

It's just prediction based Questions so don't believe blindly on these.

MANAGEMENT SYSTEM

DBMS-1

TO DATABASE SYSTEMS

1

YEARS QUESTIONS

Q.3 Explain the concepts of Primary key. [R.T.U. 2019]

Ans. Every record must have one unique identifier called the primary key that has a unique value within the table or collection. Primary keys can be concatenated which means that the uniqueness can be made up of one or more fields.

Syntax to define a primary key at column level:

Column name data type [CONSTRAINT constraint_name] PRIMARY KEY

Syntax to define a primary key at table level:

[CONSTRAINT Constraint_name] PRIMARY KEY

[Column_name], column_name2]

Q.4 What is Entity? Explain with a suitable example. [R.T.U. 2019]

Ans. Entity : In database systems, objects are referred as entity. "An entity is a thing or object in the real world that is distinguishable from other things or objects."

For example, each person is an entity and banks accounts can be considered as entity.

Note : Object and class of OOPs are referred as an entity and an entity set in database system respectively.

[R.T.U. 2019]

Q.5 What is DBMS?

Ans. DBMS : A Database Management system is a collection of interrelated data and collection of programs to access that data. The data describes one particular enterprise. Database systems are ubiquitous today and most people interact either directly or indirectly, with

DBMS-2

databases many times everyday. A major purpose of the database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.

Underlying the structure of a database is the data model- a collection of conceptual tools for describing data, data relationships, data semantics and data constraints. The entity relationship (E-R) data model is a widely used data model which provides a convenient graphical representation to view data, relationship and constraints.

Q.6 What is the difference between logical data independence and physical data independence? [R.T.U. 2016, 2015]

Ans.	Logical Data Independence	Physical Data Independence
S. No.		
1.	Indicates that conceptual/logical schema can be changed without affecting the existing external (view) schemas.	Indicates that the physical storage structures or devices used for storing the data could be changed without any change in the conceptual view or external view.
2.	The change would be absorbed by the mapping between the external and conceptual (logical) levels.	The change would be absorbed by the mapping between the conceptual and internal levels.
3.	Modifications such as the deletion of a conceptual view field or record may require changes in the external view and application program.	Modifications of physical level improve the performance.

Q.7 What do you mean by constraints? Explain different types of constraints with examples. [R.T.U. 2016]

Ans. Constraints : Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the data integrity during an update/delete/insert into a table.

Different Types of constraints that can be created in RDBMS

1. Mapping Constraints :

Mapping Cardinalities : Express the number of entities to which another entity can be associated via a relationship.

For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:

- One-to-one** : 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.
- One-to-many** : An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.
- Many-to-one** : An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.
- Many-to-many** : Entities in A and B are associated with any number from each other.

Q.8 Explain physical and external view of data.

Ans. External View : In the relational model, the external view also presents data as a set of relations. It is tailored to the needs of a particular category of users. Portions of stored data should not be seen by some users and begins to implement a level of security and simplifies the view for these users. For example, students should not see faculty salaries, faculty should not see billing or payment data, etc.

Physical View : The physical view describes the details of how data is stored: files, indices, etc., on the random access disk system. It also typically describes the record layout of files and type of files (hash, b-tree, flat).

Q.9 Differentiate between candidate keys and super keys.

Ans. Difference between Candidate Keys and Super Keys

S.No.	Candidate Keys	Super Keys
1.	A candidate key is any set of one or more columns whose combined value is unique through out the table.	An attribute, or group of attributes, that is sufficient to distinguish every tuple in the relation from every other one.
2.	Each candidate key is not called super key and no component of a candidate key is allowed to be null.	Each super key is called a candidate key.

Q.10 Explain the difference between partial & total participation.

Database Management System

Ans. Difference between Partial and Total Participation

S.No.	Partial Participation	Total Participation
1.	All instances need not participate.	Every entity instance must be connected through the relationship to another instance of the other participating entity types.
2.	Represented by single line from entity rectangle to relationship diamond.	Represented by double line from entity rectangle to relationship diamond.

Q.11 Differentiate between entity and entity sets.

Ans. Difference between Entity and Entity Sets

S. No.	Entity	Entity Set
1.	Entities are the people, places, things, events and concepts of interest to an organization.	Entity set represents collection of things. Example : An employee entity set might represent a collection of all the employees that work for an organization.
2.	Entity can be concrete like employee, book, project etc. and can be abstract or a concept. Example : title, job company etc.	Entity set need not be disjoint. For example : the entity set employee (all employee of bank) and the entity set customer (all customer of the bank) may have members in common.

PART-B

Q.12 What is E-R model? What are the features of E-R model? Draw and explain E-R model for Library Management System. [R.T.U. 2019]

Ans. E-R Model : The entity-relationship (E-R) data model is based on a perception of a real world that consists of a collection of basic objects, called entities and of

relationships among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. For example, each person is an entity and bank accounts can be considered as entities.

Entities are described in a database by a set of attributes. For example, the attributes *account-number* and *balance* may describe one particular account in a bank and they form attributes of the *account* entity set. Similarly, attributes *customer-name*, *customer-street* address and *customer-city* may describe a *customer* entity.

An extra attribute *customer-id* is used to uniquely identify customers (since it may be possible to have two customers with the same name, street address and city).

A unique customer identifier must be assigned to each customer. In the United States, many enterprises use the social-security number of a person (a unique number the U.S. government assigns to every person in the United States) as a customer identifier.

A **relationship** is an association among several entities. For example, a *depositor* relationship associates a customer with each account that she has. The set of all entities of the same type and the set of all relationships of the same type are termed an **entity set** and **relationship set**, respectively.

The overall logical structure (schema) of a database can be expressed graphically by an **E-R diagram**, which is built up from the following components -

- Rectangles** : Which represent entity sets.
- Ellipses** : Which represent attributes.
- Diamonds** : Which represent relationships among entity sets.
- Lines** : Which link attributes to entity sets and entity sets to relationships.

Each component is labeled with the entity or relationship that it represents.

As an illustration, consider part of a database banking system consisting of customers and of the accounts that these customers have. Figure 1 shows the corresponding E-R diagram.

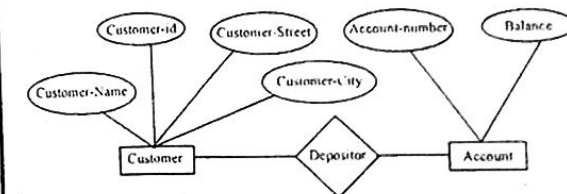


Fig. 1 : A simple E-R Diagram

DBMS-4

The E-R diagram indicates that there are two entity sets, *customer* and *account*, with attributes. The diagram also shows a relationship *depositor* between *customer* and *account*.

In addition to entities and relationships, the E-R model represents certain constraints to which the contents of a database must conform. One important constraint is **mapping cardinalities**, which express the number of entities to which another entity can be associated via a relationship set. For example - if each account must belong to only one customer, the E-R model can express that constraint.

E-R diagrams also provide a way to indicate more complex constraints on the number of times each entity participates in relationships in a relationship set. An edge between an entity set and a binary relationship set can have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* is the maximum cardinality. A minimum value of 1 indicates total participation of the entity set in the relationship set. A maximum value of 1 indicates that the entity participates in at most one relationship, while a maximum value * indicates no limit.

Note that a label 1..* on an edge is equivalent to a double line.

For example, consider Figure 2. The edge between *loan* and *borrower* has a cardinality constraint of 1..1, meaning the minimum and the maximum cardinality both are 1. That is, each loan must have exactly one associated customer. The limit 0..* on the edge from *customer* to *borrower* indicates that a customer can have zero or more loans. Thus, the relationship *borrower* is one to many from *customer* to *loan* and further the participation of *loan* in *borrower* is total.

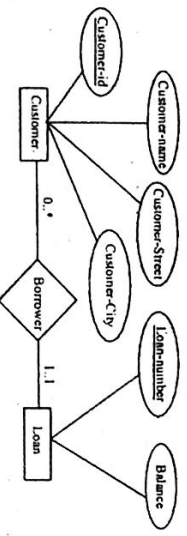


Fig. 2 : Cardinality Limits on Relationship Sets

It is easy to misinterpret the 0..* on the edge between *customer* and *borrower* and think that the relationship *borrower* is many to one from *customer* to *loan*-this is exactly the reverse of the correct interpretation.

B.Tech. (IV Sem.) C.S. Solved Papers

If both edges from a binary relationship have a maximum value of 1, the relationship is one to one. If we had specified a cardinality limit of 1..* on the edge between *customer* and *borrower*, we would be saying that each customer must have at least one loan.

