S.Y. B.Sc. (IT) : Sem. III

Database Management Systems

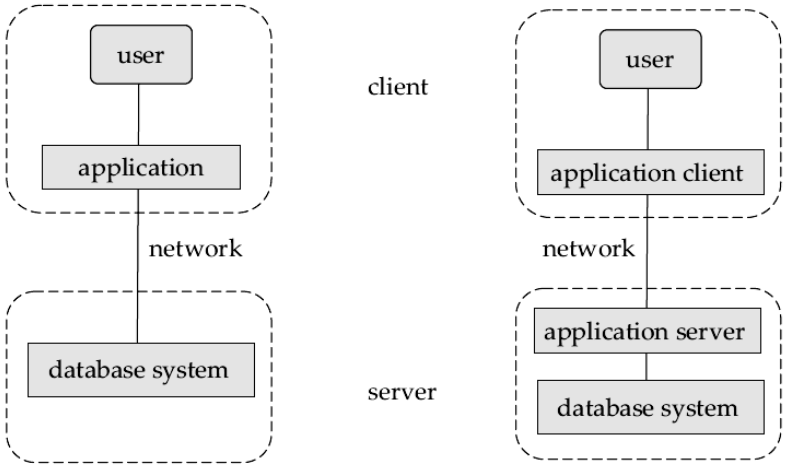
Time : 2½ Hrs.] Prelim Question Paper Solution **[Marks : 75**

Q.1 Attempt the following (any THREE) [15]

Q.1(a)Explain database architecture. [5]

Ans.:

- The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.
- Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database users work, and server machines, on which the database system runs. Database applications are usually partitioned into two or three parts, as in Figure.



(a) Two-tier architecture (b) Three-tier architecture

Fig.: Two-tier and three-tier architectures.

- In a two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

- In contrast, in a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the Worldwide Web.

Q.1(b)Explain hierarchical and Network model. [5]

Ans.:

1. The hierarchical model was developed in the 1960s to manage large amounts of data for complex manufacturing projects, such as the Apollo rocket that landed on the moon in 1969. The model's basic logical structure is represented by an upside-down tree.

2. The hierarchical structure contains levels, or segments. A segment is the equivalent of a file system's record type. Within the hierarchy, a higher layer is perceived as the parent of the segment directly beneath it, which is called the child. The hierarchical

model depicts a set of one-to-many (1:M) relationships between a parent and its children segments. (Each parent can have many children, but each child has only one parent.)

3. The network model was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard. In the network model, the user perceives the network database as a collection of records in 1:M relationships. However, unlike the hierarchical model, the network model allows a record to have more than one parent.

4. While the network database model is generally not used today, the definitions of standard database concepts that emerged with the network model are still used by modern data models:

- The schema is the conceptual organization of the entire database as viewed by the database administrator.
- The subschema defines the portion of the database "seen" by the application programs that actually produce the desired information from the data within the database.
- A data manipulation language (DML) defines the environment in which data can be managed and is used to work with the data in the database.
- A schema data definition language (DDL) enables the database administrator to define the schema components.

5. As information needs grew and more sophisticated databases and applications were required, the network model became too cumbersome. The lack of ad hoc query capability put heavy pressure on programmers to generate the code required to produce even the simplest reports. Although the existing databases provided limited data independence, any structural change in the database could still produce havoc in all application programs that drew data from the database. Because of the disadvantages of the hierarchical and network models, they were largely replaced by the relational data model in the 1980s.

Q.1(c) Explain the concept of entity relationship model. [5]

Ans.: 1. The entity-relationship (E-R) data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.

The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the E-R model. The E-R data model employs three basic concepts: entity sets, relationship sets, and attributes, which we study first. The E-R model also has an associated diagrammatic representation, the E-R diagram.

2. Entity Sets :

(a) An entity is a "thing" or "object" in the real world that is distinguishable from all other objects. For example, each person in a university is an entity. An entity has a set of properties, and the values for some set of properties may uniquely identify an entity. For instance, a person may have a person id property whose value uniquely identifies that person.

(b) An entity set is a set of entities of the same type that share the same properties, or attributes. The set of all people who are instructors at a given university, for example, can be defined as the entity set instructor. Similarly, the entity set student might represent the set of all students in the university.

(c) An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set. The designation of an attribute for an entity set expresses that the database stores similar information concerning each entity in the entity set; however, each entity may have its own value for each attribute. Possible attributes of the instructor entity set are ID, name, dept name, and salary.

(d) Each entity has a value for each of its attributes. For instance, a particular instructor entity may have the value 12121 for ID, the valueWu for name, the value Finance for dept name, and the value 90000 for salary.

3. Relationship Sets :

(a) A relationship is an association among several entities. For example, we can define a relationship advisor that associates instructor Katz with student Shankar. This relationship specifies that Katz is an advisor to student Shankar. A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on $n \geq 2$ (possibly nondistinct) entity sets. If E1, E2, . . . , En are entity sets, then a relationship set R is a subset of {(e1, e2, . . . , en) | e1 ∈ E1, e2 ∈ E2, . . . , en ∈ En} where (e1, e2, . . . , en) is a relationship.

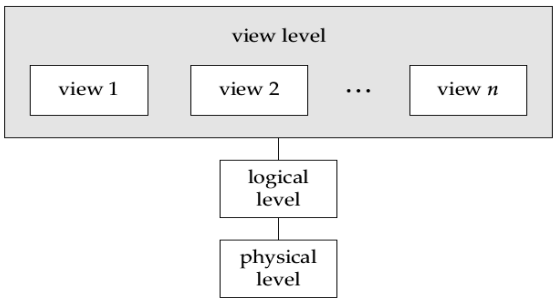
(b) The association between entity sets is referred to as participation; that is, the entity sets E1, E2, . . . , En participate in relationship set R. A relationship instance in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled. As an illustration, the individual instructor entity Katz, who has instructor ID 45565, and the student entity Shankar, who has student ID 12345, participate in a relationship instance of advisor. This relationship instance represents that in the university, the instructor Katz is advising student Shankar.

Q.1(d) List and Explain different levels of abstraction. [5]

Ans.: Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

We have three levels of abstraction :

- **Physical level :** This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.
- **Logical level :** This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.
- **View level:** Highest level of data abstraction. This level describes the user interaction with database system.



