(Y), R1(X), COMMIT(T2), W1(X), COMMIT(T1)

Conflict equivalent – if one schedule can be converted to another schedule by swapping of non-conflicting instruction then they are called conflict equivalent schedule.

T_1	T_2
$R(A)$	
$A=A-50$	
	$R(B)$
	$B=B+50$
$R(B)$	
$B=B+50$	
	$R(A)$
	$A=A+10$

T_1	T_2
	$R(B)$
	$B=B+50$
$R(A)$	
$A=A-50$	
$R(B)$	
$B=B+50$	
	$R(A)$
	$A=A+10$

Q Consider a schedule of transactions T_1 and T_2 :

Here, RX stands for “Read(X)” and WX stands for “Write(X)”. Which one of the following schedules is conflict equivalent to the above schedule? (Gate-2020) (2 Marks)

a)

S	
T_1	T_2
	RB
	WB
	RD
RA	
RC	
WD	
WB	
	WC
Commit	
	Commit

b)

S	
T_1	T_2
RA	
RC	
WD	
WB	
	RB
	WB
	RD
	WC
Commit	
	Commit

c)

S	
T_1	T_2
RA	
RC	
WD	
RB	
WB	
RD	
WC	
Commit	
	Commit

d)

S	
T_1	T_2
	RB
	WB
	RD
	WC
RA	
RC	
WD	
WB	
Commit	
	Commit

S	
T_1	T_2
RA	
RB	
WB	
RC	
RD	
WD	
WC	
WB	
Commit	
	Commit

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? **(GATE - 2021) (2 Marks)**

- 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

Procedure for determining conflict serializability of a schedule

- It can be determined using PRECEDENCE GRAPH method:
- A precedence graph consists of a pair $G(V, E)$
- V = set of vertices consisting of all the transactions participating in the schedule.
- E = set of edges consists of all edges $T_i \square T_j$, for which one of the following conditions holds:
 - T_i executes write(Q) before T_j executes read(Q)
 - T_i executes read(Q) before T_j executes write(Q)
 - T_i executes write(Q) before T_j executes write(Q)

- If an edge $T_i \square T_j$ exists in the precedence graph, then in any serial schedule S' equivalent to S , T_i must appear before T_j .
- **If the precedence graph for S has no cycle, then schedule S is conflict serializable, else it is not.** This cycle detection can be done by cycle detection algorithms, one of them based on depth first search takes $O(n^2)$ time.
- The serializability order of transactions of equivalent serial schedule can be determined using **topological order** in a precedence graph.

Q Consider the following four schedules due to three transactions (indicated by the subscript) using read and write on a data item X, denoted by $r(X)$ and $w(X)$ respectively. Which one of them is conflict serializable? **(NET-NOV-2018) (GATE - 2014) (2 Marks)**

$S_1: r_1(X); r_2(X); w_1(X); r_3(X); w_2(X)$ $S_2: r_2(X); r_1(X); w_2(X); r_3(X); w_1(X)$

$S_3: r_3(X); r_2(X); r_1(X); w_2(X); w_1(X)$ $S_4: r_2(X); w_2(X); r_3(X); r_1(X); w_1(X)$

(a) S_1

(b) S_2

(c) S_3

(d) S_4

Q Consider the following transactions with data items P and Q initialized to zero: (GATE-2012) (2 Marks)

T_1 : read (P);

 read (Q);

 if P = 0 then Q: = Q + 1;

 write (Q);

T_2 : read (Q);

 read (P);

 if Q = 0 then P: = P + 1;

 write (P);

Any non-serial interleaving of T_1 and T_2 for concurrent execution leads to

- (A) A serializable schedule
- (B) A schedule that is not conflict serializable
- (C) A conflict serializable schedule
- (D) A schedule for which a precedence graph cannot be drawn

Q Consider two transactions T_1 and T_2 which form schedules S_1, S_2, S_3 and S_4 as follows: -

Which of the above schedules is conflicts serializable? (GATE - 2009) (2 Marks)

- a) Only S_1
- b) Both S_1 and S_2
- c) Both S_1 and S_4
- d) Both S_3 and S_4

$T_1: R_1[A], W_1[A], W_1[B]$

$T_2: R_2[A], R_2[B], W_2[B]$

$S_1: R_1[A], R_2[A], R_2[B], W_1[A], W_2[B], W_1[B]$

$S_2: R_1[A], R_2[A], R_2[B], W_1[A], W_1[B], W_2[B]$

$S_3: R_2[A], R_1[A], R_2[B], W_1[A], W_1[B], W_2[B]$

$S_4: R_1[A], R_2[A], R_2[B], W_1[B], R_2[B], W_2[B]$

Q Consider the following schedule for transactions T_1 , T_2 and T_3 : (GATE – 2010) (2 Marks)

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)	
			Read (Y)
			Write (Y)
		Write (X)	
			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. (GATE - 2006) (2 Marks)

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

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

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

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

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

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.

Q Consider three data items D_1 , D_2 and D_3 and the following execution schedule of transactions T_1 , T_2 and T_3 . In the diagram, $R(D)$ and $W(D)$ denote the actions reading and writing the data item D respectively. **(GATE – 2003) (2 Marks)**

Which of the following statements is correct?