E-R Notations :

1. **E** entity set =

customer, **employee**, **loan**, **branch**, **account**

These are entity sets.

2. **Weak entity** → Loan payment are weak entity.

3. **Relationship set** → Shows the relationship

4. **A** → Show primary set

Customer_id, branch name, account-no. These are primary keys.

5. **○** - All the attributes are shown through ellipse.

6. **⬢** - Derived attribute

Employment-length is derived attribute.

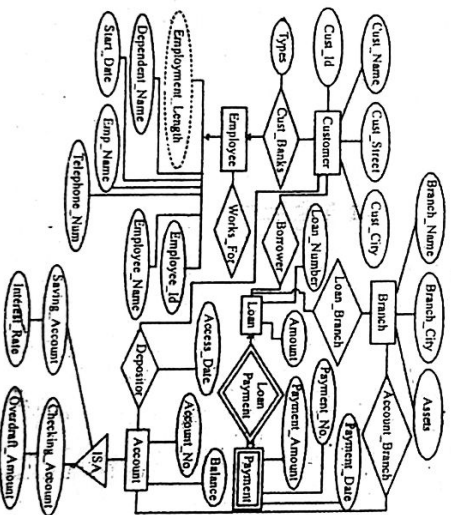
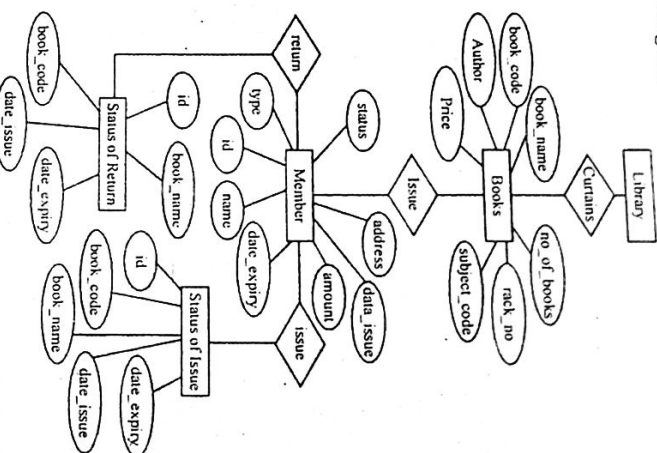


Fig. 3 : E-R Diagram

The E-R model defines the conceptual view of a database. E-R model is a high-level conceptual model for database design.

Database Management System

ER Diagram for Library Management System :



Useful in designing Real World Database

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

Then we have the notion of entity sets. An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students teachers of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

It is this important aspect of the entities and the design of E-R Model, that make the Entity-Relationship Model useful in designing Real-World Databases, because real-world objects can directly be taken as entities. There can directly be taken as attributes for entities. Thus, E-R Model is useful for designing real world databases.

Convert ER-Diagram to Relational Database

1. Members :

Column Name	Id no (PK)
Data Type	Text

2. Add Books

Column Name	Book Name (PK)
Data Type	Text

3. Issue :

Column Name	Id no (FK)
Data Type	Text

Q13) Different Explain it example.

Ans. File System and File Manager

File Manager

File System is a general file system which less security and Data Redundancy file management system in file system.

Centralisation is when it comes to File Management System User locates the address of the file data in File Manager System.

Security is low in File Management System stores unstructured isolated data files

stem :

Convert ER-Diagram into Tables :

DBMS.5

1. Members :

Column Name	Id_no (PK)	Name	Address	Date of Issue	Date of Expiry	Status
Data Type	Text	Text	Text	Date/time	Date/time	Text

2. Add_Books :

Column Name	Book_name	Book_code (PK)	Author	Date_of_arrival	Price	Rack_no	No_of_books	Subject_code
Data Type	Text	Text	Text	Date/time	Date/time	Text	Text	Text

3. Issue :

Column Name	Id_no (FK)	Book_Name	Issue_date	Due_date
Data Type	Text	Text	Date/time	Date/time

Q.13 Differentiate between file system and DBMS. Explain the ternary relationship with a suitable example. [R.T.U. 2019]

Ans. File System vs DBMS - Difference between File System and DBMS

File Management System	Database Management System
File System is a general, easy-to-use system to store general files which require less security and constraints.	Database management system is used when security constraints are high.
Data Redundancy is more in file management system.	Data Redundancy is less in database management system.
Data Inconsistency is more in file system.	Data Inconsistency is less in database management system.
Centralisation is hard to get when it comes to File Management System.	Centralisation is achieved in Database Management System.
User locates the physical address of the files to access data in File Management System.	In Database Management System, user is unaware of physical address where data is stored.
Security is low in File Management System.	Security is high in Database Management System.
File Management System stores unstructured data as isolated data files/entities.	Database Management System stores structured data which have well defined constraints and interrelation.

atabase
ther animate
for example,
classes, and
es. All these
at give them

ns of their
e values. For
lass, and age

ts. An entity
An entity set
nilar values.
the students
tain all the
y sets need

es and the
relationship
s, because
ies. There
a student
hus, E-R
ses.

DBMS.6

Ternary Relationship : Suppose we want to keep track of which person got which degree from which University. We could represent this with a ternary relationship as follows :

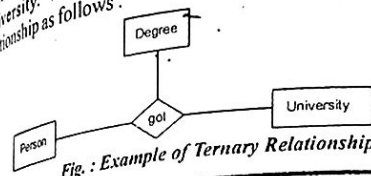


Fig. : Example of Ternary Relationship

Draw and explain architecture of RDBMS. [R.T.U. 2017]

Ans. Architecture of RDBMS : There are five major components that are exercised in a typical interaction with RDBMS:

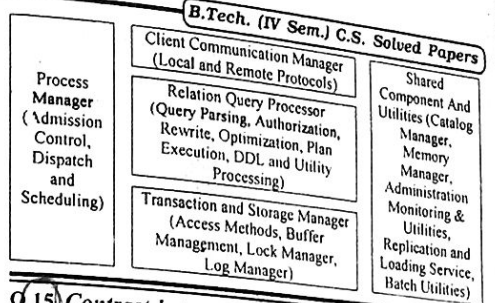
(1) Client Communication Manager : In order to communicate with a database an application needs to make a connection with a database over a network. An application establishes a connection with the Client Communication Manager. This component enables communication between various database clients through both local and remote protocols. Its main responsibility is to remember communication state, return data and control messages (result codes, errors) as well as forward the client's request to other parts of the DBMS.

(2) Process Manager : The process manager is responsible for providing a "thread of computation" for each database request from a database client. It links the threads data and control output to the appropriate communication manager client. The first decision to be made by the process manager is to determine if enough system resources are available to execute the query or defer the same until a later time.

(3) Relational Query Processor : On receiving a request to process a query the relation query processor first checks if the user is authorised to run the query. It then compiles the query into an interim query plan which is further optimised. The resulting plan is executed by the "plan executor" which eventually makes use of the transaction and storage monitor.

(4) Transaction and Storage Manager : Once a query is parsed it retrieves the requested data from the Transaction and Storage Manager. It the gatekeeper to all data access and manipulation calls. The transactional and storage manager also make sure that ACID properties of a transaction are adhered too and thus the need for a lock and log manager.

(5) Shared Components and Utilities : There are a number of shared components and utilities that are essential for a database to run.



Q.15 Contrast between DDL and DML.

Differential between DDL and DML using syntax for them. [R.T.U. 2017]

Differentiate between DDL and DML. [R.T.U. 2016]

Ans. Difference between DDL and DML : Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE : to create objects in the database
- ALTER : alters the structure of the database
- DROP : delete objects from the database
- TRUNCATE : remove all records from a table, including all spaces allocated for the records are removed
- COMMENT : add comments to the data dictionary
- RENAME : rename an object

Data Manipulation Language (DML) statements are used for managing data within schema objects. Some examples:

- SELECT : retrieve data from the database
- INSERT : insert data into a table
- UPDATE : updates existing data within a table
- DELETE : deletes all records from a table, the space for the records remain
- MERGE : UPSERT operation (insert or update)

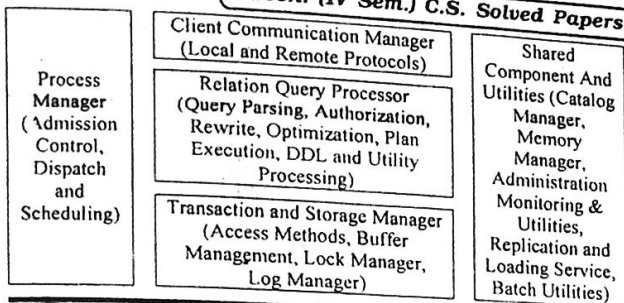
Transaction control statements manage changes made by DML statements. The transaction control statements are:

- COMMIT
- ROLLBACK
- SAVEPOINT
- SET TRANSACTION

All transaction control statements, except certain forms of the COMMIT and ROLLBACK commands, are supported in PL/SQL.