At the logical level these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Q.1(e)What are the basic building blocks of data model? Explain with example. [5]

Ans.: **1.** The basic building blocks of all data models are entities, attributes, relationships, and constraints. An entity is a person, place, thing, or event about which data will be collected and stored. An entity represents a particular type of object in the real world, which means an entity is "distinguishable"—that is, each entity occurrence is unique and distinct. For example, a CUSTOMER entity would have many distinguishable customer occurrences, such as John Smith, Pedro Dinamita, and Tom Strickland. Entities may be

physical objects, such as customers or products, but entities may also be abstractions, such as flight routes or musical concerts.

2. An **attribute** is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone number, customer address, and customer credit limit. Attributes are the equivalent of fields in file systems.
3. A **relationship** describes an association among entities. For example, a relationship exists between customers and agents that can be described as follows: an agent can serve many customers, and each customer may be served by one agent. Data models use three types of relationships: one-to-many, many-to-many, and one-to-one. Database designers usually use the shorthand notations 1:M or 1..*, M:N or *..*, and 1:1 or 1..1, respectively. (Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.) The following examples illustrate the distinctions among the three relationships.
   - **One-to-many (1:M or 1..*) relationship :** A painter creates many different paintings, but each is painted by only one painter. Thus, the painter (the "one") is related to the paintings (the "many"). Therefore, database designers label the relationship "PAINTER paints PAINTING" as 1:M. Note that entity names are often capitalized as a convention, so they are easily identified. Similarly, a customer (the "one") may generate many invoices, but each invoice (the "many") is generated by only a single customer. The "CUSTOMER generates INVOICE" relationship would also be labeled 1:M.
   - **Many-to-many (M:N or *..*) relationship :** An employee may learn many job skills, and each job skill may be learned by many employees. Database designers label the relationship "EMPLOYEE learns SKILL" as M:N. Similarly, a student can take many classes and each class can be taken by many students, thus yielding the M:N label for the relationship expressed by "STUDENT takes CLASS."
   - **One-to-one (1:1 or 1..1) relationship :** A retail company's management structure may require that each of its stores be managed by a single employee. In turn, each store manager, who is an employee, manages only a single store. Therefore, the relationship EMPLOYEE manages STORE" is labeled 1:1.

     The preceding discussion identified each relationship in both directions; that is, relationships are bidirectional:
     1. One CUSTOMER can generate many INVOICEs.
     2. Each of the many INVOICEs is generated by only one CUSTOMER.
4. A **constraint** is a restriction placed on the data. Constraints are important because they help to ensure data integrity. Constraints are normally expressed in the form of rules. For example:
   - An employee's salary must have values that are between 6,000 and 350,000.
   - A student's GPA must be between 0.00 and 4.00.
   - Each class must have one and only one teacher.
5. How do you properly identify entities, attributes, relationships, and constraints? The first step is to clearly identify the business rules for the problem domain you are modeling.

Q.1(f) State and explain any 5 Codd's rules for relational database. [5]

Ans.:

1. **Information :** All information in a relational database must be logically represented as column values in rows within tables.
2. **Guaranteed access :** Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3. **Systematic treatment of nulls :** Nulls must be represented and treated in a systematic way, independent of data type.

4. **Dynamic online catalog based on the relational model :** The metadata must be stored and managed as ordinary data—that is, in tables within the database. Such data must be available to authorized users using the standard database relational language.
5. **Comprehensive data sublanguage :** The relational database may support many languages; however, it must support one well-defined, declarative language as well as data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6. **View updating :** Any view that is theoretically updatable must be updatable through the system.
7. **High-level insert, update, and delete :** The database must support set-level inserts, updates, and deletes.
8. **Physical data independence :** Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9. **Logical data independence :** Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of columns or inserting columns).
10. **Integrity independence :** All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11. **Distribution independence :** The end users and application programs are unaware of and unaffected by the data location (distributed vs. local databases).
12. **Nonsubversion :** If the system supports low-level access to the data, users must not be allowed to bypass the integrity rules of the database.
13. **Rule zero :** All preceding rules are based on the notion that to be considered relational, a database must use its relational facilities exclusively for management.

Q.2 Attempt the following (any THREE) [15]

Q.2(a) Define Relational Algebra .Explain Selection and projection operation [5]

Ans.:

1. The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are select, project, union, set difference, Cartesian product, and rename. In addition to the fundamental operations, there are several other operations—namely, set intersection, natural join, and assignment.
2. **The Select Operation :** The select operation selects tuples that satisfy a given predicate. We use the lowercase Greek letter sigma to denote selection.
   Thus, to select those tuples of the instructor relation where the instructor is in the "Physics" department, we write:
   $\sigma_{dept\_name = \text{"Physiscs"}}$ (instructor)
3. **The Project Operation :** Suppose we want to list all instructors' ID, name, and salary, but do not care about the dept name. The project operation allows us to produce this relation. The project operation is a unary operation that returns its argument relation, with certain attributes left out. Since a relation is a set, any duplicate rows are eliminated. Projection is denoted by the uppercase Greek letter pi.
   $\Pi_{ID, name, salary}$ (instructor)

Q.2(b) Explain the concept of determination and functional dependencies. [5]

Ans.: Two types of functional dependencies that are of special interest in normalization are partial dependencies and transitive dependencies:

(a) A partial dependency exists when there is a functional dependence in which the determinant is only part of the primary key (remember the assumption that there is only one candidate key). For example, if (A, B) → (C, D), B → C, and (A, B) is the primary

key, then the functional dependence B → C is a partial dependency because only part of the primary key (B) is needed to determine the value of C. Partial dependencies tend to be straightforward and easy to identify.

(b) A transitive dependency exists when there are functional dependencies such that X → Y, Y → Z, and X is the primary key. In that case, the dependency X → Z is a transitive dependency because X determines the value of Z via Y. Unlike partial dependencies, transitive dependencies are more difficult to identify among a set of data. Fortunately, there is an effective way to identify transitive dependencies: they occur only when a functional dependence exists among nonprime attributes. In the previous example, the actual transitive dependency is X → Z. However, the dependency Y → Z signals that a transitive dependency exists. Hence, throughout the discussion of the normalization process, the existence of a functional dependence among nonprime attributes will be considered a sign of a transitive dependency. To address the problems related to transitive dependencies, changes to the table structure are made based on the functional dependence that signals the transitive ependency's existence. Therefore, to simplify the description of normalization, from this point forward the signaling dependency will be called the transitive dependency.

