MAY-2017

Database Management System (MCA-204)

1(a) What does defining manipulating and sharing of a database mean?

Sol: **Database** is a collection of related data or database is an integrated collection of data records, files and other database objects.

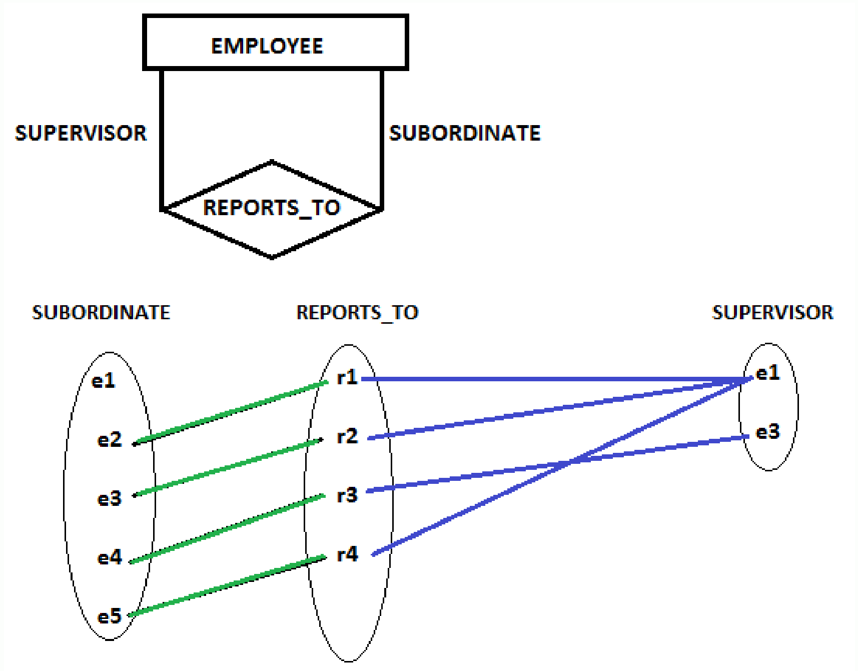
Defining : specifying the data types, structures, and constraints for the data.

Manipulating :includes querying the database to retrieve specific data, updating the data to reflect changes in data, generating reports from data.

Sharing : allows multiple users to access the database concurrently.

1(b) what is recursive relationship type? Give an example.

Sol: If the same entity type participate more than once in a relationship type in different roles then such relationship types are called recursive relationship. For example, in the below figure REPORTS_TO is a recursive relationship as the Employee entity type plays two roles – 1) Supervisor and 2) Subordinate.



1(c) Discuss the characteristics of relations that make a relation different from file.

- No redundant data – Redundancy removed by data normalization
- Data Consistency and Integrity – data normalization takes care of it too
- Secure – Each user has a different set of access

- Privacy – Limited access
- Easy access to data
- Easy recovery
- Flexible

1 (d) What is meant by a safe expression in relational calculus?

Sol: A SAFE EXPRESSION is one that is guaranteed to yield a finite number of tuples as its results. Otherwise, it is called UNSAFE.
{ t | not(EMPLOYEE) }
is UNSAFE!

1(e) How does SQL implement referential integrity constraint of relation data model?

Sol: SQL Foreign key or Referential Integrity :

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be a defined as a Primary Key in the table which it is referring. One or more columns can be defined as Foreign key.

Syntax to define a Foreign key at column level:

[CONSTRAINT constraint_name] REFERENCES Referenced_Table_name(column_name)

Syntax to define a Foreign key at table level:

[CONSTRAINT constraint_name] FOREIGN KEY(column_name) REFERENCES referenced_table_name(column_name);

1(f) Write the Armstrong's inference rules in their basic form.

Sol: Reflexive rule − If alpha is a set of attributes and beta is_subset_of alpha, then alpha holds beta.

Augmentation rule − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.