Session control statements dynamically manage the properties of a user session. These statements do not implicitly commit the current transaction.

B.Tech. (IV Sem.) C.S. Solved Papers



Q.15 Contrast between DDL and DML.

[R.T.U. 2017]

OR

Differential between DDL and DML using syntax for them.

[R.T.U. 2014]

OR

Differentiate between DDL and DML. [R.T.U. 2016]

Ans. Difference between DDL and DML : Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE: to create objects in the database
- ALTER : alters the structure of the database
- DROP : delete objects from the database
- TRUNCATE : remove all records from a table, including all spaces allocated for the records are removed
- COMMENT : add comments to the data dictionary
- RENAME : rename an object

Data Manipulation Language (DML) statements are used for managing data within schema objects. Some examples:

- SELECT : retrieve data from the database
- INSERT : insert data into a table
- UPDATE : updates existing data within a table
- DELETE : deletes all records from a table, the space for the records remain
- MERGE : UPSERT operation (insert or update)

Transaction control statements manage changes made by DML statements. The transaction control statements are:

COMMIT
ROLLBACK
SAVEPOINT
SET TRANSACTION

All transaction control statements, except certain forms of the COMMIT and ROLLBACK commands, are supported in PL/SQL.

Session control statements dynamically manage the properties of a user session. These statements do not implicitly commit the current transaction.

Database Management System

PL/SQL does not support session control statements. The session control statements are:

ALTER SESSION

SET ROLE

- CALL : call a PL/SQL or Java subprogram
- EXPLAIN PLAN : explain access path to data
- LOCK TABLE : control concurrency

DDL Commands Syntax

The Create Table command

CREATE TABLE [schema.]table

({ column datatype [DEFAULT expr] [column_constraint] ... | table_constraint }

[, { column datatype [DEFAULT expr] [column_constraint] ... | table_constraint }]...)

[AS subquery]

The DROP Command

Syntax:

DROP TABLE <table_name>

Example:

DROP TABLE Student;

It will destroy the table and all data which will be recorded in it

DML Command Syntax

Viewing data in the table (Select Command) : Once data has been inserted into a table, the next most logical operation would be to view what has been inserted. The SELECT SQL verb is used to achieve this.

All rows and all columns

Syntax: SELECT * FROM Table_name;

eg: Select * from Student; It will show all the table records.

SELECT First_name, DOB FROM STUDENT WHERE Reg_no='S101'; Cover it by single inverted comma if its datatype is varchar or char.

This Command will show one row. because you have given condition for only one row and particular records. If condition which has given in WHERE Clause is true then records will be fetched otherwise it will show no records selected.

Q.16 Discuss types of DBMS.

[R.T.U. 2017]

Ans. Types and Classification of Database Management System : On the basis of data model, we have five types of DBMS:

(1) Relational Database : This is the most popular data model used in industries. It is based on the SQL. They are table oriented which means data is stored in different access control tables, each has the key field whose task is to identify each row. The tables or the files

with the data the row or fields. For source), Or server(Micro

(2) O here is in the programming programming natural data Examples an

(3) O DBMS are incorporating leading to a or object rel

(4) H about the gro in the recor Here the da attached to platforms. registry(Mic

(5) N digital comp database is database, th network o (COMPUT

While types of DB

(i) S can support the personal to a single write the da

(ii) M concurrently database sho is not need student in th his informati related to hi the fee secti student's dat even though departments

Q.34 *What is RDBMS? Describe integrity constraints in relational data model.* [R.T.U. 2013]

Ans. Relational Data Base Management System (RDBMS) : A relational database consists of a collection of tables, each having a unique name. A row in a table represents a relationship among a set of value. A table is a entity set and row is an entity. Thus a table represents a collection of relationships.

There is a direct correspondence between the concept of a table and the mathematical concept of a relation, from which the relational data model takes its name.

Basic Structure

Consider the employee table. It has 3 column headers : emp_ID, emp_Name and salary. If we follows the terminology of the relational data model, we refer to these headers as attributes. For each attributes, there is set of permitted values called the domain of the attribute. For the attribute emp_Name, the domain is the set of all employee names. Let D_1 denote the set of all employee IDs, D_2 the set of all employee name and D_3 the set of all salary.

Then, any row of employee must consist of a 3-tuple (V_1, V_2, V_3) where V_1 is an employee id (i.e. V_1 is in domain D_1) V_2 is a employee name (i.e. V_2 is in domain D_2) and V_3 is a salary (i.e. V_3 is in domain D_3)

$$V_1 \in D_1, V_2 \in D_2, V_3 \in D_3$$

In general employee will contain only a subset of the set of all possible rows. Therefore, employee is a subset of

$$D_1 * D_2 * D_3 \dots$$

In general a table of n- attributes must be a subset of

$$D_1 * D_2 * \dots * D_{n-1} * D_n$$

$$X_{i=1}^n D_i \text{ (all possible rows)}$$

Integrity Constraints

An Integrity constraint is a condition that is enforced automatically by the DBMS and whose violation prevents the data from being stored in the database. The DBMS enforces integrity constraints in that it only permits legal instances to be stored in the database. The key constraints and the referential integrity constraints are identified as the two minimum constraints that must be enforced by the DBMS.

Constraints can be Defined in Two Ways

1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table-level definition.

Example of Integrity Constraints

1. No two account can have the same account number.
2. An account number cannot be null.

(i) **Primary Key** : Refer to Q.3.

(ii) **Referential Integrity** : Refer to Q.2.

(iii) **SQL Not Null Constraint** : This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

Syntax to define a not null constraint :

[CONSTRAINT constraint_name] Not Null

(iv) **SQL Unique Key** : This constraint ensures that a column or a group of columns in each row have a distinct value.

A columns () can have a null value but the values cannot be duplicated.

Syntax to define a unique key at column level :

[CONSTRAINT constraint_name] UNIQUE

Syntax to define a unique key at table level:

[CONSTRAINT constraint_name] UNIQUE
(column_name)

(v) **SQL Check Constraint** : This constraint defines a business rule on a column. All the rows must satisfy this rule. This constraint can be applied for a single column or a group of columns.

Syntax to define a check constraint :

[CONSTRAINT Constraint_name] CHECK
(Condition)

Q.35 Discuss the various type of keys that are used in relational model. [R.T.U. Dec.2013]

Ans. Various types of Keys : A key is a single attribute or a combination of attributes whose values uniquely identify each tuple in that relation.

In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes. A key allows us to identity a set of attributes that are sufficient to distinguish relationship from each other.

There are various types of keys :

- (i) Super Key
- (ii) Candidate Key
- (iii) Primary Key
- (iv) Alternate Key
- (v) Foreign Key

We can understand these keys with the help of following example.

Table : Student

St_Name	St_Roll No.	Aadhar No	Sem	Email	Street	City	Dept No
Dheer	143	71234569	VII	Dheer@xyz.com	Sitapura	Jaipur	5
:	:	:	:	:	:	:	:

Table : Department

Dept No.	Dept Name	HOD	Dept_location
5	CS	M.K.	1st Floor
:	:	:	:

(i) **Super Key** : A Super key is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.

For example :

The St_Roll No. attribute of the student table is sufficient to distinguish are student from another. Thus, St_Roll No. is a superkey. Similarly, the combination of St_Name and St_Roll No. is a super key. The St_Name is not a super key, because several student might have the same name i.e. In our example.(student table) following are Super Keys.

St_Roll No

St_Roll No, St_Name	✓
St_Roll No, City	✓
St_Roll No, Sem	✓
St_Name	×
St_Name, Sem	×
St_Name, Email	✓
St_Name, DeptNo	×
Email	✓
Aadhar No.	✓
Aadhar No, St_Name	✓

Suppose that students from same street having different names then

Street	×
St_Name	×
St_Name, Street	✓
St_Name, Street, City	✓
St_Name, Street, Sem	✓

(ii) **Candidate Key** : Minimal superkeys are called candidate keys.

In our example (student table) following are candidate keys.