(c) Functional dependence : The attribute B is fully functionally dependent on the attribute A if each value of A determines one and only one value of B.
Example: PROJ_NUM → PROJ_NAME
(read as PROJ_NUM functionally determines PROJ_NAME)
In this case, the attribute PROJ_NUM is known as the determinant attribute, and the attribute PROJ_NAME is known as the dependent attribute.

(d) Functional dependence (generalized definition)
Attribute A determines attribute B (that is, B is functionally dependent on A) if all of the rows in the table that agree in value for attribute A also agree in value for attribute B.

(e) Fully functional dependence (composite key)
If attribute B is functionally dependent on a composite key A but not on any subset of that composite key, the attribute B is fully functionally dependent on A.

Q.2(c) What is normalization? What is its objective? Give a distinguishing characteristic of 1NF, 2NF, 3NF and BCNF. [5]

Ans.: Normalization is a process for evaluating & connecting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies.

The objective of normalization is to ensure that each table conforms to the concept of well-formed relational – in other words, tables that have the following characteristics :

- Each table represents a single subject.
- No data item will be unnecessarily studies more than one table.
- All non-prime attributes is a table are dependent on the PK
- Each table is void of in section.

Normal forms

Normal form	Characteristics
• First Normal form (1NF)	Table format, no repeating groups and PK identified.
• Second Normal form (2NF)	1 NF and no partial dependence
• Third Normal form (3NF)	2 NF and no transitive dependence
• Fourth Normal form (4NF)	3 NF and no independents multivalued dependence
• Boyce-Codd Normal form (BCNF)	Every determinant is a candidate key

Q.2(d) (i) Using the EMPLOYEE table structure shown in table 1, write the relational schema, draw its dependency diagram and identify all dependencies (including all partial and transitive dependencies). You can assume that the table does not contain repeating groups and that any invoice number may reference more than one product. (*Hint*: This table uses a composite primary key.) **[5]**

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.8
		101	John G. News	Database Designer	$105.00	19.4
		105	Alice K. Johnson *	Database Designer	$105.00	35.7
		106	William Smithfield	Programmer	$35.75	12.6
		102	David H. Senior	Systems Analyst	$96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	$48.10	24.6
		118	James J. Frommer	General Support	$18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	$96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	$45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	$96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	$48.10	23.6
		111	Geoff B. Wabash	Clerical Support	$26.87	22.0
		106	William Smithfield	Programmer	$35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	$35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	$96.75	45.8
		101	John G. News *	Database Designer	$105.00	56.3
		114	Annelise Jones	Applications Designer	$48.10	33.1
		108	Ralph B. Washington	Systems Analyst	$96.75	23.6
		118	James J. Frommer	General Support	$18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	$45.95	41.4

A TABLE WHOSE STRUCTURE MATCHES THE REPORT FORMAT

(ii) Using the initial dependency diagram drawn in question (i), remove all partial dependencies, draw the new dependency diagrams, and identify the normal forms for each table structure you created.

(iii) Using the table structures you created in question (ii), remove all transitive dependencies and draw the new dependency diagrams. Also identify the normal forms for each table structure you created.

Ans.:

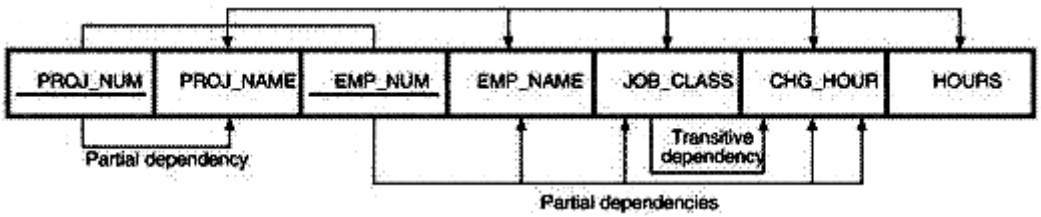


Fig.1 : A dependency diagram : First Normal Form (1NF)

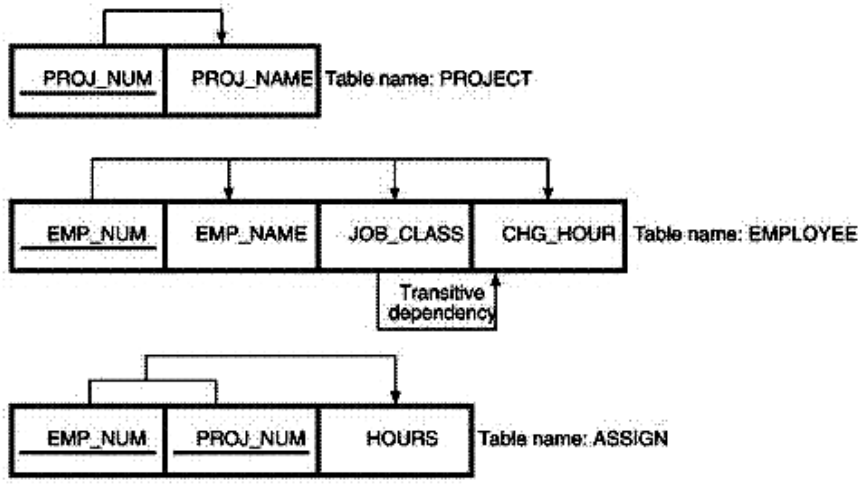


Fig.2: Second Normal Form (2NF) Conversion results

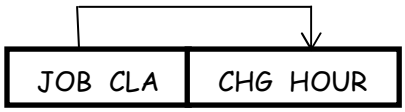


Table name:JOB

THIRD NORMAL FORM RESULTS

PROJECT (PROJ_NUM, PROJ_NAME)

ASSIGN (PROJ_NUM, EMP_NUM, HOURS)

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

JOB (JOB_CLASS, CHG_HOUR)

Q.2(e) Define Super key, Candidate key, Primary key and Foreign key with examples. [5]

Ans.: Table 1 : Student

Stud_No.	Stud_Name	Stud_Phone	Stud_State	Stud_Country	Stud_Age
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291281	Rajasthan	India	18
4	SURESH		Punjab	India	21

Table 2 : Student_Course

Stud_No.	Stud_Name	Stud_Phone
1	C1	DBMS
2	C2	Computer Networks
3	C3	Computer Networks

- **Candidate Key :** The minimal set of attribute which can uniquely identify a tuple is known as candidate key. For example, STUD_NO in STUDENT relation.
  The value of Candidate Key is unique and non-null for every tuple.
  There can be more than one candidate key in a relation. For example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT.
  The candidate key can be simple (having only one attribute) or composite as well. For example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

- **Super Key :** The set of attributes which can uniquely identify a tuple is known as Super Key. For example, STUD_NO, {STUD_NO, STUD_NAME) etc.
  Adding zero or more attributes to candidate key generates super key.
  A candidate key is a super key but vice versa is not true.