Transitivity rule − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

1(g) When are latches used?

Sol: A latch is a type of a lock that can be very quickly acquired and freed. Latches are typically used to prevent more than one process from executing the same piece of code at a given time.

1(h) how does a stored procedure created in pl/sql?

Sol: A procedure is created with the **CREATE OR REPLACE PROCEDURE** statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows −

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

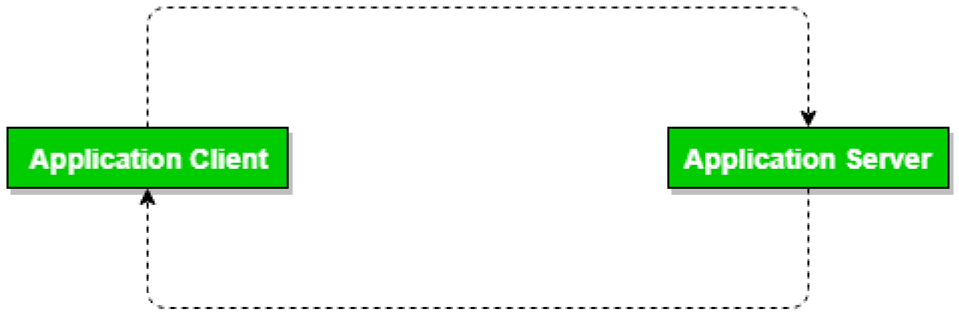
Where,

- *procedure-name* specifies the name of the procedure.
- [OR REPLACE] option allows the modification of an existing procedure.
- The optional parameter list contains name, mode and types of the parameters. **IN** represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- *procedure-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone procedure.

2(a) DBMS Architecture 2-Level, 3-Level

Two tier architecture:

Two tier architecture is similar to a basic **client-server** model. The application at the client end directly communicates with the database at the server side. API's like ODBC,JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side in order to communicate with the DBMS. An advantage of this type is that maintenance and understanding is easier, compatible with existing systems. However this model gives poor performance when there are a large number of users.



Three Tier architecture:

In this type, there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for exchange of partially processed data between server and client. This type of architecture is used in case of large web applications.

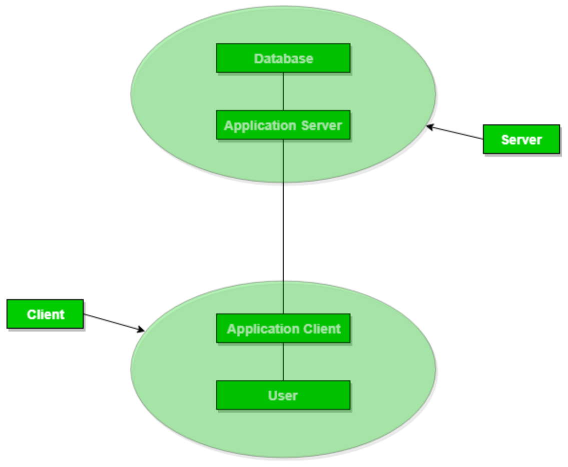
Advantages:

- **Enhanced scalability** due to distributed deployment of application servers. Now,individual connections need not be made between client and server.
- **Data Integrity** is maintained. Since there is a middle layer between client and server, data corruption can be avoided/removed.
- **Security** is improved. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

Disadvantages:

Increased complexity of implementation and communication. It becomes difficult for this sort of interaction to take place due to presence of middle layers.

Three-TierArchtecture



2(b) Describe two alternatives for specifying structural constraints on relationship types. What are the advantages and disadvantages?

Ans: Cardinality ratio:

The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in. For example, in the WORKS_FOR binary relationship type, DEPARTMENT: EMPLOYEE is of cardinality ratio l:N, meaning that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department. The possible cardinality ratios for binary relationship types are 1:1, l:N, N:l, and M:N.