- (A) The schedule is serializable as $T_2; T_3; T_1$
- (B) The schedule is serializable as $T_2; T_1; T_3$
- (C) The schedule is serializable as $T_3; T_2; T_1$
- (D) The schedule is not serializable

	T1	T2	T3
time ↓			
	$R(D1);$ $W(D1);$	$R(D3);$ $R(D2);$ $W(D2);$	$R(D2);$ $R(D3);$
		$R(D1);$ $R(D2);$ $W(D2);$ $W(D3);$	$W(D2);$ $W(D3);$

Q Suppose a database schedule S involves transactions T_1, T_2, \dots, T_n . Consider the precedence graph of S with vertices representing the transactions and edges representing the conflicts. If S is serializable, which one of the following orderings of the vertices of the precedence graph is guaranteed to yield a serial schedule?

(NET-NOV-2017) (GATE- 2016) (1 Marks)

- (a) Topological order**
- (b) Depth - first order**
- (c) Breadth - first order**
- (d) Ascending order of transaction indices**

Q Let $R_i(z)$ and $W_i(z)$ denote read and write operations on a data element z by a transaction T_i , respectively. Consider the schedule S with four transactions.

$S: R_4(x) R_2(x) R_3(x) R_1(y) W_1(y) W_2(x) W_3(y) R_4(y)$

Which one of the following serial schedules is conflict equivalent to S ? **(GATE 2022) (2 MARKS)**

- (a)** $T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$
- (b)** $T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$
- (c)** $T_4 \rightarrow T_1 \rightarrow T_3 \rightarrow T_2$
- (d)** $T_3 \rightarrow T_1 \rightarrow T_4 \rightarrow T_2$

S			
T_1	T_2	T_3	T_4
			R4(x)
	R2(x)		
		R3(x)	
R1(y)			
W1(y)			
	W2(x)		
		W(y)	
			R(y)

Q. Consider the following read-write schedule S over three transactions T_1 , T_2 and T_3 . Where the subscripts in the schedule indicate transaction IDs: (Gate 2024,CS) (2 Marks) (MSQ)

S: $r_1(z); w_1(z); r_2(x); r_3(y); w_3(y); w_2(x); w_2(y);$

Which of the following transaction schedule is/are conflict equivalent to S?

(a) $T_1 T_2 T_3$

(b) $T_1 T_3 T_2$

(c) $T_3 T_2 T_1$

(d) $T_3 T_1 T_2$

T_1	T_2	T_3
$r_1(z)$		
$w_1(z)$	$r_2(x)$	
		$r_3(y)$
		$w_3(y)$
	$r_2(y)$	
	$w_2(x)$	
	$w_2(y)$	

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? **(GATE - 2021) (2 Marks)**

- 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)
	R(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 four schedules due to three transactions (indicated by the subscript) using read and write on a data item X, denoted by $r(X)$ and $w(X)$ respectively. Which one of them is conflict serializable? **(NET-NOV-2018) (GATE - 2014) (2 Marks)**

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

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

	S_3	
T_1	T_2	T_3
		$R(X)$
	$R(X)$	
		$R(X)$
		$W(X)$
	$W(X)$	

	S_4	
T_1	T_2	T_3
		$R(X)$
	$R(X)$	
		$W(X)$
		$R(X)$
	$R(X)$	
		$W(X)$

- A) $r_1(X); r_2(X); w_1(X); r_3(X); w_2(X)$
- B) $r_2(X); r_1(X); w_2(X); r_3(X); w_1(X)$
- C) $r_3(X); r_2(X); r_1(X); w_2(X); w_1(X)$
- D) $r_2(X); w_2(X); r_3(X); r_1(X); w_1(X)$

Q Consider following schedules involving two transactions :

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

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

Which of the following statement is true ? **(NET-JAN-2017) (GATE - 2007) (2 Marks)**

- (a)** Both S_1 and S_2 are conflict serializable.
- (b)** S_1 is conflict serializable and S_2 is not conflict serializable.
- (c)** S_1 is not conflict serializable and S_2 is conflict serializable.
- (d)** Both S_1 and S_2 are not conflict serializable.

Q Consider following schedules involving two transactions :

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

T_1	T_2
$r(x)$	
$r(y)$	
	$r(x)$
	$r(y)$
	$w(y)$
$w(x)$	

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

T_1	T_2
$r(x)$	
	$r(x)$
	$r(y)$
	$w(y)$
	$r(y)$
	$w(x)$

Which of the following statement is true ? **(NET-JAN-2017) (GATE - 2007) (2 Marks)**

- (a)** Both S_1 and S_2 are conflict serializable.
- (b)** S_1 is conflict serializable and S_2 is not conflict serializable.
- (c)** S_1 is not conflict serializable and S_2 is conflict serializable.
- (d)** Both S_1 and S_2 are not conflict serializable.

Q Consider the following transactions with data items P and Q initialized to zero: (GATE-2012) (2 Marks)

T_1 : read (P);

 read (Q);

 if P = 0 then Q: = Q + 1;

 write (Q);

T_1
r(P)
r(Q)
w(Q)

T_2
r(Q)
r(P)
w(P)

T_2 : read (Q);

 read (P);

 if Q = 0 then P: = P + 1;

 write (P);

Any non-serial interleaving of T1 and T2 for concurrent execution leads to

- (A) A serializable schedule
- (B) A schedule that is not conflict serializable
- (C) A conflict serializable schedule
- (D) A schedule for which a precedence graph cannot be drawn

Q. Consider two transactions T_1 and T_2 which form schedules S_1 , S_2 , S_3 and S_4 as follows:

Which of the above schedules is conflicts serializable? (GATE - 2009) (2 Marks)