- **Primary Key :** There can be more than one candidate key in a relation out of which one can be chosen as primary key. For example, STUD_NO as well as STUD_PHONE both are candidate keys for relation STUDENT but STUD_NO can be chosen as primary key (only one out of many candidate keys).

- **Foreign Key :** If an attribute can only take the values which are present as values of some other attribute, it will be foreign key to the attribute to which it refers. The relation which is being referenced is called referenced relation and corresponding attribute is called referenced attribute and the relation which refers to referenced relation is called referencing relation and corresponding attribute is called referencing attribute. Referenced attribute of referencing attribute should be primary key. For example, STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

Q.2(f) Given the relational schemas and relational algebra write the domain calculus for the following: [5]

(i) For R(A,B) and S(C,D) write domain calculus for r × s

(ii) For R(A,B,C,D) and S(B,D,E) write domain calculus for r ⋈ s

(iii) For R(A,B) write domain calculus for ΠA(σB=17 (r))

(iv) For R(A,B,C) and S(A,B,C) write domain calculus for r – s

Ans.: (i) For R(A,B) and S(C,D) write domain calculus for $r \times s$
Domain Calculus: $\{\langle a, b, c, d\rangle \mid \langle a, b\rangle \in r \wedge \langle c, d\rangle \in s\}$

(ii) For R(A,B,C,D) and S(B,D,E) write domain calculus for $r \bowtie s$
$\{\langle a, b, c, d, e\rangle \mid \langle a, b, c, d\rangle \in r \wedge \langle b, d, e\rangle \in s\}$

(iii) For R(A,B) write domain calculus for ΠA(σB=17 (r))
Domain Calculus: $\{\langle a, b\rangle \mid \langle a, b\rangle \in r \wedge b = 17\}$

(iv) For R(A,B,C) and S(A,B,C) write domain calculus for $r - s$
Domain Calculus: $\{\langle a, b, c\rangle \mid \langle a, b, c\rangle \in r \wedge \langle a, b, c\rangle \notin s\}$

Q.3 Attempt the following (any THREE) [15]

Q.3(a) State the rules for performing DML operations on a view. [5]

Ans.: You can perform DML operations on data through a view if those operations. Follow certain rules.

You can remove a new from a view unless it constraints any of the following :

- Group functions.
- A group by clause.
- A distinct keyword
- The pseudo Row number keyword

You can add data through a view unless it contains only of the items listed in the slide or there are NOT NULL columns without default values in the base table they are not selected by the view. All required value must be present in the view. We add values directly into the underlying table through the view, You cannot add data through a view if the view Includes,

- A group functions, A group by clause. The distinct keyword, the pseudo column ROWNUM Keyword, Columns defined by expression, NOT null columns defined by expression, Not null columns in the base table that are not selected by the view.
  You cannot modify data in a view if it contains :–
- Group Function
- A group by clause.
- The distinct keyword
- The pseudo column row num keyword
- Columns defined by expressions. You can ensure that no DML operation.

Occur on your view by creating it with the WILL READ ONLY option.

Q.3(b) Write short note on DML,DDL and DCL. [5]

Ans.: **1. Data Definition Language (DDL) :**

- The DDL is also used to specify the structure of each table, set of associated values with each attribute, integrity constraints, security and authorization information for all the tables and physical storage structure of all the tables on the disk.
- Let's take SQL for instance to categorize the statements that comes under DDL.
  - To create the database instance — CREATE
  - To alter the structure of database — ALTER
  - To drop database instances — DROP
  - To rename database instances — RENAME

2. **Data Manipulation Language (DML) :**
   - The Data Manipulation Language (DML) is used for accessing and manipulating data in a database. DML provides a set of functionalities to support the basic data manipulation operations on the data stored in the database.
   - It allows users to access, insert, update, and delete data from the database.
     - To access or read records from table — SELECT
     - To insert record into the table — INSERT
     - Update the records in table — UPDATE
     - Delete the records from the table — DELETE

3. **Data Control Language (DCL) :**
   - Data Control Language (DCL) is used to control the user access to the database related elements like tables, views, functions, procedures and packages.
   - It provides different levels of access to the objects in the database.
     - To grant access to user — GRANT
     - To revoke access from user — REVOKE
   - Grant : GRANT is used to provide the privileges to the users on the database objects.
   - Revoke: REVOKE removes the privileges given on the database objects.

4. **Transaction Control Language (TCL) :**
   - TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.
     - BEGIN Transaction – opens a transaction
     - COMMIT Transaction – commits (Save permanently) transactions
     - ROLLBACK Transaction – ROLLBACK (Cancels, undo) transactions in case of any

Q.3(c) What are constraints? What are the different types of constraints? Explain. [5]

Ans.: Constraints enforce rules at the table level constraints prevent deletion of a table if there are dependencies.

The following are the five types of constraints :
1) Not NULL 2) Unique 3) Primary key 4) Foreign key 5) check

1) **Not Null :** The not null constraints ensures that the column constraints no null values columns without the NOT NULL constraint can contain null value by default. The NOT NULL constraint can be specified only at the column level, not as the table level.

2) **Unique :** Unique key, constraint request that every value in a column or set of columns be unique that is, no two rows of a table can have duplicate values in a specified column or set of columns.

3) **The primary key :** Primary key constraint creates a primary key for the table. Only one primary key can be created for each table. The primary key, constraint is a column or set of columns that uniquely identifies each row in a table. This constraints enforces uniqueness of the column or column combination and ensures that is part of primary key can constraint a null value. Primary key can be defined at the column level or table level. A composite primary key is created by using the table level distribution.