Participation constraint:

The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the minimum cardinality constraint. There are two types of participation constraints-total and partial.

3(a) Discuss different types of user friendly interfaces and database utilities.

Sol: Following are the different types of interfaces provided by the DBMS:

Menu Based Interfaces for Web Clients:
These type of interfaces presents users with options in the form of menus and leads the users through formulation of request and the query.

Graphical User Interface:
The graphical user interfaces displays a schema to the user in diagram form and they use both forms and menus.

Interfaces for Parametric Users:
Parametric users have small set of operations they perform. Analysts and programmers design and implement a special interface for each class of naive users.

Form Based Interfaces:
These interfaces display a form to the user, user can fill out form to insert new data or fill out only certain entries.

Natural Language Interfaces:
These interfaces are designed to accept requests in written English, or other languages and attempt to understand them.

Database System Utilities

Loading : A loading utility is used to load existing data files-such as text files or sequential files-into the database.

Backup: A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape. The backup copy can be used to restore the database in case of catastrophic failure.

File Reorganization: This utility can be used to reorganize a database file into a different file organization to improve performance.

Performance Monitoring: Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether or not to reorganize files to improve performance.

3(b) Why er model is considered as high level conceptual model?

Sol: The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database.

In ER modeling, the structure for a database is portrayed as a diagram, called an entity-relationship diagram (or ER diagram), that resembles the graphical breakdown of a sentence into its grammatical parts. Entities are rendered as points, polygons, circles, or ovals. Relationships are portrayed as lines connecting the points, polygons, circles, or ovals. Any ER diagram has an equivalent relational table, and any relational table has an equivalent ER diagram. ER diagramming is an invaluable aid to engineers in the design, optimization, and debugging of database programs.

In a logical sense, entities are the equivalent of grammatical nouns, such as employees, departments, products, or networks. An entitycan be defined by means of its properties, called attributes. Relationships are the equivalent of verbs or associations, such as the act of purchasing, the act of repairing, being a member of a group, or being a supervisor of a department. A relationship can be defined according to the number of entities associated with it, known as the degree.

4(a) Discuss the DIVISION operator. How is it represented & what are the requirements of the numerator and denominator relations? Explain with an example.

Sol.: Division Operator (÷):

Division operator A÷B can be applied if and only if:

- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
- The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

Consider the relation STUDENT_SPORTS and ALL_SPORTS

STUDENT_SPORTS

ROLL_NO	**SPORTS**
1	Badminton
2	Cricket
2	Badminton
4	Badminton

ALL_SPORTS

SPORTS
Badminton
Cricket

To apply division operator as

STUDENT_SPORTS÷ ALL_SPORTS

- The operation is valid as attributes in ALL_SPORTS is a proper subset of attributes in STUDENT_SPORTS.
- The attributes in resulting relation will have attributes {ROLL_NO,SPORTS}-{SPORTS}=ROLL_NO
- The tuples in resulting relation will have those ROLL_NO which are associated with all B's tuple {Badminton, Cricket}. ROLL_NO 1 and 4 are associated to

Badminton only. ROLL_NO 2 is associated to all tuples of B. So the resulting relation will be:

ROLL_NO
2

4(b) What is view? How is it different from table? How views can be created, altered and destroyed?

Sol: Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

A table contains data, a view is just a SELECT statement which has been saved in the database (more or less, depending on your database).

The advantage of a view is that it can join data from several tables thus creating a new view of it. Say you have a database with salaries and you need to do some complex statistical queries on it.

Instead of sending the complex query to the database all the time, you can save the query as a view and then SELECT * FROM view.

Types of views :

1. *Read-only View* : Allows only SELECT operations.
2. *Updateable View* : Allows SELECT as well as INSERT , UPDATE and DELETE operations.

Creating a View :

The ORDER BY clause cannot be used while creating a view. The columns of the table are related to the view using a one-to-one relationship.