St_Roll No	✓
Aadhar No.	✓
Email	✓

And if in our system, students from same street are having different names then,

St_Name, Street	✓
-----------------	---

But these super keys are not candidate keys
 St_Roll No, St_Name × {St_Roll No. alone is sufficient to distinguish.
 St_Name, Street, City × {St_Name, street is sufficient to distinguish.
 St_Roll No., Email × {St_Roll No. or Email is sufficient to distinguish.

i.e. A attribute or a set of minimum attributes which are sufficient to distinguish is called candidate key and combination of other attributes with these attributes can not be a candidate key but it would be a super key.

(iii) **Primary Key** : The candidate key which is never change or really change is choosen as a primary key.

In our example (student table), among four candidate keys (St_Roll No, St_Name, Street, Email, Aadhar No.) The Aadhar No. never change, so we would select Aadhar no. as a primary key.

(iv) **Alternate Keys** : The candidate keys which are not choosen as primary key are called alternate keys.

In our example, remaining candidate keys St_Name, Street, Roll No. Email are alternate keys.

(v) **Foreign Key** : The attribute or set of attributes which are used to link two tables are called foreign key.

In our example, dept No. is a foreign key which is used to link tables student and department. In department table, dept No. is a primary key but it is not a primary key i-student table.

RELATIONSHIP ALGEBRA AND CALCULAS

2

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 Explain the role of Triggers in SQL programming.

[R.T.U. 2019]

Ans. Triggers : A trigger is a statement that the system executes automatically as a side effect of a modification to the database.

Triggers are special type of stored procedures executed automatically when certain events take place. There are different types of triggers for update, for insert and for delete.

Each trigger is associated with a single database table. Triggers are parameterless procedure that are triggered or fired by the event and not by choice. They cannot have parameters to design a trigger mechanism, we must meet two requirements. Those are-

1. Specify when a trigger is to be executed. This is broken up into an event that causes the trigger to be checked and a condition that must be satisfied for trigger execution to proceed.
2. Specify the actions to be taken when trigger executes.

Q.2 What is the meaning of sub-query in terms of SQL?

[R.T.U. 2019]

Ans. SQL - Sub Queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.

Q.3 Consider the relation schema:

Works(person_name, company_name, salary)

Lives(person_name, street, city)

Located-in(company_name, city)

Managers(person_name, manager_name)

where manager_name refers to person_name.

Give the relational algebra for the following queries:

- (i) List the names of the persons work for the company 'SBC' along with the cities they live in.
- (ii) Find the name of the persons who live in the same city and same street as their manager.
- (iii) Find the persons whose salaries are more than the salary of everybody who works for the company 'SBC'.

[R.T.U. 2015]

Ans.

(i) $\Pi_{\text{person_name, city}} (\text{Lives} \bowtie (\sigma_{\text{company_name} = \text{'SBC'}} (\text{Works})))$

(ii) $\Pi_{\text{person_name}} ((\text{Lives} \bowtie \text{Managers}) \bowtie_{(\text{manager_name} = \text{person_name})} (\text{Lives2} \bowtie_{(\text{Lives2.person_name} \wedge \text{Lives2.street} = \text{Lives2.street} \wedge \text{Lives2.city} = \text{Lives2.city})} (\text{Lives})))$

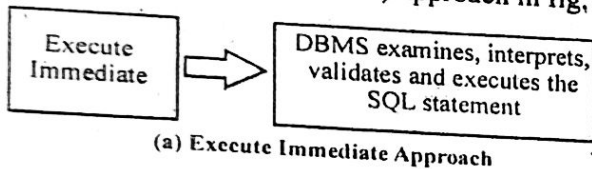
(iii) $\Pi_{\text{person_name}} (\text{Works}) - (\Pi_{\text{person_name}} (\text{Works} \bowtie_{(\text{Works.salary} < \text{Works2.salary} \wedge \text{Works2.company_name} = \text{'SBC'})} \text{Works2} (\text{Works})))$

Q.4 Explain aggregate operators.

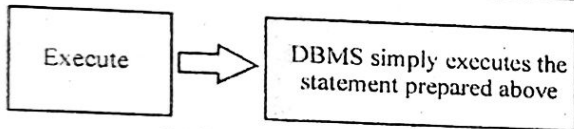
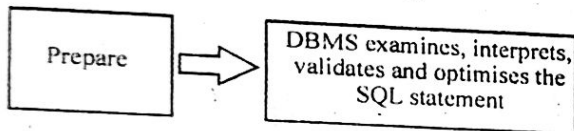
- Performance can be far better. This is because the statement is parsed, validated and optimised far ahead of execution time.
- Using parameters, the same statement can be executed in various ways.

In this model, two statements are required :
PREPARE and EXECUTE.

The PREPARE statement creates an SQL object that contains the SQL statement to be executed. When this statement is executed, the DBMS examines, interprets, validates and optimises it. It is stored until an EXECUTE (and not EXECUTE IMMEDIATE) statement is encountered. At this stage, the statement is actually executed. This approach is contrasted with the earlier (EXECUTE IMMEDIATE) approach in fig.



(a) Execute Immediate Approach



(b) Prepare-Execute Approach

Fig. : Dynamic SQL approaches

Q.7 Write SQL queries for following operations:

- Create student registration table and insert records in it.
- Update records based on a key.
- Display name and Roll numbers of students who have scored more than 60% marks.
- Delete records and table. [R.T.U. 2017]

Ans.(a) Create Student registration table and insert records in it :

```
CREATE TABLE Student_Reg
```

```
(
  RollNo int,
  Name Varchar(50),
  Course Varchar(10),
  Branch Varchar(10),
```

Database Management System

Score float,

PRIMARY KEY(RollNo)

);

INSERT into Student_Reg

VALUES (1, Alok, B.Tech, CSE, 70.5);

Ans.(b) Update records based on a key

UPDATE Student_Reg

SET Course = 'M.Tech'

WHERE Marks > 70.0;

Ans.(c) Display name and roll no of students who have scored more than 60% marks

SELECT RollNo, Name

FROM Student_Reg

WHERE Marks > 60;

Ans.(d) Delete records and table

//Delete Records

DELETE from Student_Reg

WHERE Marks < 33

//Delete Table

DROP Student_Reg

Q.8 Consider the employee database given below :

Employee (emp_name, street, city)

Works (emp_name, company_name, salary)

Company (company_name, city)

Manager (emp_name, manager_name)

Give an expression in SQL for each of the following queries :

- Modify the database so that Jones now lives in New town.
- Give all managers of first bank corporation a 10 Percent raise.
- Give all managers of First bank corporation a 10 percent raise unless the salary becomes greater than \$100,000. In such cases give only a 3 percent raise.
- Find the names of all employees in the database who live in the same city as the company for which they work. [R.T.U. 2016]

Ans.(i) update employee

set city='newtown'

where emp_name = 'jones';

returned by the subquery with the expression in the WHERE clause.

Consider following payments table :

- Payments(customerNumber, checkNumber, paymentDate, amount)

Consider the following query :

```
SELECT customerNumber,
       checkNumber,
       amount
FROM payments
WHERE amount = (
    SELECT MAX(amount)
    FROM payments
);
```

Q.11 Explain Relationship algebra joins.

[R.T.U. 2014]

[Note : This should be relational algebra joins.]

OR

Discuss the difference between five join operation: theta join, equi join, natural join, outer join and semi join.

[R.T.U. Dec.2013]

Ans. Relational Algebra Join Operations

Join is combination of cartesian product followed by selection process. Join operation pairs two tuples from different relations if and only if the given join condition is satisfied.

Following are the different types of joins:

1. Theta Join
2. Equi Join
3. Natural Join
4. Outer Join
5. Semi Join

A theta join allows for arbitrary comparison relationships (Such as \geq).

- An equi join is a theta join using the equality operator.
- A natural join is an equi join on attributes that have the same name in each relationship.
- We will now discuss them one by one to understand the difference between them.

1. Theta (θ) Join

Theta join is the join condition. Theta joins combines tuples from different relations provided they satisfy the theta condition.

Database Management System

Notation :

$R1 \bowtie_{\theta} R2$

$R1$ and $R2$ are relations with their attributes ($A1, A2, \dots, An$) and ($B1, B2, \dots, Bn$) such that no attribute matches that is $R1 \cap R2 = \phi$. Here θ is condition in form of set of conditions C .

Theta join can use all kinds of comparison operators.