4) **The foreign key, constraint :** The foreign key or referential integrity constraint, designates a column or combination of columns as a foreign key and establishes a relationship between a primary key or unique key in the same table or a different table.
   - Foreign key is used to define the column in the child table at the table constraint level.
   - References identifies the table and column in the parent table.

5) **The check constraint :** The check constraint defines a condition that each row must satisfy. The condition can use the same constraints as query conditions, with the following exceptions.
   - References to the current / next val, level and rownum pseudo columns.
   - Calls at sysdate, U & D, User, userrear functions
   - Queries that refer to other values in other rows.

   A single column can have multiple check constraints which refer to the column in its definition. There is no limit to the number of check constraint which you can define on a column.
   Check constraints can be defined at the column level or table level.

Q.3(d) Explain GROUP BY AND ORDER BY clauses with examples. [5]

Ans.: The group by clause :

You can use the group by clause to divide the rows in table into groups. You can then use the group functions to return summery information for each group.

In the system : Group by–expression the basis for grouping rows.

- If you include a group function in a select clause, you cannot select individual column appears in the group by clause. You receive an error message if you fail to include the column list in the group by clause.
- Using a where clause, you can exclude rows before dividing them into groups.
- You must include the columns in the group by clause.
- By default rows are sorted by ascending order of the columns included in the group by list. You can override this, by using the order by clause.
  When using the group by clause, make sure that all column in the select list that are not group functions are included in the group by clause Restricting group results.
  In the same way, that use the clause to restrict the rows that you selected, you use the having clause to restrict group.
  Select column, group–function & from table.
  [Where condition] [Group by group_by_expr_]
  [having group_conditions]
  [order by column]
- The order by clause.
  The order of rows returned in a query result is undefined. The order by clause can be used to sort the rows of the order by clause, it must be the last clause of the SQL statement. You can specify on expression, or on alias or column position as the sort condition.

Adding New Column in a Table :

```
ALTER TABLE table_name
ADD column_name datatype;
```

For example, Adding column grade in the table student.

```
ALTER TABLE student
ADD Grade varchar(2);
```

Dropping Column from Table :

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

For example, Deleting column grade from the table student. ALTER TABLE student

```
ALTER TABLE student
DROP Column Grade;
```

Modifying Column of a Table :
ALTER TABLE table_name
MODIFY COLUMN column_name data_type;

For example, Changing the data type and size of column roll_number of student table.
ALTER TABLE student
modify Column roll_no varchar(4);

Deleting all the records from Table :
Syntax :
TRUNCATE TABLE

For example, Deleting all the records from newstudent1
TRUNCATE TABLE newstudent1;

Deleting Table :
DROP TABLE query is used to delete table permanently from the database.
Syntax :
drop table table_name;

For Example: Deleting the newstudent1 table from the database.
Drop table newstudent1;

Q.3(e) What are joins? Explain different types of joins. [5]

Ans.: The joins clause is used to combine records from two or more tables in a database. A Joins is a means for combining fields from two tables by using values common to both. Several operates can be used to join tables such as =, <, >, < >, < =, >=, !=. Between, like and NOT. However the most common operator is the equal to symbol.

There are different types of joins such as

1) Inner Join – returns rows when there is a match is both tables.
Syntax :
Select table1.col1, table2.col2….
From table1
Inner Join table2
An table1.common field = table2.common–field.

2) Left join – returns all rows from the left table, even if there are no matches in the right table.
Syntax :
Select table1 col1, table2 col2, …..
From table1
Left join table2
An table1.common_field = table2.common_field.

3) Right join – returns all rows from the right table even if there are no matches in the left table.
Syntax :
Select table col1, table2, col2…
Form table1
Right join table2
On table1.common_field = table2.common_field.

4) Full join – returns rows when there is a match is one of the tables.
Syntax :
```
Select table1.col.1, table2.col2….
From table1
Full join table2
On table1_common_field=table2.common_field=table2.common_field.
```

5) Self join – is used to join a table to itself as if the tables were two tables, temporarily renaming at least one table in the SDL statement.
Syntax :
```
Select a.col_name, b.col_name…
From table1 a, table1 b
Where a.common _field = b. common_field
```

6) Cartesian join – returns the Cartesian product of the sets of records from the two or more joined tables.
Syntax :
```
Select table1.col1, table2, col2….
From table1, table2 [,table3]
```

Q.3(f) Difference between tables and Views. [5]

Ans.:

Sr. No.	Tables	Views
1	Table is a primary storage for storing data in relational database management system	A view is a virtual table whose contents are defined by a query
2	When data is huge, there may complexity in tables	A view hides the complexity of the database tables from end users
3	Space required by tables in memory is more	Space required by views in memory very less
4	Tables can be created independently	Views are always created on tables
5	Table is designed as limited number of columns and unlimited number of rows	View is designed as virtual table that is extracted from a database
6	View can integrate several tables in to one virtual table	Multiple tables are required to store linked data and records

Q.4 Attempt the following (any THREE) [15]

Q.4(a) State and Explain the ACID properties of transactions. [5]

Ans.: 1) **Atomicity:** Either all operations of the transaction are reflected properly in the database or none are.

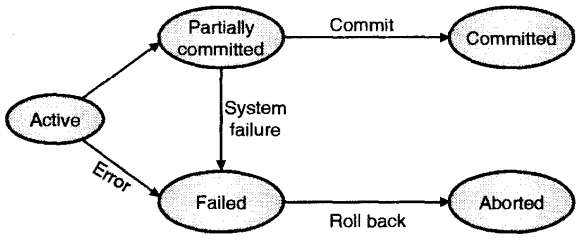
2) **Consistency:** Execution of a transaction is isolation (that is, with no other to transaction executing concurrently) preserves – the consistency of the condition.

3) **Isolation:** Even – though multiple transaction may execute concurrently, the system guarantees that, for every pair of transaction T_i and T_j, it appears to T_i that either T_j finished execution before T_i started or T_j started execution after it finished. Thus, each transaction is unawares of other transaction executing concurrently in the system,

4) **Durability:** After a transaction completes successfully, the changes it has made to the database persist, even it there are any system failure.