T_1 : $R_1[x] W_1[x] W_1[y]$

T_2 : $R_2[x] R_2[y] W_2[y]$

S_1 : $R_1[x] R_2[x] R_2[y] W_1[x] W_1[y] W_2[y]$

S_2 : $R_1[x] R_2[x] R_2[y] W_1[x] W_2[y] W_1[y]$

S_3 : $R_1[x] W_1[x] R_2[x] W_1[y] R_2[y] W_2[y]$

S_4 : $R_2[x] R_2[y] R_1[x] W_1[x] W_1[y] W_2[y]$

S ₁	
T ₁	T ₂
R(x)	
	R(x)
	R(y)
W(x)	
W(y)	
	W(y)

S ₂	
T ₁	T ₂
R(x)	
	R(x)
	R(y)
W(x)	
	W(y)

S ₃	
T ₁	T ₂
R(x)	
W(x)	
	R(x)
W(y)	
	R(y)
	W(y)

S ₄	
T ₁	T ₂
	R(x)
	R(y)
R(x)	
W(x)	
W(y)	
	W(y)

A) S_1 and S_2

B) S_2 and S_3

C) S_3 only

D) S_4 only

Q. Consider the transactions T_1 , T_2 & T_3 and the schedules S_1 & S_2 given below. (GATE - 2014) (2 Marks)

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

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

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

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

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

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

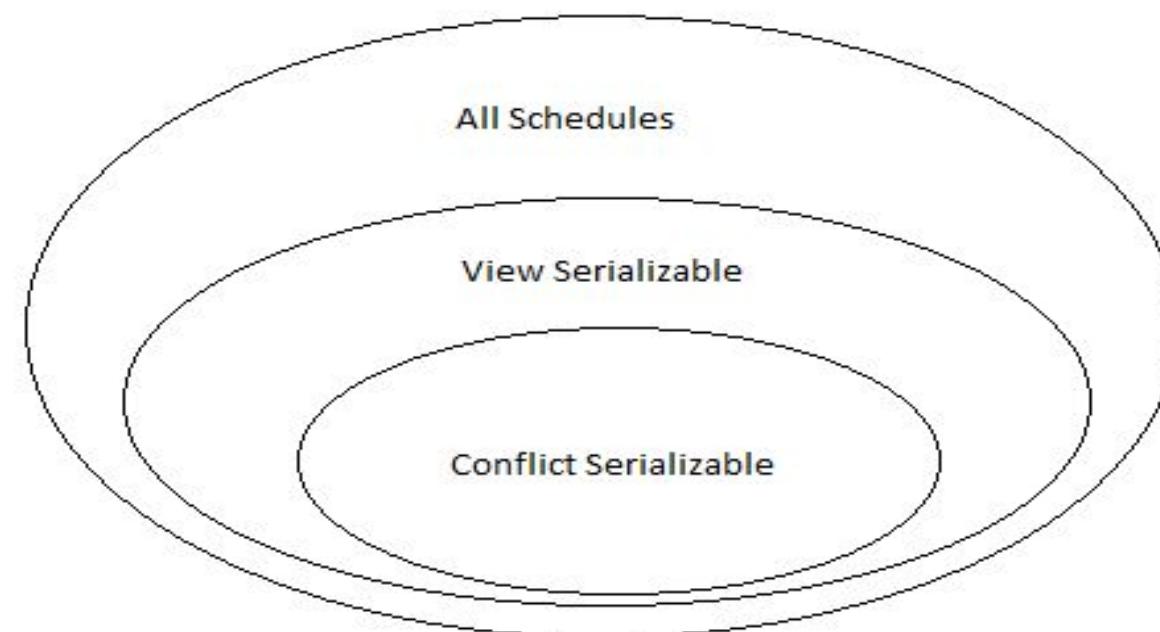
S1		
T1	T2	T3
R(X)		
		R(Y)
		R(X)
	R(Y)	
	R(Z)	
		W(Y)
	W(Z)	
R(Z)		
W(X)		
W(Z)		

S2		
T1	T2	T3
R(X)		
		R(Y)
		R(Y)
	R(Y)	
		R(X)
R(Z)		
	R(Z)	
		W(Y)
W(X)		
		W(Z)
W(Z)		