Table : Student Relation

SID	Name	Std
101	Dheer Singh	10
102	Saveen	11

Table : Subjects Relation

Class	Subject
10	Math
10	English
11	Music
11	Sports

Student_Detail = STUDENT \bowtie Student_Sid = Subject_Class
SUBJECT

Table : Output of Theta Join

SID	Name	Std	Class	Subject
101	Dheer Singh	10	10	Math
101	Dheer Singh	10	10	English
102	Sawra	11	11	Music
102	Sawra	11	11	Sports

2. Equi-Join

When Theta join uses only equality comparison operator it is said to be Equi-Join. The above example corresponds to equi-join.

3. Natural Join (\bowtie)

Natural join does not use any comparison operator. It does not concatenate the way Cartesian product does. Instead, Natural Join can only be performed if there is at least one common attribute exists between relation. Those attributes must have same name and domain.

Natural join acts on those matching attributes where the values of attributes in both relation is same.

Table 1 : Relation Courses

SID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME ₂
EE01	Electronics	EE

Table 2 : Relation HOD

Dept	Head
CS	Shailesh Aravatiya
ME	Kavita
EE	P. Mangal

Table 3 : Relation Courses \bowtie HOD

Dept	CID	Course	Head
CS	CS01	Database	Shailesh Aravatiya
ME	ME01	Mechanics	Kavita
EE	EE01	Electronics	P. Mangal

4. Outer Joins

All joins mentioned above, that is Theta Join, Equi Join and Natural Join are called inner-joins. An inner-join process includes only tuples with matching attributes, rest are discarded in resulting relation. There exists methods by which all tuples of any relation are included in the resulting relation.

There are three kinds of outer joins :

(a) Left Outer Join ($R \bowtie_{LO} S$)

All tuples of Left relation, R , are included in the resulting relation and if there exists tuples in R without any matching tuple in S then the S -attributes of resulting relation are made NULL.

Table : Left Relation

A	B
100	Database
101	Mechanics
102	Electronics

Table : Right Relation

A	B
100	Shailesh Aravatiya
102	Kavita
104	P. Mangal

Table : Left Outer Join Output Courses \bowtie_{LO} HOD

A	B	C	D
100	Database	100	Shailesh Aravatiya
101	Mechanics	—	—
102	Electronics	102	Kavita

(b) Right Outer Join ($R \bowtie_{RO} S$)

All tuples of the Right relation, S , are included in the resulting relation and if there exists tuples in S without any matching tuples in R then the R -attributes of resulting relation are made NULL.

Table : Right Outer Join Output

A	B	C	D
100	Database	100	Shailesh Aravatiya
102	Electronics	102	Kavita
—	—	104	P. Mangal

(c) Full Outer Join : $(R \bowtie S)$

All tuples of the both participating relations are included in the resulting relation and if there no matching tuples for both relations, their respective unmatched attributes are made NULL.

Table : Full Outer Join Output Courses \bowtie HOD

A	B	C	D
100	Database	100	Shailesh Aravatiya
101	Mechanics	—	—
102	Electronics	102	Kavita
—	—	104	P. Mangal

5. Semi Join

In semi join, first we take the natural join of two relations then we project the attributes of first table only. So after join and matching the common attribute of both relations only attributes of first relation are projected.

Example :

Table : Faculty

facId	facName	Dept	salary	rank
F234	Monika	CSE	21000	lecturer
F235	Amidya	ENG	23000	Asso Prof
F236	Neelam	CSE	27000	Asso Prof
F237	Manju	IT	32000	Professor

Table : Course

CrS Code	CrS Title	FId
C3456	TOC	F234
C3457	FM	
C3458	DBMS	F236
C3459	OS	F237

If we take the semi join of two relations faculty and course then the resulting relation would be as under :

Table : Semi join operation on Faculty \bowtie Course

facId	facName	Dept	Salary	Rank
F234	Monika	CSE	21000	lecturer
F236	Neelam	CSE	27000	Asso Prof
F237	Manju	IT	32000	Professor

DBMS.33

Table 1

DeptNo	Avg of Salary	Dept No
1	15000	1
3	9833	3
4	9667	4

Table 2

Count of Employees	Max of Salary
1	1200
3	15000
3	11500

PART-C

Q.14 What is embedded SQL?

Write the following queries in SQL by considering the employee data base ...

- (a) Find all the employees in database who live in the same cities as the companies for which they work.
- (b) Find all the employees who earn more than the average salary. [R.T.U. 2019]

Ans. Embedded SQL : Embedded SQL is a method of inserting inline SQL statements or queries into the code of a programming language, which is known as a host language. Because the host language cannot parse SQL, the inserted SQL is parsed by an embedded SQL preprocessor. Embedded SQL is a robust and convenient method of combining the computing power of a programming language with SQL's specialized data management and manipulation capabilities. The SQL standard defines embedding of SQL as embedded SQL and the language in which SQL queries are embedded is referred as host language. The result of the query is made available to the program one tuple (record) at a time. To identify embedded SQL requests to the preprocessor, we use EXEC SQL statement.

EXEC SQL, embedded SQL statement, END-EXEC
Note : A semi-colon is used instead of END-EXEC when SQL is embedded in C or Pascal.

Embedded SQL statements: declare cursor, open, and fetch statements.

DBMS.34

EXEC SQL

declare c cursor for
select cname, ccity
from deposit, customer
where deposit.cname = customer.cname
and deposit.balance > :amount
END-EXEC

where amount is a host-language variable.
EXEC SQL open c END-EXEC

This statement causes the DB system to execute the query and to save the results within a temporary relation. A series of fetch statement are executed to make tuples of the results available to the program.

Need to Access a Database Using a General Purpose Programming Language : Impedance mismatch is the term used to refer to the problems that occur because of differences between the database model and the programming language model.

Impedance mismatch is less of a problem when a special database programming language is designed that uses the same data model and data types as the database model. One example of such a language is Oracle's PLUSQL. For object databases, the object data model is quite similar to the data model of the Java programming language, so the impedance mismatch is greatly reduced when Java is used as the host language for accessing a Java-compatible object database. Several database programming languages have been implemented as research prototypes.

Most database access in practical situations is through software programs that implement database applications. This software is usually developed in a general purpose programming language such as Java, COBAL or C/C++. The database language such as SQL is, therefore, embedded into general-purpose programming language for accessing the database. When database statements are included in a program, the general-purpose programming language is called the host language, whereas the database language-SQL, in our case is called the data sub-language.

Embedding database commands in a general-purpose programming language : In this approach, database statements are embedded into the host programming language, but they are identified by a special prefix. For example, the prefix for embedded SQL is the string EXEC SQL, which precedes all SQL commands in a host language program. A precompiler or preprocessor scans the source program code to identify database statements and extract them for processing by the DBMS. They are

B.Tech. (IV Sem.) C.S. Solved Papers

replaced in the program by function calls to the DBMS-generated code. This technique is generally referred to as embedded SQL.

(a) select E.person_name
from Employee as E, Works as W, Company as C
where E.person_name=W.person_name and
E.city=C.city

and W.company_name=C.company_name

(b) SELECT * FROM employees

WHERE salary >

ALL(SELECT avg(salary) FROM employees
GROUP BY department_id);

Q.15 Explain following operations in relational algebra:

- (a) Selection
(b) Projection
(c) Join
(d) Rename.

[R.T.U. 2017]

Ans.(a) Select Operation : The select operation selects tuples that satisfies a given predicate. We use the lowercase Greek letter sigma (σ) to denote selection. The predicate appears as a subscript to σ . The argument relation is in parenthesis after the σ .

Loan-number	Branch-Name	Amount
L - 11	Round Hill	900
L - 14	Down town	1500
L - 15	Perryridge	1500
L - 16	Perryridge	1300
L - 17	Downtown	1000
L - 23	Red wood	2000
L - 93	Mianus	500

To select those tuples of the loan relation where branch is "Perryridge".

$\sigma_{\text{branch_name} = \text{"Perryridge"}}(\text{loan})$

We can find all tuples in which the amount loan is more than \$1200.

$\sigma_{\text{amount} > 1200}(\text{loan})$

Thus to find those tuples pertaining to loan of more than \$1200 made by the Perryridge branch,

$\sigma_{\text{branch_name} = \text{"Perryridge"}}(\sigma_{\text{amount} > 1200}(\text{loan}))$

Ans.(b) Project (π) : We have a schema named the loan scheme before which we give a formal definition of the tuple relational calculus, we return to some of the queries for which we wrote relational.

PART-A

Q.1 *What is the purpose of normalization in DBMS?*

[R.T.U. 2019]

OR

Discuss the need of normalization.