Syntax:

CREATE <OR REPLACE> VIEW <ViewName> AS SELECT <ColumnName1 >, <ColumnName2> FROM <TableName> WHERE <ColumnName> = < Expression List> <WITH READ ONLY> ;

This statements creates a view based on query specified in SELECT statement.
OR REPLACE option recreates the view if it is already existing maintaning the privileges granted to view viewname.
WITH READ ONLY option creates readonly view.

Example :

Creating a view stu based on student table and then update it.

```
SQL>create view stu as select enroll,name from student;

View Created.

SQL>select * from stu;

ENROLL             NAME
-----------    ----------
       4866        ABCD
       4546        BDSG
```

Updateable Views :

Views on which data manipulation can be done are called Updateable Views.
When an updateable view name is given in an Insert Update, or Delete SQL statement, modifications to data in the view will be immediately passed to the underlying table.
For a view to be updateable, it should meet the following criteria:

- Views defined from Single table
- If the user wants to INSERT records with the help of a view, then the PRIMARY KEY column(s) and all the NOT NULL columns must be included in the view .
- The user can UPDATE, DELETE records with the help of a view even if the PRIMARY KEY column and NOT NULL column(s) are excluded from the view definition .

```
SQL>update stu set name='xyz' where enroll=4866;

1 Row updated.

SQL>select * from stu;

ENROLL             NAME
-----------    ----------
       4866        xyz
       4546        BDSG
```

Destroying a View :

The drop command drops the specified view.

Syntax :

```
DROP VIEW Viewname;
```

Example:

```
SQL>drop view stu;

View dropped.
```

Advantages of View :

- Flexible enforcement of using Security.
- Simplification of complex Query.

5(a) Discuss different types of update operations on a relation with an example.

Sol: 1. INSERT a tuple
2. DELETE a tuple
3. MODIFY a tuple
Integrity constraints should not be violated by the update operations. Several update operations may have to be grouped together. Updates may propagate to cause other updates automatically. This may be necessary to maintain integrity constraints.
In case of integrity violation, several actions can be taken:
1. Cancel the operation that causes the violation (REJECT option)
2. Perform the operation but inform the user of the violation
3. Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
4. Execute a user-specified error-correction routine

5(b) What are the various types of inner join operations? Why is theta join required?

Sol: INNER JOINS: These joins are the one that has the tuples that satisfy some conditions and rest are discarded. Further they are classified as

- Theta join
- Equi join
- Natural join

Theta (θ) Join: Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol θ.

Notation

$R1 \bowtie_{\theta} R2$

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is R1 ∩ R2 = Φ.

Theta join can use all kinds of comparison operators.

Student		
SID	**Name**	**Std**
101	Alex	10
102	Maria	11

Subjects	
Class	**Subject**
10	Math
10	English
11	Music
11	Sports

Student_Detail −

```
STUDENT ⋈Student.Std = Subject.Class SUBJECT
```

Student_detail				
SID	**Name**	**Std**	**Class**	**Subject**
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Equijoin

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Natural Join (⋈)

Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.

Courses		
CID	**Course**	**Dept**
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD	
Dept	**Head**
CS	Alex
ME	Maya
EE	Mira

Courses ⋈ HoD			
Dept	**CID**	**Course**	**Head**
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Mira

6(a) Describe conceptually how an sql retrieval query will be executed by specifying the conceptual order of executing each of the six clauses

Sol: The logical query processing order starts with the FROM clause. Here is the logical query processing order of the six main query clauses:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Each phase operates on one or more tables as inputs and returns a virtual table as output. The output table of one phase is considered the input to the next phase. This is in accord with operations on relations that yield a relation. Note that if an ORDER BY is specified, the result isn't relational. This fact has implications that are discussed later in this Training Kit, in Chapter 3 and Chapter 4.