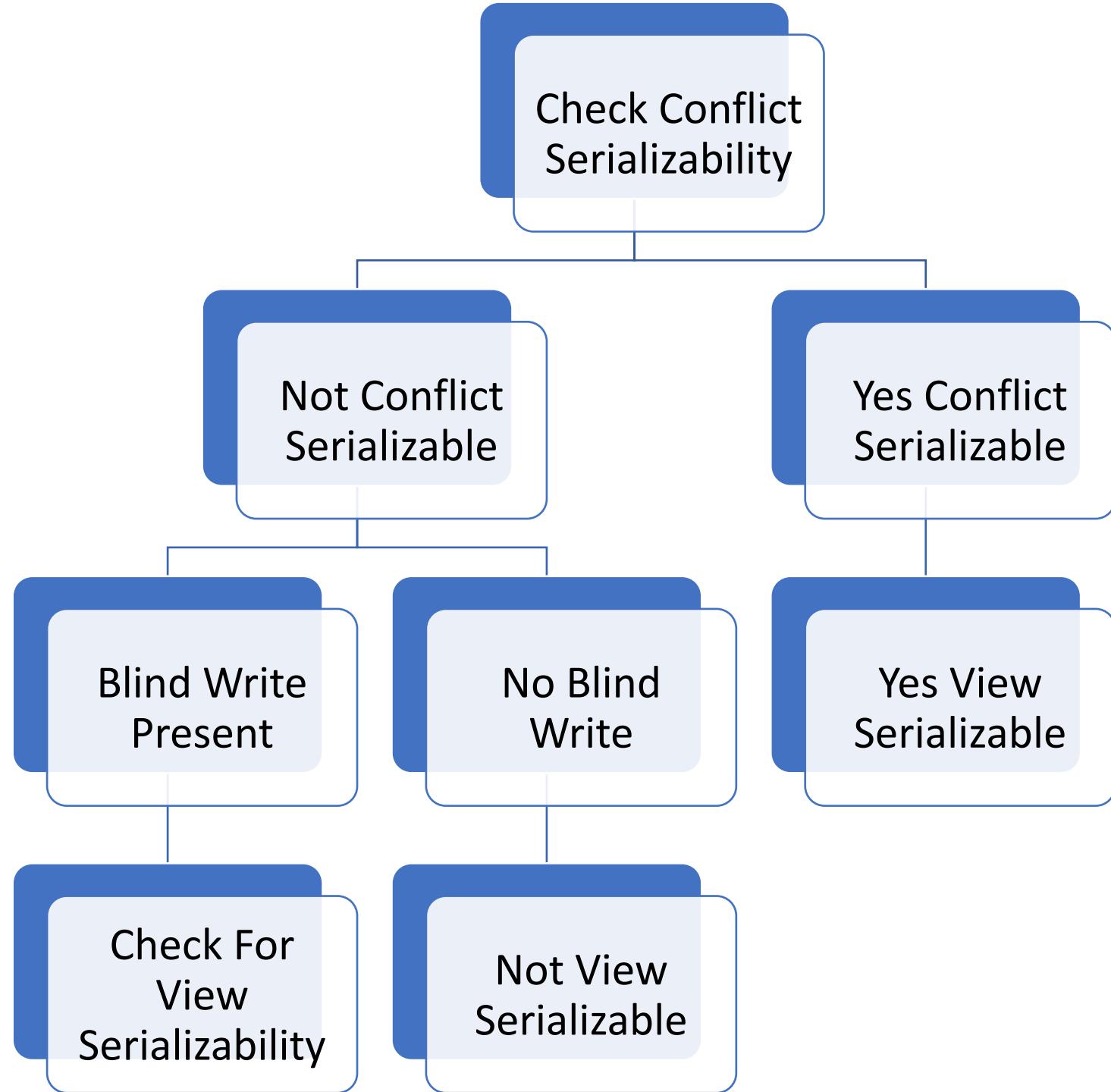
- A) Only S_1 is conflict-serializable.
- B) Only S_2 is conflict-serializable.
- C) Both S_1 and S_2 are conflict-serializable.
- D) Neither S_1 nor S_2 is conflict-serializable.

VIEW SERIALIZABLE

- If a schedule is not conflict serializable, still it can be consistent, so let us study a weaker form of serializability called View serializability, and even if a schedule is view serializable still it can be consistent.
- If a schedule is conflict serializable then it will also be view serializable, so we must check view serializability only if a schedule is not conflict serializable.



- if a schedule is not conflict serializable and it does not contain any blind write then it can never be view serializable, but if not conflict serializable and have blind write then may or may not be view serializable.
- If a schedule is not conflict serializable and if there exist a blind write.
 - First tabulate all serial schedules possible. Then check one by one whether given schedule is view equivalent to any of the serial schedule.
 - If yes then schedule is view serializable otherwise not.



- 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 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)

- Complexity wise finding the schedule is view serializable or not is a **NP- complete** problem.

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

Q Consider the following schedule 'S' with three transactions.

S: $R_1(B)$; $R_3(C)$; $R_1(A)$; $W_2(A)$; $W_1(A), W_2(B)$; $W_3(A)$; $W_1(B)$; $W_3(B), W_3(C)$

Which of the following is TRUE with respect to the above schedule?

- a)** It is conflict serializable with sequence $[T_1, T_2, T_3]$
- b)** It is conflict serializable with sequence $[T_2, T_1, T_3]$
- c)** It is view serializable but not conflict serializable
- d)** It is neither conflict serializable nor view serializable

Q Consider the following schedule 'S' with three transactions.

S: $R_1(B); R_3(C); R_1(A); W_2(A); W_1(A), W_2(B); W_3(A); W_1(B); W_3(B), W_3(C)$

Which of the following is TRUE with respect to the above schedule?

- a) It is conflict serializable with sequence $[T_1, T_2, T_3]$
- b) It is conflict serializable with sequence $[T_2, T_1, T_3]$
- c) It is view serializable but not conflict serializable
- d) It is neither conflict serializable nor view serializable

S		
T_1	T_2	T_3
$R(B)$		
		$R(C)$
$R(A)$		
	$W(A)$	
$W(A)$		
	$W(B)$	
		$W(A)$
$W(B)$		
		$W(B)$
		$W(C)$

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_j appears before T_i . Such a schedule is called Non- Recoverable schedule.

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

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

Q Consider the following schedule S of transactions T_1, T_2, T_3, T_4 (GATE - 2014) (2 Marks)

Which one of the following statements is CORRECT?

- (A) S is conflict-serializable but not recoverable
- (B) S is not conflict-serializable but is recoverable
- (C) S is both conflict-serializable and recoverable
- (D) S is neither conflict-serializable nor is it recoverable

T1	T2	T3	T4
Writes(X) Commit	Reads(X) Writes(Y) Reads(Z) Commit	Writes(X) Commit	Reads(X) Reads(Y) Commit

Q Let S be the following schedule of operations of three transactions T_1 , T_2 and T_3 in a relational database system: $R_2(Y), R_1(X), R_2(Z), R_1(Y)W_1(X), R_2(Z), W_2(Y), R_3(X), W_3(Z)$

Consider the statements P and Q below:

- **P:** S is conflict-serializable.
- **Q:** If T_3 commits before T_1 finishes, then S is recoverable.