Ans. Need of Normalization : When you normalize a database, you have four goals: arranging data into logical groupings such that each group describes a small part of the whole; minimizing the amount of duplicate data stored in a database; organizing the data such that, when you modify it, you make the change in only one place; and building a database in which you can access and manipulate the data quickly and efficiently without compromising the integrity of the data in storage.

Data normalization helps you design new databases to meet these goals or to test databases to see whether they meet the goals. Sometimes database designers refer to these goals in terms such as data integrity, referential integrity, or keyed data access. Ideally, you normalize data before you create database tables. However, you can also use these techniques to test an existing database.

Q.5 Explain Decomposition.

Ans. Decomposition : A functional decomposition is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail. In decomposition, one function is described in greater detail by a set of other supporting functions.

The decomposition of a relation scheme R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R .

Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.

There are two types of decomposition :

1. Lossy Decomposition
2. Lossless Join Decomposition

PART-B

Q.6 What is Normalization? Also explain functional dependencies with a suitable example.

[R.T.U. 2019]

Ans. Normalization : Refer to Q.3.

Functional Dependencies : The single most important concept in a relational schema design is that of functional dependency. A functional dependency is a constraint between two set of attributes in a relation from a database.

Given a relation R a set of attributes x in R is said to functionally determine. Another attribute y also in R , (written $x \rightarrow y$) if and only if each x value is associated with at most one y value. We call x the determinant set and y the dependent attribute. Thus given a tuple and the values of the attribute in x . One can determine the corresponding value of the y attribute.

A functional depending set ' S ' is irreducible if the set has three following properties :

1. Each right set of a functional dependency of ' S ' contains only one attribute.
2. Each left set of functional dependency of ' S ' is irreducible. It means that reducing any one attribute from left set would not change the content of S .
3. Reducing any functional dependency will change the content of S .

A functional dependency, denoted by $x \rightarrow y$, between two sets of attributes x and y that are subsets of the attributes of relation R .

Functional dependencies and keys : We say that a set of one or more attributes $\{A_1, \dots, A_n\}$ is a key for a relation R if :

1. Those attributes functionally determine all other attributes of the relation.
2. No proper subset of those attributes functionally determines all other attributes of R .

A set of attributes that contains a key is called a superkey. Thus every key is a superkey but not every key is minimal. If a relation has more than one key, one of the keys is designed as the **primary key**.

Closures : Let a relation ' R ' has some functional dependencies ' F ' specified. The closure is the set of all functional dependencies that may be logically derived from F . ' F ' is the set of most obvious and important functional dependencies and F^+ , the closure is the set of all the functional dependencies including F and those that can be deduced from F^+ . The closure is important and may be needed in finding one or more candidate keys of the relation.

A set of rules that may be used to infer additional dependencies was proposed by Armstrong in 1974. These rules are complete set of rules called Axioms or inference rules such that all possible functional dependencies may be derived from them. The rules are :

1. **Reflexivity Rule :** If ' x ' is a set of attributes and ' y ' is a subset of ' x ', then $x \rightarrow y$ holds.
2. **Augmentation Rule :** If $x \rightarrow y$ holds and ' w ' is the set of attributes, then $wx \rightarrow wy$ holds.
3. **Transitivity Rule :** If $x \rightarrow y$ and $y \rightarrow z$ hold, then $x \rightarrow z$ holds.

These rules are called Armstrong's Axioms.

The most important additional axioms are :

1. **Union Rule :** If $x \rightarrow y$ and $x \rightarrow z$ hold then $x \rightarrow yz$ holds.
2. **Decomposition Rule :** If $x \rightarrow yz$ holds, so do $x \rightarrow y$ and $x \rightarrow z$.
3. **Pseudotransitivity Rule :** If $x \rightarrow y$ and $wy \rightarrow z$ hold then so does $wx \rightarrow z$.

Full Functional Dependencies (FFD):

A functional dependency $X \rightarrow Y$ is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

If all the non-key attributes of the entity completely and functionally depend on the key attribute of the same entity so it is known as full Functional Dependencies.

For example, we have Student table (Entity) having 4 column (attribute).

- (1) Sid
- (2) Sname
- (3) Add
- (4) CourseName

The Sname, Add and CourseName are the non-key attribute which completely depend on Sid (Key Attribute).

If we want to retrieve any information about student we only need key attribute i.e. Sid and rest of the information we can get on the basis of key attribute.

So here all the attribute (Sname, Add, CourseName) fully depend on key attribute, and hence it is called full Functional Dependencies.

Q.7 Explain functional dependencies with the help of suitable examples. [R.T.U. 2017]

OR

Describe the concept of full functional dependency. [R.T.U. 2015]

OR

Define Functional Dependency. Explain Armstrong's axioms or rules, with examples. [R.T.U. 2015]

OR

Describe the concept of full functional dependency (FFD). [R.T.U. Dec. 2013, 2008]

Ans. Functional Dependencies : Refer to Q.6.

Q.8 What is Decomposition? Explain Lossy and Lossless join decomposition. [R.T.U. 2016]

Ans. Decomposition : Refer to Q.5.

Lossy Decomposition : "The decomposition of relation R into R1 and R2 is lossy when the join of R1 and R2 does not yield the same relation as in R."

One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.

Consider that we have table STUDENT with three attributes roll_no, sname and department.

STUDENT:

Roll no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relations no_name and name_dept :

No_name	Name_dept
Roll_no	Sname
111	parimal
222	parimal

Sname	Dept
parimal	COMPUTER
parimal	ELECTRICAL

In lossy decomposition, spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
111	parimal	ELECTRICAL
222	parimal	COMPUTER
222	parimal	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

Lossless Join Decomposition : "The decomposition of relation R into R1 and R2 is lossless when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.

This is also referred as non-additive decomposition.

The lossless-join decomposition is always defined with respect to a specific set F of dependencies.

Consider that we have table STUDENT with three attributes roll_no, sname and department.

STUDENT :

Roll no.	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name	Stu_dept
Roll no	Sname
111	parimal
222	parimal

matter which other attributes appear in the FD. Allow * to represent any set of attributes in R, then F^+ is $BD \rightarrow B$, $BD \rightarrow D$, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow BD$, $B \rightarrow D$, $B \rightarrow B$, $B \rightarrow BD$, and all FDs of the form $A^* \rightarrow \alpha$, $BC^* \rightarrow \alpha$, $CD^* \rightarrow \alpha$, $E^* \rightarrow \alpha$ where α is any subset of $\{A, B, C, D, E\}$. The candidate keys are A, BC, CD, and E.

PART-C

Q.11 Explain Boyce-Codd normal form and 3-NF in detail. [R.T.U. 2019]

Ans. BCNF (Boyce-Codd Normal Form)

One of the most desirable normal forms that we can obtain is Boyce-Codd Normal Form (BCNF). A relation schema R is in BCNF with respect to a set of functional dependencies if for all functional dependencies in F^+ of a form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$).
- α is a superkey for schema R.

A database design is in BCNF if each member of the set of relation schema that constitutes the design is in BCNF.

Often testing of a relation to see if it satisfies BCNF can be simplified.

- To check if a nontrivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF, compute α^+ and verify that it includes all attributes of R; that is, it is a superkey of R.
- To check if a relation schema R is in BCNF, it suffices to check only the dependencies in the given set F for violation of BCNF, rather than to check all dependencies in F^+ .

Example of BCNF

Stud-ID	S-Name	Subject	Grade
1001	Axay	Physics	A
1001	Axay	Chemistry	C
1001	Axay	Maths	C
1002	Sparsh	Physics	A
1002	Sparsh	Chemistry	A
1002	Sparsh	Maths	B

In this relation following FDs exist:

S_Name, Subject \rightarrow Grade
Stud_ID, Subject \rightarrow Grade

S_Name \rightarrow Stud_ID
Stud_ID \rightarrow S_Name

Moreover, in this relation two candidate keys (S_Name, Subject) and (Stud_ID, Subject) exist, which are composite keys and contain a common attribute subject. This relation is in 3NF, however a lot of data repetition is there in terms of S-Name and Stud_ID. So for this relation to be in BCNF, we will have to do the decomposition. The rule for decomposition for a relation R, which is not in BCNF, is as follows:

Let $x \subseteq R$, A be the single attribute in R_x and $x \rightarrow A$ be a FD that causes a violation of BCNF. Decompose R into $R - A$ and R_A .