Q.4(b) List and explain different states through which transaction goes through during its execution. [5]

Ans.: 1. There are five states of transaction. During execution, transaction will be in one of the following state.



2. **Active** : This is the first state of transaction. In this state the transaction is being executed. This state is entry point to every transaction. Transactions which are in active state indicate that their execution has been started.
3. **Partially Committed** : When a transaction completes all operations, it will enter into partially committed state. Even when the last operation has been performed; data is still not saved to the database.
4. **Failed** : If execution of transaction cannot proceed due to failure of the system or failure in database, then the transaction is said to be in failed state.
5. **Abort** : In case of a failed transaction, the modification done in database during transaction processing must be rolled back to ensure the atomicity and consistency of database. The transaction enters into 'Abort' state. The transaction is thus in abort state after roll back operation is performed. It is the end of transaction when any fault occurs.
6. **Committed** : If without any error a transaction gets completed successfully it will come into committed state which will allow to made changes permanent into database. This is the last step of a transaction, if it executes without any failure.
   Every transaction goes through at least three states among of five. Either transaction may goes through active -partially committed - committed or through active - Failed - aborted, or through active - partially committed - failed - abort.

Q.4(c) Explain lock based protocols. **[5]**

Ans.: **1. Two-Phase Locking (2PL)**

(a) In this scheme, each transaction makes lock and unlock request in 2 phases:
- A Growing Phase (or An Expanding Phase or First Phase) : In this phase, new locks on the desired data item can be acquired but none can be released.
- A Shrinking Phase (or Second Phase) : In this phase, existing locks can be released but no new locks can be acquired.

(b) In the beginning, the transaction is in growing phase in which it acquire lock as per requirement. After completing the work, the transaction releases the locks and enters into shrinking phase. The transaction cannot request for new locks after_releasing the locks.

(c) The point in the schedule where the transaction acquires its final lock is called as lock point. This point is the end of growing phase. The transactions can be ordered as per their lock points. This sequence of transactions is the serailizability ordering for the transactions. This protocol assures serializability.

(d) In Serializability the main issue is regarding write operation. The parallel write operations may create inconsistency in the database. The parallel read operations do not create any problem.

(e) In serial schedule, as the transactions are executed one after other, there is no risk of parallel write operations which may lead to inconsistency.

(f) The database transactions can be controlled in 2-Phase locking mechanism by applying the exclusive lock. The exclusive (write) lock is released in shrinking phase. Unless the exclusive lock of a data item is released by the transaction, other transaction cannot acquire exclusive lock on that data item. This helps to maintain the serailizability. It is not sure that Two-phase locking is free from deadlocks.

(g) In two-phase locking, there is possibility of cascading roll-back. This can be avoided by using strict two-phase locking in which a transaction must hold all its exclusive locks till it commits/aborts.

(h) Rigorous two-phase locking is more stricter. In it all the locks are held till commit/abort. Here transactions can be serialized in the order in which they commit.

2. Strict Two-Phase Locking :

(a) In Strict two phase locking protocol the first step of acquiring all the locks is same as first phase of two phase locking protocol. But the difference in Strict two phase locking and two phase locking is that Strict two phase locking protocol does not release the lock after using it.

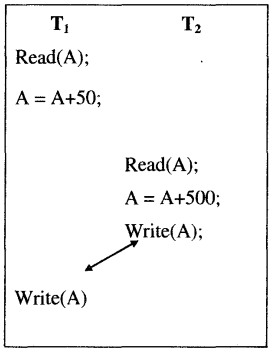
(b) It holds all the locks till it does not reach to commit point. When it reached to the commit point it releases all locks at a time. Cascading rollback can be prevented by using the strict phase locking protocol. This protocol is used to make Cascadless schedules.

Q.4(d) List the problems in concurrent execution of Transactions. [5]

Ans.: The problem occurs when two transactions are dealing with same data item and if one of the transactions performs write operation. If any transaction contains following situations then the conflicts are occurred:

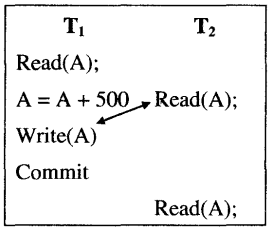
(a) Write-Write conflicts :

- Here both the transactions T1 and T2 read initial values of data item (A) [assume A=1000].
- Initially T2 adds 500 to A and write new value 1500 of A into database.
- Now T1 update the initial value of A(1000) to 1050 and write it in the database.
- In between these transactions, the value updated by transaction T2 is lost.
- Thus write-write conflicts affects lost update on database.



(b) Read-Write conflicts:

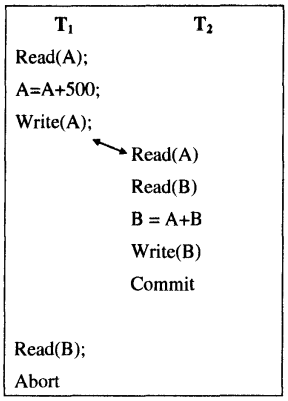
- Transaction T1 and T2 reads initial value of A (1000)
- T1 updates value of A to 1500 by adding 500 to it and writes it into database and commit.
- Now T2 again read the value of A which is now 1500.
- Here transaction T2 reads two different values of same data element A.
- This is read-write conflict.



(c) Write-Read conflicts :

- Transaction T1 reads value of A(1000) from database. Add 500 to it and write new value 1500 in the database.

- Transaction T2 reads updated value of A(1500) from database, add it into B and writes value of B in the database.
- But transaction T1 get aborted due to failure and all changes are rolled back.
- Here value read by T2 is dirty read i.e. value updated by uncommitted transaction.



Q.4(e) Define Deadlock. Explain deadlock detection. **[5]**

Ans.: 1. A system is said to be in a state of Deadlock, if every transaction in the schedule is waiting for another transaction in the schedule to release the lock of some data item. Consider transactions from T0 to Tn. In this case transaction T1 is waiting for T2, T2 is waiting for T3 and so on. And at the last the transaction T is waiting for transaction T0 in cyclic manner.

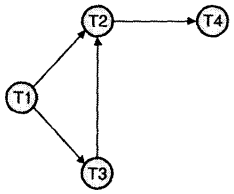
2. The transactions present in deadlock are either rollback or restarted. A system with a deadlock indicates bad behavior of the system. The rollback of the transaction may be partial. That is a transaction may rolled back to the point where it obtained a lock whose release resolves the deadlock.