Which one of the following choices is correct? **(GATE 2021) (2 MARKS)**

- (a) Both P and Q are true
- (b) P is true and Q is false
- (c) P is false and Q is true
- (d) Both P and Q are false

Q Let S be the following schedule of operations of three transactions T_1 , T_2 and T_3 in a relational database system: $R_1(Y), R_1(X), R_2(Z), R_1(Y)W_1(X), R_2(Z), W_2(Y), R_3(X), W_3(Z)$

Consider the statements P and Q below:

- **P:** S is conflict-serializable.
- **Q:** If T_3 commits before T_1 finishes, then S is recoverable.

Which one of the following choices is correct? (GATE 2021) (2 MARKS)

(a) Both P and Q are true

(b) P is true and Q is false

(c) P is false and Q is true

(d) Both P and Q are false

S		
T_1	T_2	T_3
	$R(Y)$	
$R(X)$		
		$R(Z)$
$W(X)$		$W(Z)$
	$R(z)$	
	$W(Y)$	
		$R(X)$
		$W(z)$

CASCADING ROLLBACK

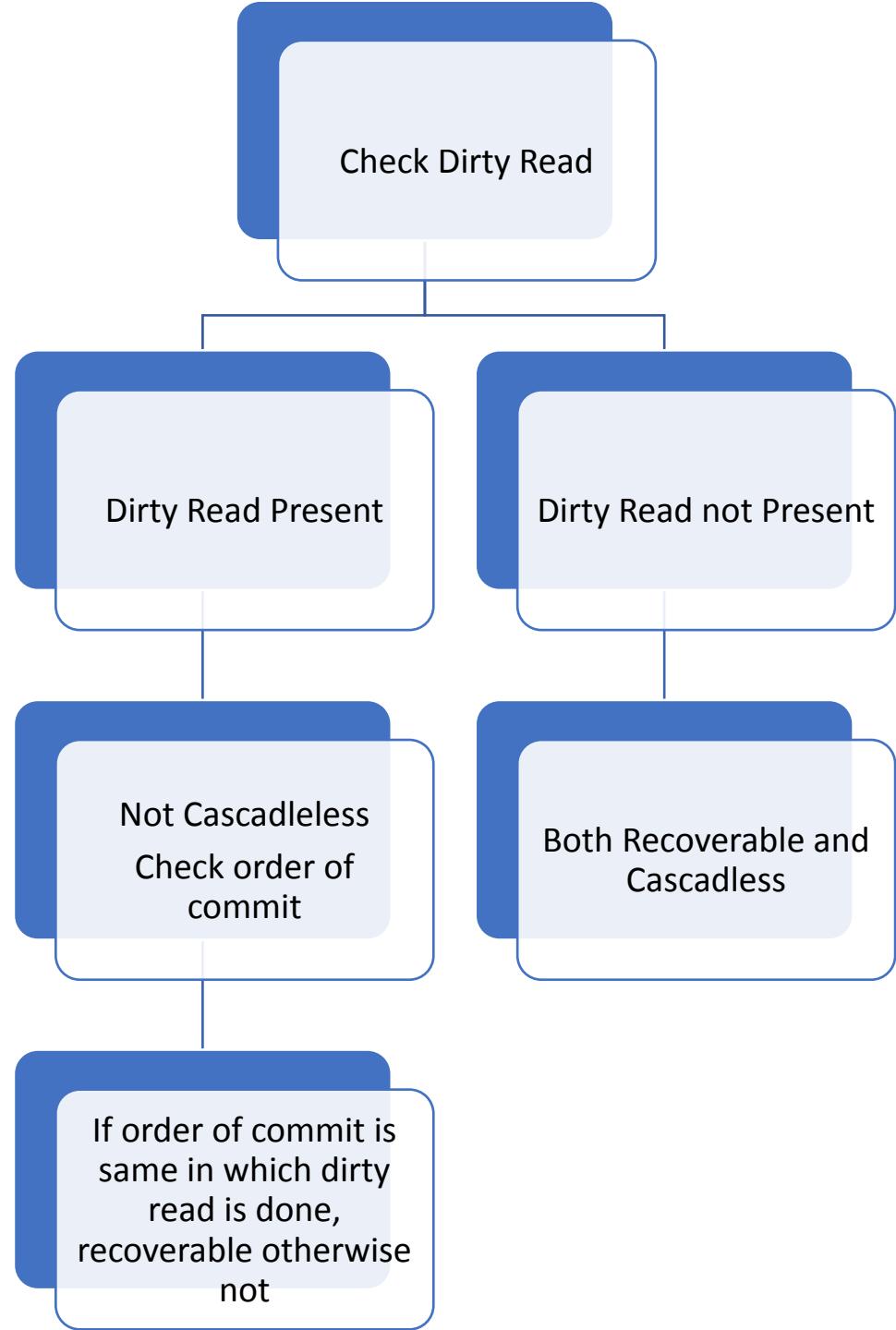
- It is a phenomenon, in which a single transaction failure leads to a series of transaction rollbacks, is called cascading rollback. Even if the schedule is recoverable the, the commit of transaction may lead lot of transaction to rollback.
- Cascading rollback is undesirable, since it leads to undoing of a significant amount of work. Uncommitted reads are not allowed in cascade less schedule.

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

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)		
C		
	R(X)	
	W(X)	
	C	
		R(X)
		C



Q Consider the following partial Schedule S involving two transactions T_1 and T_2 . Only the read and the write operations have been shown. The read operation on data item P is denoted by $\text{read}(P)$ and the write operation on data item P is denoted by $\text{write}(P)$.