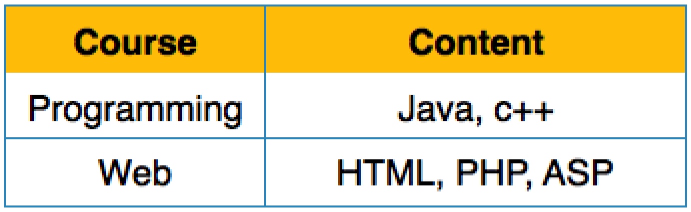
Consider the following query as an example.

```
SELECT country, YEAR(hiredate) AS yearhired, COUNT(*) AS numemployees

FROM HR.Employees

WHERE hiredate >= '20030101'

GROUP BY country, YEAR(hiredate)

HAVING COUNT(*) > 1

ORDER BY country , yearhired DESC;
```

This query is issued against the HR.Employees table. It filters only employees that were hired in or after the year 2003. It groups the remaining employees by country and the hire year. It keeps only groups with more than one employee. For each qualifying group, the query returns the hire year and count of employees, sorted by country and hire year, in descending order.

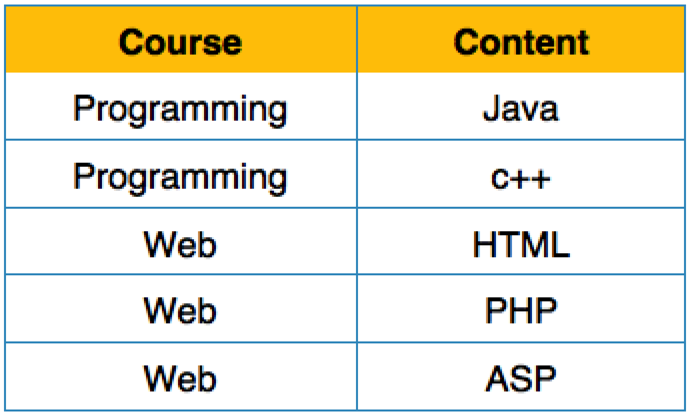
6(b) Illustrate how the process of creating first normal form relations may lead to multivalued dependencies

Sol: First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Each attribute must contain only a single value from its pre-defined domain.

This new table contains multivalued dependencies.

7(a) Discuss the design and implementation issues for active database.

Sol: An active database allows users to make the following changes to triggers (rules)

- Activate
- Deactivate
- Drop

An event can be considered in 3 ways

- Immediate consideration
- Deferred consideration

        - Detached consideration
- ✓ Immediate consideration
    - Part of the same transaction and can be one of the following depending on the situation
        - Before
        - After
        - Instead of
- ✓ Deferred consideration
    - Condition is evaluated at the end of the transaction
- ✓ Detached consideration
    - Condition is evaluated in a separate transaction

7(b) Define join dependency and 5NF. Why is 5NF also called project join normal form? Illustrate.

Sol: Fifth Normal Form (5NF)

Definition 1 :

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R.

Definition 2 :

A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated, when relations are reunited through a natural join.

What is a Join Dependency(JD) ??

Let R be a relation. Let A, B, …, Z be arbitrary subsets of R's attributes. R satisfies the JD