So for above relation to be in BCNF, the decomposition will be as follows:

Stud_ID	S_Name
1001	Axay
1002	Sparsh

Third Normal Form (3NF): There are schemas where a BCNF decomposition cannot be dependency preserving. For such schemas, we have two alternatives if we wish to check if an update violates any functional dependencies:

- Pay the extra cost of computing joins to test for violations.
- Use an alternative decomposition, third normal form (3NF), which we present below, which makes testing of updates cheaper, unlike BCNF, 3NF decompositions may contain some redundancy in the decomposed schema.

Definition: BCNF requires that all nontrivial dependencies be of the form $\alpha \rightarrow \beta$, where α is a superkey. 3NF relaxes this constraint slightly by allowing nontrivial functional dependencies whose left side is not a superkey.

A relation schema R is in third normal form (3NF) with respect to a set F of functional dependencies if for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$ at least one of the following holds:

- $\alpha \rightarrow \beta$ is a trivial functional dependency.
- α is a superkey for R.
- Each attribute A in $\beta \rightarrow \alpha$ is contained in a candidate key for R.

The two alternatives are the same as the two alternatives in the definition of BCNF. The third alternative of the 3NF definition seems rather unintuitive and it is not obvious why it is useful. It represents, in some sense, a

DBMS-50

minimal relaxation of the BCNF conditions that helps to ensure that every schema has a dependency preserving decomposition into 3NF. Its purpose will become more clear.

Third Normal Form:

Following synthesis algorithm, R_1 has a candidate key

$R_1 = ISQ, R_2 = SD, R_3 = IB, R_4 = BO$

decomposition of R into R_1, R_2, R_3 and R_4 is lossless and dependency preserving.

Q.12 (a) Explain 3rd NF with suitable example. [R.T.U. 2017]

(b) Explain BCNF with suitable example. [R.T.U. 2017]

Ans.(a) Refer to Q.11.
Ans.(b) Refer to Q.11.

Q.13 Discuss the purpose of BCNF and describe how BCNF different from 3NF. Provide an example to illustrate your answer. [R.T.U. 2016, 2012]

OR

Why BCNF to be considered stricter than 3NF? Explain decomposition of non-BCNF scheme into BCNF scheme. [R.T.U. 2015]

OR

Define BCNF. How does it differ from 3NF? Why it is considered a stronger form of 3NF?

[R.T.U. Dec. 2013, 2008]

OR

Why BCNF considered to be stricter than 3NF? How is non BCNF scheme decomposed into BCNF scheme? [R.T.U. 2011]

OR

Discuss the purpose of BCNF and describe how BCNF different from 3NF. Provide an example to illustrate your answer.

[R.T.U. 2010; Raj. Univ. 2005, 2003]

Ans. BCNF (Boyce-Codd Normal Form): Refer to Q.11.

Third Normal Form (3NF): Refer to Q.11.

Comparison of BCNF and 3NF: Of the two normal forms for relational database schemas, 3NF and BCNF, there are advantages to 3NF in that we know that it is always possible to obtain a 3NF design without sacrificing a lossless-join or dependency preservation. Nevertheless, there are disadvantages to 3NF. If we do not eliminate all transitive relation schema dependencies, we may have to use null values to represent some of the possible meaningful

TRANSACTION PROCESSING

4

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 What is the need of serializability in transaction processing? [R.T.U. 2019]

OR

What is serializability?

Ans. Serializability is the classical concurrency scheme. It ensures that a schedule for executing concurrent transactions is equivalent to one that executes the transactions serially in some order.

Q.2 What is concurrency? [R.T.U. 2019]

OR

What is a concurrent execution of transaction?

Ans. Concurrent execution of transaction means multiple transactions execute/run concurrently in RDBMS with each transaction doing its atomic unit of work for the operations encapsulated in the particular transaction.

Q.3 What is cascadeless schedule?

Ans. A cascadeless schedule (also known as recoverable schedule) is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i , the commit operation of T_i appears before the read operation of T_j .

Q.4 What is graphical form of recoverable schedule?

Ans.

If:

T_1	$w_1(A) \rightarrow r_k(A)$	
T_2	$w_2(B) \rightarrow r_k(B)$	T_k
T_3	$w_3(C) \rightarrow r_k(C)$	

Then:

T_1 commit	\rightarrow	T_k commit
T_2 commit	\rightarrow	
T_3 commit	\rightarrow	

The transaction T_k must commit only after the transactions T_1 , T_2 and T_3 have committed.

Q.5 What is use of serialization graph?

Ans. Serialization graph is used to test the serializability of a schedule.

PART-B

Q.6 Write a short note on transaction properties and recoverable schedules. ACID [R.T.U. 2019]

Ans. Transaction Properties : The four basic properties of a Transaction Processing System are referred to as the ACID properties. Atomicity, Consistency, Isolation and Durability. These properties are used to test that a transaction is never unfinished, the data held in the system

is always consistent, concurrent transactions are independent and the effects of a transaction continue after the transaction is completed.

Atomicity : The transaction must happen completely or it should be undone completely. If a transaction fails, the effects of all operations that make up the transaction need to be cancelled and the data needs to be returned to its original previous state. If we look at the first part of the word "Atomicity", atom, the term suggests its meaning at the time the concept was first used. An atom is considered the smallest item of matter that can exist.

An example of how a transaction processing system handles atomicity is a student making a purchase from the school canteen. Money changes hands and the student later realizes that the drink is past its use by date. As there are no other drinks available, the student returns the drink and the canteen refunds the purchase price. This transaction is not recorded and so, no transaction occurred.

Consistency : For a successful transaction to take place, all parties must agree on the facts of the exchange. When a successful transaction is completed, the data should be in a consistent state. This means that all data should be able to be accounted for and that each step of the transaction is carried out in the same way each time. This ensures that data is correct for each part of the transaction.

An example of this property would see the sale of two drinks in the canteen should increase the takings by the canteen and decrease the stock held in the canteen by two drinks. The steps are to exchange the drinks for money. The two should balance so that the data is consistent.

Durability : The effects of a completed transaction should always be durable. In a large organization, transactions need to be recorded so that they can be checked at any time in the future. Backups need to be kept of this data. This is also part of an organizations responsibility to pay tax and meet other obligations.

Isolation : Transaction must be independent of each other. This doesn't actually happen. The user just believes that it does. Isolation involves the appearance of treating each transaction separately and keeping the data for each transaction separate.

The aim of isolation is to prevent the same data being treated twice. For example, an organization selling

tickets to rock concerts such as Ticketed need to ensure they do not sell the same seat to more than one person. There may be many outlets selling tickets as well as the online booking website. Customers who access the online booking site need to be sure that their transaction is separate. It is vital that no one else can book the same seat through another agent. Isolation is a bigger issue as the Transaction Processing System becomes larger as it becomes difficult to make it appear as if each transaction is being handled sequentially.

Recoverable Schedule : Refer to Q.3 and Q.4.

Q.7 What is the states of transactions? Explain briefly.

Ans. A transaction goes through many different states throughout its life cycle. These states are called as transaction states. Transaction states are as follows :

- (i) Active state
- (ii) Partially committed state
- (iii) Committed state
- (iv) Failed state
- (v) Aborted state
- (vi) Terminated state

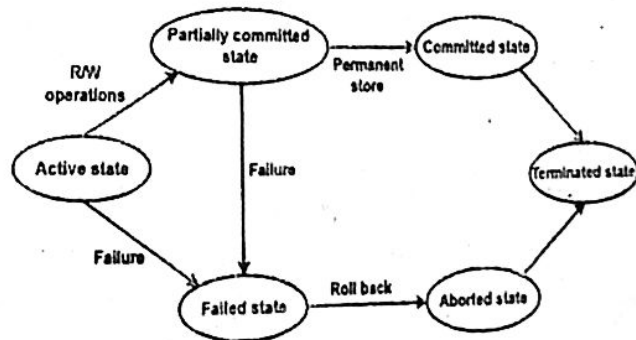


Fig. Transaction States in DBMS

(i) Active State : This is the first state in the life cycle of a transaction. A transaction is called in an active state as long as its instructions are getting executed. All the changes made by the transaction now are stored in the buffer in main memory.

(ii) Partially Committed State : After the last instruction of transaction has executed, it enters into a partially committed state. After entering this state, the transaction is considered to be partially committed. It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

(iii) **Committed State:** After all the changes made by the transaction have been successfully stored into the database, it enters into a committed state. Now, the transaction is considered to be fully committed.

Note : After a transaction has entered the committed state; it is not possible to roll back the transaction. In other words, it is not possible to undo the changes that have been made by the transaction. This is because the system is updated into a new consistent state. The only way to undo the changes is by carrying out another transaction called as compensating transaction that performs the reverse operations.