Suppose that the transaction T_1 fails immediately after time instance 9. Which one of the following statements is correct? **(GATE-2015) (2 Marks)**

- (A) T_2 must be aborted and then both T_1 and T_2 must be re-started to ensure transaction atomicity
- (B) Schedule S is non-recoverable and cannot ensure transaction atomicity
- (C) Only T_2 must be aborted and then re-started to ensure transaction atomicity
- (D) Schedule S is recoverable and can ensure atomicity and nothing else needs to be done

Time	Transaction-id	
	T_1	T_2
1	$\text{read}(A)$	
2	$\text{write}(A)$	
3		$\text{read}(C)$
4		$\text{write}(C)$
5		$\text{read}(B)$
6		$\text{write}(B)$
7		$\text{read}(A)$
8		commit
9	$\text{read}(B)$	

Q. A schedule of three database transactions T_1 , T_2 , and T_3 is shown. $R_i(A)$ and $W_i(A)$ denote read and write of data item A by transaction T_i , $i=1,2,3$. The transaction T_1 aborts at the end. Which other transaction(s) will be required to be rolled back? **(Gate 2025)**

$R_1(X)$ $W_1(Y)$ $R_2(X)$ $R_2(Y)$ $R_3(Y)$ $ABORT(T_1)$

- A) Only T_2
- B) Only T_3
- C) Both T_2 and T_3
- D) Neither T_2 nor T_3

Strict Schedule

- 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 and write operation of T_j .

S_1	
T_1	T_2
R(a)	
W(a)	
	W(a)
C	
	R(a)
	C

S_2	
T_1	T_2
R(a)	
W(a)	
C	
	W(a)
	R(a)
	C

S_3	
T_1	T_2
R(a)	
	R(b)
W(a)	
	W(b)
C	
	R(a)
	C

Recoverable Schedules

Cascadeless Schedules

Strict Schedules

Q Consider the following database schedule with two transactions, T_1 and T_2 . $S = r_2(X); r_1(X); r_2(Y); w_1(X); r_1(Y); w_2(X); a_1; a_2$, where $r_i(Z)$ denotes a read operation by transaction T_i on a variable Z , $w_i(Z)$ denotes a write operation by T_i on a variable Z and a_i denotes an abort by transaction T_i . Which one of the following statements about the above schedule is **TRUE?** (**GATE-2016**) (1 Marks)

- (a) S is non-recoverable
- (b) S is recoverable, but has a cascading abort
- (c) S does not have a cascading abort
- (d) S is strict

Q Consider the following database schedule with two transactions, T_1 and T_2 . $S = r_2(X); r_1(X); r_2(Y); w_1(X); r_1(Y); w_2(X); a_1; a_2$, where $r_i(Z)$ denotes a read operation by transaction T_i on a variable Z , $w_i(Z)$ denotes a write operation by T_i on a variable Z and a_i denotes an abort by transaction T_i . Which one of the following statements about the above schedule is **TRUE?** (GATE-2016) (1 Marks) [Amcat]

- (a) S is non-recoverable
 - (b) S is recoverable, but has a cascading abort
 - (c) S does not have a cascading abort
 - (d) S is strict

S	
T ₁	T ₂
R(X)	R(Y)
W(X)	
R(Y)	
	W(X)
a ₁	
	a ₂

Q Consider the following schedules:

$S_1: R_1(x) W_1(x) R_1(y) R_2(x) W_2(x) C_2, C_1;$

$S_2: R_2(x) W_2(x) R_1(y) R_1(x) W_2(x) C_2, C_1;$

Which of the following is true?

- a) Both S_1 and S_2 are recoverable
- b) S_1 is recoverable but S_2 is not
- c) S_2 is recoverable but S_1 is not
- d) Both schedule are not Recoverable

Q Consider the following schedules:

$S_1: R_1(x) W_1(x) R_1(y) R_2(x) W_2(x) C_2, C_1;$

$S_2: R_2(x) W_2(x) R_1(y) R_1(x) W_2(x) C_2, C_1;$

S_1	
T_1	T_2
$R(x)$	
$W(x)$	
$R(y)$	
	$R(x)$
	$W(x)$
	C
C	

Which of the following is true?

- a) Both S_1 and S_2 are recoverable
- b) S_1 is recoverable but S_2 is not
- c) S_2 is recoverable but S_1 is not
- d) Both schedule are not Recoverable

S_2	
T_1	T_2
	$R(x)$
	$W(x)$
$R(y)$	
$R(x)$	
	$W(x)$
	C
C	

Q Consider the following schedule 'S'.

S: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; c_1 ; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$; c_3 ; c_2 ;

The schedule 'S' is

- a) Recoverable**
- b) Cascade less**
- c) recoverable and cascade-less**
- d) None**

Q Consider the following schedule 'S'.

S: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; c_1 ; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$; c_3 ; c_2 ;

The schedule 'S' is

a) Recoverable

b) Cascade less

c) recoverable and cascade-less

d) None

S		
T_1	T_2	T_3
$R(X)$		
	$R(Z)$	
		$R(X)$
$R(Z)$		
	$R(Y)$	
		$R(Y)$
$W(X)$		
C		
	$W(Z)$	
		$W(Y)$
	$W(Y)$	
		C
	C	

Q Consider the given schedule

$R_1(X), R_2(Z), R_1(Z), R_3(X), R_3(Y), W_1(X), Commit_1, W_3(Y), Commit_3, R_2(Y), W_2(Z), W_2(Y)$,
 $Commit_2$. The given schedule is

- a)** Recoverable only
- b)** Cascade less only
- c)** recoverable and cascade-less
- d)** None