3. It's not good to have any transaction that causes deadlock. To handle the deadlock situation, we have two options one is to avoid a system to enter in deadlock by Deadlock Prevention method or after deadlock try to recover it by deadlock detection and deadlock recovery method.

Deadlock detection :

1. Detecting deadlock situation in advance is always good instead of aborting a transaction. A deadlock avoidance mechanism is used to detect deadlock situation in advance. An algorithm that examines the state of the system is called periodically to check the occurance of deadlock. In deadlock avoidance "wait-for graph" method is used.

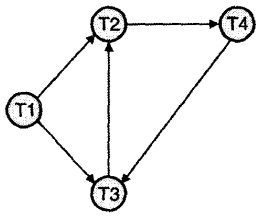
2. **Wait-for Graph:**
   "Wait-for-graph" scheme uses a graph (set of edges and vertices (nodes)) to detect deadlock in the system.
   - A node is created when a transaction enters into system.
   - When a transaction T, requests for a lock on resource X which is held by some other transaction T1, a directed edge is created from T to T1.
   - This edge is removed when T releases resource X, and T1 locks that resource. Wait-for graph indicates which transaction is waiting for another transaction to complete. A graph with cycle indicates the occurrence of deadlock in the system.



   - The wait for graph has following situation
     - Transaction Ti is waiting for transaction T2 and T3

- Transaction T3 is waiting for transaction T2
- Transaction T2 is waiting for transaction T4

- Here the graph has no cycle; hence it is not in deadlock state. Consider the transaction T4 is requesting an item held by T3 then a directed edge from T4 to T3 will be drawn.



Now the graph has a cycle — T2-T4-T3-*T2
Here the transactions T2, T3 AND T4 are in deadlock state.

Q.4(f) Explain shadow paging recovery in detail. **[5]**

Ans.: **1.** The shadow paging does not require the use of a log in a single-user environment. In this scheme database is considered to be made up of a number of fixed-size disk pages or blocks for recovery purposes.

2. A directory with n entries is created. In it the ith entry points to the ith database page on disk. This directory is kept in main memory. It records references of all reads or writes to database pages. When execution of transaction is started we copy the current directory whose entries points to the most recent database pages on disk into a shadow directory.

3. The current directory is used by the transaction and the shadow directory is saved on disk.

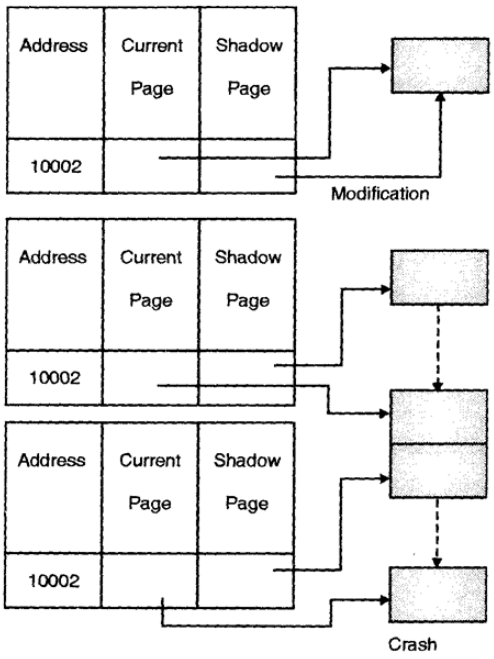
4. While executing the transaction, no changes are made in shadow directory. If a 'write' operation is performed, then new copy of modified database page is created. Both new and old copies are kept. That means for pages which are updated during the transaction, old and new, both versions are kept. The shadow directory refers the old version while the current directory refers the new version.

5. In case of failure, for recovery purpose the modified database pages are freed and current directory is discarded. The shadow directory provides the state of the database before transaction execution. The state is recovered by restoring the shadow directory. In this way the database is returned to its state prior to the transaction that was executing when the crash occurred and any modified pages are discarded.

6. This technique can be categorized as a NO-UNDO/NO-REDO technique for recovery of database because for recovery, neither undoing nor redoing data items is performed.

7. In case of multiuser system where transactions are performed concurrently, logs and checkpoints are used in the shadow paging technique.

8. One of disadvantage of this technique is that the database pages which are updated, change location on disk. Because of this, it is difficult to keep related database pages close together on disk. It requires complex storage management strategies to manage those pages.



Q.5 Attempt the following (any THREE) [15]

Q.5(a) What is a cursor? Explain implicit and explicit cursors. [5]

Ans.: Cursor:

1. Cursor is a pointer to the private area in SQL that stores information about processing a specific DML statement. The drawback with select statement is that it returns only one row at a time in a PL/SQL block.
2. Cursors are capable of holding more than one row. It used for grouping a set of rows and implementing a similar logic to all the records of that group. Set of rows retrieved in such an area is called the active set and its size depends on the number of rows retrieved by the search condition of the query.
3. Oracle reserves an area in memory called cursor, populates this area with appropriate data and frees the memory area when the task is completed.
4. Cursors can be classified as:
   - **Implicit Cursor:** These cursors are also called as internal cursors and are managed by Oracle itself
   - **Explicit Cursor:** These are User-defined Cursor used for external Processing

Implicit Cursor:

1. Implicit cursor is a session cursor which is opened every time you run a SELECT or DML statement in the PL/SQL block. An implicit cursor closes after its associated statement run but its attributes values remain available until another SELECT or DML statement is executed. Cursor attributes return information about the state of the cursor. The syntax for an implicit cursor attribute is SQLattribute (e.g. SQL%FOUND). SQLauribute will always refer to the recently run DML or SELECT INTO statement.

2. Implicit cursor attributes are:

- **SQL%ISOPEN:** Oracle automatically opens and closes implicit cursors associated to any DML or Select statement. Therefore the value for this attribute is always false.
- **SQL % FOUND:** If any DML or Select into statement returned any row then the value of this attribute is true, if no rows were returned then the value is false otherwise when no recent statement is executed then this value is null.
- **SQL%NOTFOUND:** It works like the SQL%FOUND but contrary results are obtained with SQL%NOTFOUND that is false if rows are returned, true if no rows are returned and null if no recent statement is executed.
- **SQL % ROWCOUNT:** It stores null if no recent DML statements are executed and stores the count of the rows affected by the DML statement of Select into statement.