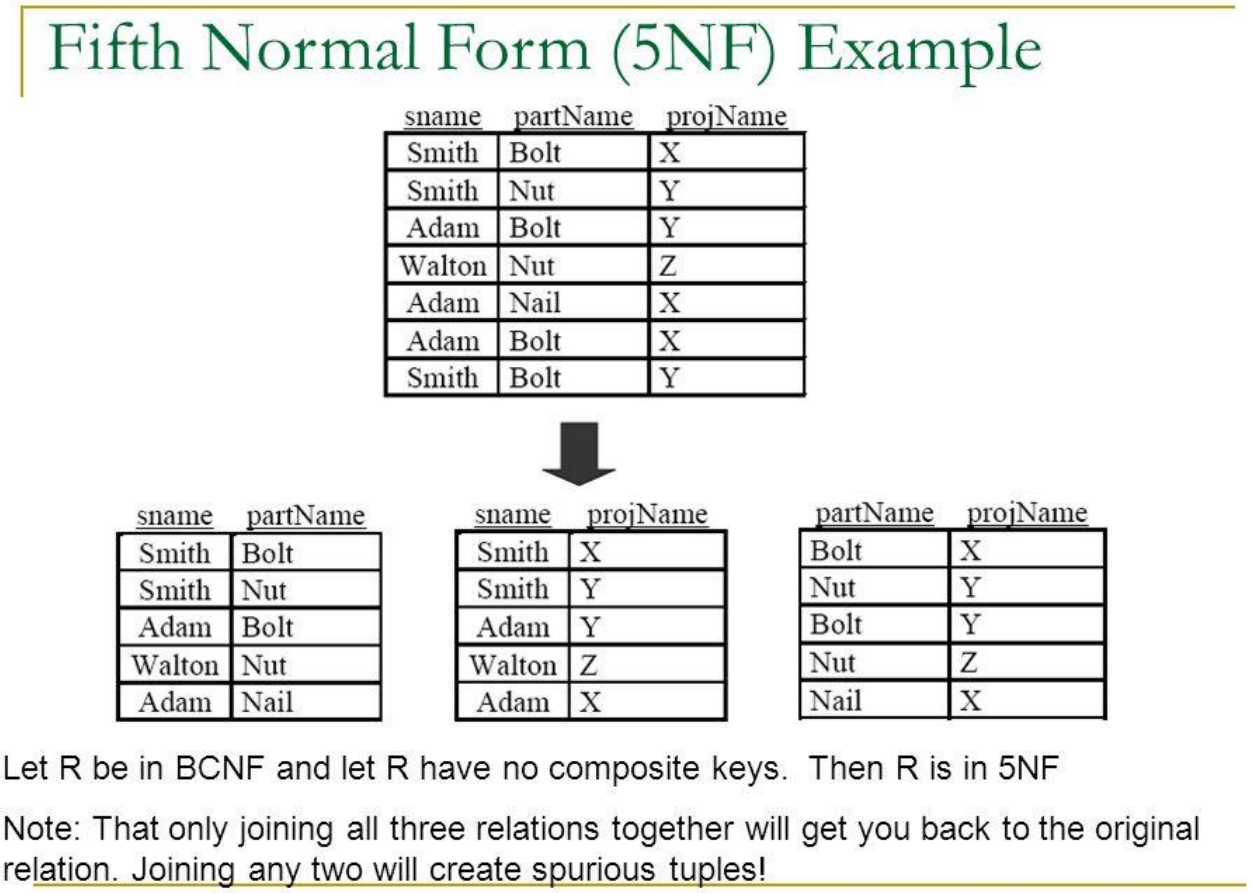
* (A, B, …, Z)

if and only if R is equal to the join of its projections on A, B, …, Z.

A join dependency JD(R1, R2, …, Rn) specified on relation schema R, is a trivial JD, if one of the relation schemas Ri in JD(R1, R2, ….,Rn) is equal to R.

It's also known as Project-join normal form(PJ/NF).

Fifth normal form is satisfied when all tables are broken into as many tables as possible in order to avoid redundancy. Once it is in fifth normal form it cannot be broken into smaller relations without changing the facts or the meaning.



Fifth Normal Form (5NF) Example

sname	partName	projName
Smith	Bolt	X
Smith	Nut	Y
Adam	Bolt	Y
Walton	Nut	Z
Adam	Nail	X
Adam	Bolt	X
Smith	Bolt	Y

sname	partName
Smith	Bolt
Smith	Nut
Adam	Bolt
Walton	Nut
Adam	Nail

sname	projName
Smith	X
Smith	Y
Adam	Y
Walton	Z
Adam	X

partName	projName
Bolt	X
Nut	Y
Bolt	Y
Nut	Z
Nail	X

Let R be in BCNF and let R have no composite keys. Then R is in 5NF

Note: That only joining all three relations together will get you back to the original relation. Joining any two will create spurious tuples!