(iv) **Failed State :** When a transaction is getting executed in the active state or partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into a failed state.

(v) **Aborted State :** After the transaction has failed and entered into a failed state, all the changes made by it have to be undone. To undo the changes made by the transaction, it becomes necessary to roll back the transaction. After the transaction has rolled back completely, it enters into an aborted state.

(vi) **Terminated State :** This is the last state in the life cycle of a transaction. After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end.

Q.8 What is the properties of transaction processing?

Ans. Refer to Q.6.

PART-C

Q.9 Explain conflict v/s view serializability in detail.

[R.T.U. 2019]

Ans. Conflict Serializability : Instructions I_i and I_j , of transactions T_i and T_j respectively, conflict if and only if there exists some item P accessed by both I_i and I_j , and atleast one of these instructions wrote P .

Consider the below operations :

- $I_i = \text{read}(P)$, $I_j = \text{read}(P)$. I_i and I_j don't conflict.
- $I_i = \text{read}(P)$, $I_j = \text{write}(P)$. They conflict.
- $I_i = \text{write}(P)$, $I_j = \text{read}(P)$. They conflict.
- $I_i = \text{write}(P)$, $I_j = \text{write}(P)$. They conflict.

A conflict between I_i and I_j forces a temporal order between them.

If I_i and I_j are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

If a schedule S can be transformed in to a schedule S' by a series of swaps of non-conflicting instructions, then S and S' are conflict equivalent.

In other words a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

Example of a schedule that is not conflict serializable:

T3	T4
Read(P)	
	Write(P)
Write(P)	

The instructions cannot be swapped in the above schedule to obtain either the serial schedule $\langle T3, T4 \rangle$, or the serial schedule $\langle T4, T3 \rangle$.

A serial schedule $T2$ follows $T1$, by a series of swaps of non-conflicting instructions making the below Schedule conflict serializable.

T1	T2
Read(X)	
Write(X)	
	Read(X)
	Write(X)
Read(Y)	
Write(Y)	
	Read(Y)
	Write(Y)

View Serializability : S and S' are view equivalent if the following three conditions are met:

- For each data item P , if transaction T_i reads the initial value of P in schedule S , then transaction T_i must, in schedule S' , also read the initial value of P .
- For each data item P , if transaction T_i executes $\text{read}(P)$ in schedule S , and that value was produced by transaction T_j , then transaction T_i must in

schedule S' also read the value of P that was produced by transaction T_j .

- iii. For each data item P , the transaction that performs the final write(P) operation in schedule S must perform the final write(P) operation in schedule S' .

View equivalence is also based purely on reads and writes alone.

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

Every conflict serializable schedule is also view serializable.

Every view serializable schedule which is not conflict serializable has blind writes.

T3	T4	T6
Read(P)		
	Write(P)	
Write(P)		
		Write(P)

Q.10 What is cascadeless schedule? Why is cascadeless ness of schedules desirable? Are there any circumstances under which it would be desirable to allow non-cascadeless schedules? Explain and justify your answer.

[R.T.U. 2019]

Ans. Cascadeless schedule : Refer to Q.3.

Recoverability : A recoverable schedule is one where, for each pair of Transaction T_i and T_j such that T_j reads data item previously written by T_i , the commit operation of T_i appears before the commit operation T_j .

T8	T9	
read(A)		T9 is dependent on T8
write(A)		Non recoverable schedule if T9 commits before T8
	read(A)	
read(B)		

Suppose that the system allows T9 to commit immediately after execution of read(A) instruction. Thus T9 commit before T8 does.

Now suppose that T8 fails before it commits. Since T9 has read the value of data item A written by T8 we must abort T9 to ensure transaction Atomicity.

However, T9 has already committed and cannot be aborted. Thus we have a situation where it is impossible to recover correctly from the failure of T8.

Cascadeless schedules

T10	T11	T12
read(A) read(B) write(A)	read(A) write(A)	read(A)

Transaction T10 writes a value of A that is read by Transaction T11. Transaction T11 writes a value of A that is read by Transaction T12. Suppose at this point T10 fails. T10 must be rolled back, since T11 is dependent on T10, T11 must be rolled back, T12 is dependent on T11, T12 must be rolled back.

This phenomenon, in which a single transaction failure leads to a series of transaction rollbacks is called Cascading rollback.

- Cascading rollback is undesirable, since it leads to the undoing of a significant amount of work.
- It is desirable to restrict the schedules to those where cascading rollbacks cannot occur, Such schedules are called Cascadeless Schedules.
- Formally, a cascadeless schedule is one where for each pair of transaction T_i and T_j such that T_j reads data item, previously written by T_i , the commit operation of T_i appears before the read operation of T_j .

Every Cascadeless schedule is also recoverable schedule.

Q.11 Describe conflict serializability and view serializability with examples?

Ans. Refer to Q.9.

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 How many kinds of locks present in lock-based protocols?

Ans. Locks are of two types :

Binary Locks : A lock on a data item can be in two states; it is either locked or unlocked.

Shared/Exclusive : This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

Q.2 What is the definition of timestamp-based protocols?

Ans. The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp. Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would

be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one. In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Q.3 Write down the three phase present in the validation based protocol?

Ans. In the validation based protocol, the transaction is executed in the following three phases:

(i) **Read phase** : In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database.

(ii) **Validation phase** : In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability.

(iii) **Write phase** : If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back.

Q.4 What is deadlock handling?

Ans. Deadlock is a state of a database system having two or more transactions, when each transaction is waiting for a data item that is being locked by some other

Q.10 Define the whole concurrency control types and examples?

Ans. In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories.

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols: Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds.

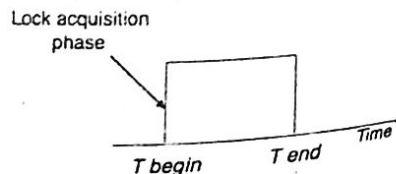
Binary Locks : A lock on a data item can be in two states; it is either locked or unlocked.

Shared/exclusive : This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

There are four types of lock protocols :

Simplistic Lock Protocol: Simplistic lock-based protocols allow transactions to obtain a lock on every object before a 'write' operation is performed. Transactions may unlock the data item after completing the 'write' operation.

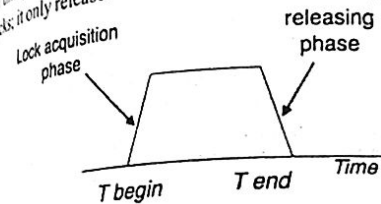
Pre-claiming Lock Protocol: Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks. Before initiating an execution, the transaction requests the system for all the locks it needs beforehand. If all the locks are granted, the transaction executes and releases all the locks when all its operations are over. If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.



DBMS.61

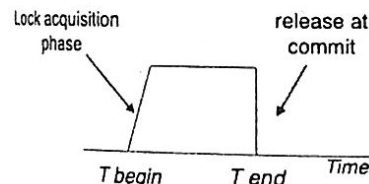
DBMS.62

Two-Phase Locking 2PL: This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.



Two-phase locking has two phases, one is growing, where all the locks are being acquired by the transaction; and the second phase is shrinking, where the locks held by the transaction are being released. To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking: The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



Strict-2PL does not have cascading abort as 2PL does.

Timestamp-based Protocols: The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the

B.Tech. (IV Sem.) C.S. Solved Papers

transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one. In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Timestamp Ordering Protocol : The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and writes operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

The timestamp of transaction T_i is denoted as $TS(T_i)$.

Read time-stamp of data-item X is denoted by $R\text{-timestamp}(X)$.

Write time-stamp of data-item X is denoted by $W\text{-timestamp}(X)$.

Timestamp ordering protocol works as follows "

If a transaction T_i issues a read(X) operation "

If $TS(T_i) < W\text{-timestamp}(X)$

Operation rejected.

If $TS(T_i) \geq W\text{-timestamp}(X)$

Operation executed.

All data-item timestamps updated.

If a transaction T_i issues a write(X) operation "

If $TS(T_i) < R\text{-timestamp}(X)$

Operation rejected.

If $TS(T_i) \geq W\text{-timestamp}(X)$

Operation rejected and T_i rolled back.

Otherwise, operation executed.

Thomas' Write Rule : This rule states if $TS(T_i) < W\text{-timestamp}(X)$, then the operation is rejected and T_i is rolled back.

Time-stamp ordering rules can be modified to make the schedule view serializable.

Instead of making T_i rolled back, the 'write' operation itself is ignored.

□□□