Q Consider the given schedule

$R_1(X), R_2(Z), R_1(Z), R_3(X), R_3(Y), W_1(X), Commit_1, W_3(Y), Commit_3, R_2(Y), W_2(Z), W_2(Y)$,
 $Commit_2$. The given schedule is

a) Recoverable only

b) Cascade less only

c) recoverable and cascade-less

d) None

S		
T_1	T_2	T_3
$R(X)$		
	$R(Z)$	
$R(Z)$		
		$R(X)$
		$R(Y)$
$W(X)$		
C		
		$W(Y)$
		C
	$R(Y)$	
	$W(Z)$	
	$W(Y)$	
	C	

Q Consider the following schedule

S; R₂(x), w₂(x), R₃(y), R₁(x), R₁(y), w₁(x), w₃(y), R₃(x), R₁(y), C₃, C₂, C₁;

The above schedule is

- a) Recoverable but not cascade less
- b) Recoverable and cascade less but not strict
- c) Recoverable, cascade less and also strict
- d) Not recoverable

Q Consider the following schedule

S: $R_2(x)$, $w_2(x)$, $R_3(y)$, $R_1(x)$, $R_1(y)$, $w_1(x)$, $w_3(y)$, $R_3(x)$, $R_1(y)$, C_3 , C_2 , C_1 ;

The above schedule is

- a) Recoverable but not cascade less
- b) Recoverable and cascade less but not strict
- c) Recoverable, cascade less and also strict
- d) Not recoverable

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

Q Which of the following statement is/are correct

- a)** Every view serializable schedule is conflict serializable
- b)** Every strict schedule is conflict serializable
- c)** Every conflict serializable schedule is cascade less
- d)** None of the above

Q Consider a simple checkpointing protocol and the following set of operations in the log.

(start, T₄);
(write, T₄, y, 2, 3);
(start, T₁);
(commit, T₄);
(write, T₁, z, 5, 7);
(checkpoint);
(start, T₂);
(write, T₂, x, 1, 9);
(commit, T₂);
(start, T₃);
(write, T₃, z, 7, 2);

If a crash happens now and the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list (**GATE - 2015**) (2 Marks)

- (A) Undo: T₃, T₁; Redo: T₂
- (B) Undo: T₃, T₁; Redo: T₂, T₄
- (C) Undo: none; Redo: T₂, T₄, T₃; T₁
- (D) Undo: T₃, T₁, T₄; Redo: T₂

Q Consider a simple checkpointing protocol and the following set of operations in the log.

(start, T₄);
(write, T₄, y, 2, 3);
(start, T₁);
(commit, T₄);
(write, T₁, z, 5, 7);
(checkpoint);
(start, T₂);
(write, T₂, x, 1, 9);
(commit, T₂);
(start, T₃);
(write, T₃, z, 7, 2);

S			
T ₁	T ₂	T ₃	T ₄
			START
			(write, y, 2, 3)
START			
			COMMIT
(write, z, 5, 7)			
	(checkpoint)	(checkpoint)	(checkpoint)
	START		
	(write, x, 1, 9)		
	COMMIT		
		START	
		(write, z, 7, 2)	

If a crash happens now and the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list (**GATE - 2015**) (2 Marks)

- (A) Undo: T₃, T₁; Redo: T₂
- (B) Undo: T₃, T₁; Redo: T₂, T₄
- (C) Undo: none; Redo: T₂, T₄, T₃; T₁
- (D) Undo: T₃, T₁, T₄; Redo: T₂

- Q** Suppose a database system crashes again while recovering from a previous crash. Assume check pointing is not done by the database either during the transactions or during recovery. Which of the following statements is/are correct? **(GATE 2021)**
- (a)** The same undo and redo list will be used while recovering again
 - (b)** The system cannot recover any further
 - (c)** All the transactions that are already undone and redone will not be recovered again
 - (d)** The database will become inconsistent

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
 1. Belong to different transactions
 2. Must operate on same data value
 3. 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
 1. Basic 2pl
 2. Conservative 2pl
 3. Rigorous 2pl
 4. 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

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.
- Let's understand how this protocol works, here we have two idea of timestamping, one for the transaction, and other for the data item.

- Time stamp for transaction,
 - With each transaction t_i , in the system, we associate a unique fixed timestamp, denoted by $TS(t_i)$.
 - This timestamp is assigned by database system to a transaction at time transaction enters into the system.
 - If a transaction has been assigned a timestamp $TS(t_i)$ and a new transaction t_j , enters into the system with a timestamp $TS(t_j)$, then always $TS(t_i) < TS(t_j)$.

- Two things are to be noted
 1. First time stamp of a transaction remain fixed throughout the execution
 2. Second it is unique means no two transaction can have the same timestamp.
- The reason why we called time stamp not stamp, because for stamping we use the value of the system clock as stamp, advantage is,
 - T_t will always be unique as time never repeats
 - There is no requirement of refreshing and starting with fresh value.

- The time stamp of the transaction also determines the serializability order.
- Thus if $TS(t_i) < TS(t_j)$, then the system must ensure that the produced schedule is equivalent to a serial schedule in which transaction t_i appears before transaction t_j .

- Time stamp with data item, in order to assure such scheme, the protocol maintains for each data item Q two timestamp values:
 1. **W-timestamp(Q)** is the largest time-stamp of any transaction that executed write(Q) successfully.
 2. **R-timestamp(Q)** is the largest time-stamp of any transaction that executed read(Q) successfully.
- These timestamps are updated whenever a new read(Q) or write(Q) instruction is executed.