8(a) Differentiate between view and conflict serializability. How both of these are tested for a schedule.

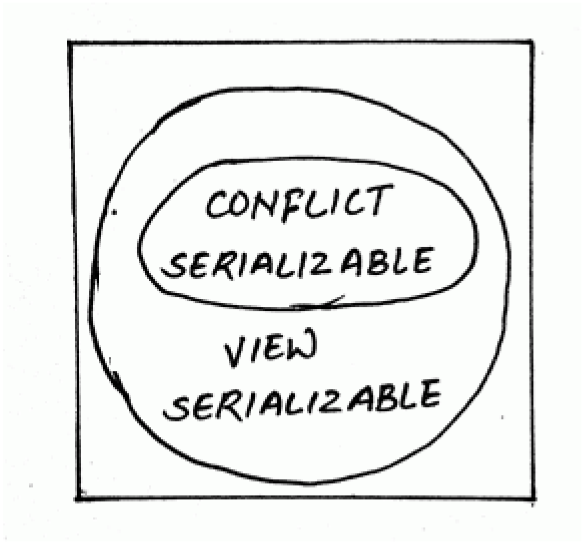
Sol:

Conflict Serializable Schedule: A Schedule is conflict serializable if it is conflict equivalent to any of serial schedule.

View serializable schedule: A Schedule is view serializable if it is view equivalent to any serial schedule.

Difference Between Conflict Schedule and View Schedule

1. Conflict serializability is easy to achieve but view serializability is difficult to achieve
2. Every conflict serializable is view serializable but the reverse is not true.
3. It is easy to test conflict serializability but expensive to test view serializability.
4. Most of the concurrency control schemes used in practice are based on conflict serializability.



Conflict serializability is tested using precedence graph.

Test for view serializability

1) Initial Read
If a transaction T1 reading data item A from initial database in S1 then in S2 also T1 should read A from initial database.

2)Updated Read
If Ti is reading A which is updated by Tj in S1 then in S2 also Ti should read A which is updated by Tj.
3)Final Write operation
If a transaction T1 updated A at last in S1, then in S2 also T1 should perform final write operations.

8(b) What is cursor? Discuss its different types. Name the commands used to control the cursor with their syntax.

Sol: A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

- Implicit cursors
- Explicit cursors

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT.

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is −

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor includes the following steps −

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data

- Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

```
CURSOR c_customers IS

  SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

```
CLOSE c_customers;
```

9(a) Discuss the optimistic concurrency control technique. Name its phases.

Sol: Optimistic concurrency control (OCC) is a concurrency control method applied to transactional systems such as relational database management systems and software transactional memory. OCC assumes that multiple transactions can frequently complete without interfering with each other. While running, transactions use data resources without acquiring locks on those resources. Before committing, each transaction verifies that no other transaction has modified the data it has read. If the check reveals conflicting modifications, the committing transaction rolls back and can be restarted.

OCC phases

More specifically, OCC transactions involve these phases:

- **Begin**: Record a timestamp marking the transaction's beginning.
- **Modify**: Read database values, and tentatively write changes.
- **Validate**: Check whether other transactions have modified data that this transaction has used (read or written). This includes transactions that completed after this transaction's start time, and optionally, transactions that are still active at validation time.
- **Commit/Rollback**: If there is no conflict, make all changes take effect. If there is a conflict, resolve it, typically by aborting the transaction, although other resolution schemes are possible. Care must be taken to avoid a TOCTTOU bug, particularly if this phase and the previous one are not performed as a single atomic operation.

9(b) Where do stored procedure and functions reside? Also discuss the steps to execute a procedure or function.

Sol: refer PL/SQL Procedures and functions