- Suppose a transaction T_i request a ***read(Q)***
 1. 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.
 2. 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)***.
 1. 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.
 2. 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.
 3. If $TS(T_i) \geq R\text{-timestamp}(Q)$, then the write operation is executed, and $W\text{-timestamp}(Q)$ is set to $\max(W\text{-timestamp}(Q), TS(T_i))$.
 4. If $TS(T_i) \geq W\text{-timestamp}(Q)$, then the write operation is executed, and $W\text{-timestamp}(Q)$ is set to $\max(W\text{-timestamp}(Q), TS(T_i))$.

- If a transaction T_i is rolled back by the concurrency control scheme as a result of either a read or write operation, the system assigns it's a new timestamp and restarts it.

	Conflict Serializability	View Serializability	Recoverability	Cascadelessness	Deadlock Freedom
Time Stamp Ordering					
Thomas Write Rule					
Basic 2PL					
Conservative 2PL					
Rigorous 2PL					
Strict 2PL					

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.

Conclusion

- It may cause starvation to occur, as if a sequence of conflicting transactions causes repeated restarting of the long transaction, and then there is a possibility of starvation of long transaction.
- It is relatively slow as before executing every instruction we have to check conditions before. Time stamping protocol ensure that the schedule designed through this protocol will always be conflict serializable.
- This protocol can also be used for determining the serializability order (order in which transaction must execute) among the transaction in advance.

- Further modifications are possible if we want to ensure recoverability and cascade lessness, using different approaches
 - By performing all writes together at the end of the transaction, i.e. while writers are in progress, no transaction is permitted to access any of data items that have been written.
 - By using a limited form of locking, where by read of uncommitted items are postponed until the transaction that uploaded the item commit.
 - Recoverability alone can be ensured by tracking uncommitted writes and allowing a transaction t_i to commit only after the commit of any transaction that wrote a value that t_i read.

THOMAS WRITE RULE

- Thomas write is an improvement in time stamping protocol, which makes some modification and may generate those protocols that are even view serializable, because it allows greater potential concurrency.
- It is a Modified version of the timestamp-ordering protocol in which Blind write operations may be ignored under certain circumstances.
- The protocol rules for read operations remain unchanged. while for write operation, there is slightly change in Thomas write rule than timestamp ordering protocol.

When T_i attempts to write data item Q,

- if $TS(T_i) < W\text{-timestamp}(Q)$, then T_i is attempting to write an obsolete value of {Q}. Rather than rolling back T_i as the timestamp ordering protocol would have done, this {write} operation can be ignored.

- This modification is valid as the any transaction with $TS(T_i) < W\text{-timestamp}(Q)$, the value written by this transaction will never be read by any other transaction performing $\text{Read}(Q)$ ignoring such obsolete write operation is considerable.
- Thomas' Write Rule allows greater potential concurrency. Allows some view-serializable schedules that are not conflict serializable.

T_3	T_4	T_6
read(Q)	write(Q)	
write(Q)		write(Q)

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

Q In a database system, unique timestamps are assigned to each transaction using Lamport's logical clock. Let $TS(T_1)$ and $TS(T_2)$ be the timestamps of transactions T_1 and T_2 respectively. Besides, T_1 holds a lock on the resource R and T_2 has requested a conflicting lock on the same resource R. The following algorithm is used to prevent deadlocks in the database system assuming that a killed transaction is restarted with the same timestamp.

```
if  $TS(T_2) < TS(T_1)$  then  
     $T_1$  is killed  
else  $T_2$  waits.
```

Assume any transaction that is not killed terminates eventually. Which of the following is TRUE about the database system that uses the above algorithm to prevent deadlocks? **(GATE-2017) (2 Marks)**

- (a) The database system is both deadlock-free and starvation-free
- (b) The database system is deadlock-free, but not starvation-free
- (c) The database system is starvation-free, but not deadlock-free
- (d) The database system is neither deadlock-free nor starvation-free

Q Which of the following time stamp ordering protocol(s) allow(s) the following schedules?

S: $W_1(A)$; $W_2(A)$; $W_3(A)$; $R_2(A)$; $R_4(A)$;

Time stamps : $T_1 : 5$, $T_2 : 10$, $T_3 : 15$; $T_4 : 20$

a) Thomas write rule

b) Basic time stamp

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	-

- 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 is using locking but neither it is conflict serializable nor independent from deadlock.

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)

- 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.

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					
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? **(NET-JUNE-2015) (NET-JUNE-2014) (GATE-2010)** (2 Marks)

(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. Which of the following statements about the Two Phase Locking (2PL) protocol is/are TRUE? (Gate 2024 CS) (1 Marks) (MSQ)

(a) 2PL permits only serializable schedules

(b) With 2PL, a transaction always locks the data item being read or written just before every operation and always releases the lock just after the operation

(c) With 2PL, once a lock is released on any data item inside a transaction, no more locks on any data item can be obtained inside that transaction

(d) A deadlock is possible with 2PL

	Conflict Serializability	View Serializability	Recoverabilit y	Cascadeless ness	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

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					
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.

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					
Strict 2PL					

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

	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					

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? **(GATE-2019) (1 Marks)**

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

Q Consider the following two-phase locking protocol. Suppose a transaction T accesses (for read or write operations), a certain set of objects $\{O_1, \dots, O_k\}$. This is done in the following manner:

Step1. T acquires exclusive locks to O_1, \dots, O_k in increasing order of their addresses.

Step2. The required operations are performed.

Step3. All locks are released.

This protocol will **(GATE- 2016) (1 Marks)**

- (a) guarantee serializability and deadlock-freedom
- (b) guarantee neither serializability nor deadlock-freedom
- (c) guarantee serializability but not deadlock-freedom
- (d) guarantee deadlock-freedom but not serializability

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