Example: Using CURSOR attributes:

```
SQL>SET SERVEROUTPUT ON
2 BEGIN
3UPDATE emp SET emp_sal  = emp_sal +1000 WHERE dept_no = 'D001';
4 IF SQL%FOUND THEN
5 DBMS_OUTPUT.PUT_LINE ('Salary Updated.....');
6 END IF;
7 IF SQL%NOTFOUND THEN
8 DBMS_OUTPUT.PUT_LINE ('Record not found.....');
9 END IF;
10 IF SQL%ROWCOUNT > 0 THEN
11 DBMS_OUTPUT.PUT_LINE (SQL%ROWCOUNT ' || 'rows are updated');
12 END IF;
13 ELSE
14 DBMS_OUTPUT.PUT_LINE ('No records to be updated');
15 END IF;
16 END;
17 /
```

3. **Drawbacks of Implicit Cursors:** The implicit cursor has the following drawbacks:

- Implicit cursors are less efficient than an explicit cursor.
- Implicit cursors are more vulnerable to data errors.
- Implicit cursors provide less flexibility to implement programming control.

Explicit Cursor: The Cursor which is declared by user is called as Explicit Cursor. Following are the steps of using an explicit Cursor:

- DECLARE Cursor in the declaration section.
- OPEN the cursor in the Execution Section.
- FETCH the data from cursor into PL/SQL variables or records in the Execution Section.
- CLOSE the cursor in the Execution Section before we end the PL/SQL Block.

Example:

```
CURSOR c_emp IS SELECT emp_no, emp_name, emp_sal, emp_addr FROM emp;
```

Statements	Action performed
OPEN c_emp;	Opens the cursor
FETCH c_emp INTO var_emo_no, var_emp_name, var_emp_sal, var_emp_addr;	Rows fetched from table into variables
ClOSEc_emp;	Closes the cursor

Example : Using cursor variables

```
1 SQL> Declare
2 var_emp_no char(4);
3 var_emp_namevarchar(10);
4 var_emp_sal number(8,2);
5 var_emp_addrvarchar(10);
6 cursor c_emp is
7 select emp_no, emp_name, emp_sal, emp_addr from emp;
8 begin
9 open c_emp;
10 loop
11 fetch c_emp into var_emp_no,var_emp_name,var_emp_sal,var_emp_addr;
12 exit when c_emp%notfound;
13 var_new_sal := var_emp_sal +5000;
14 insert into emp
15 values(var_emp_no,var_emp_name,var_new_sal,var_emp_addr);
16 end loop;
17 close c_emp;
18 end;
19 /
```

Q.5(b) Write a PL/SQL block that is used to display Fibonacci series of a number entered by user. **[5]**

Ans.:

```
declare
    first number:=0;
    second number:=1;
    third number;
    n number:=&n;
    i number;

begin
    dbms_output.put_line('Fibonacci series is:');
    dbms_output.put_line(first);
    dbms_output.put_line(second);

    for i in 2..n
    loop
        third:=first+second;
        first:=second;
        second:=third;
        dbms_output.put_line(third);
    end loop;
end;
/
```

Q.5(c) Explain the types of PL/SQL exceptions. **[5]**

Ans.: 1. Named system exception: The Named system exceptions are exceptions that have been already given names in the STANDARD package in PL/SQL. Named system exceptions need not be declared explicitly, they are raised implicitly when a predefined Oracle error occurs, also they are caught by referencing the standard name within an exception handling routine.

2. Unnamed System Exceptions: System exceptions for which oracle do not provide names are known as unnamed system exception. These Exceptions have a code and an associated message. There are two ways to handle unnamed system exceptions:
Using the WHEN OThERS exception handler, or Associating the exception code to a name and using it as a named exception. It is possible to assign a name to unnamed system exceptions using a Pragma called EXCEPTION_INIT. EXCEPTION_INIT will associate a predefined Oracle error number to a user_defined exception.

3. Named Programmer-defined exception: Due to the complex structure of relational databases it is not always possible to handle all the unwanted exceptions using Named and Un-named exceptions. Therefore there is a need to name and catch exceptions that aren't defined by PL/SQL. These are called Named Programmer-defined exceptions. Named Programmer-defined exceptions should be explicitly declared in the declaration section. They should be explicitly raised in the execution section. They should be handled by referencing the user-defined exception name in the exception section.

```
CREATE [OR REPLACE] FUNCTION function_name
[(param1 [, param2])]
RETURN return_datatype
IS | AS
[declaration_section]
Exception_name EXCEPTION;
BEGIN
Executable_section
RAISE exception_name;
EXCEPTION
WHEN exception_name THEN
[statements]
WHEN OTHERS THEN
[statements]
END [function_name];
```

Q.5(d) List the difference between a procedure and a function. **[5]**

Ans.:

Sr. No.	Procedure	Function
1	Procedure does not return a value	Function always returns a value
2	Procedures cannot be called from functions	Functions can be called from procedures
3	Exception can be handled by try-catch block in a Procedure	Try-catch block cannot be used in a Function.
4	Procedure allows SELECT as well as DML (INSERT/UPDATE/DELETE) statement	Function allows only SELECT statement in it.
5	Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section	Function can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section
6	Syntax: CREATE [OR REPLACE] PROCEDURE	Syntax: CREATE [OR REPLACE] FUNCTION

	procedure_name [(parameter [,parameter])] IS [declaration_section] BEGIN executable_section [EXCEPTION exception_section] END [procedure_name];	function_name [(parameter_name [IN | OUT | IN OUT] type [, ...])] RETURN return_datatype {IS | AS} BEGIN < function_body > END [function_name];

Q.5(e) Explain the structure of PL/SQL blocks. How is a sub program different from anonymous blocks? [5]

Ans.: 1. The basic unit of PL/SQL is called a block, which allows grouping of related declarations and statements. Any PL/SQL program is made up of blocks, which can be nested within each other. PL/SQL blocks can be categorized as:

(a) A subprogram is a PL/SQL block stored in the database and called by name from any application. A subprogram could be a procedure or a function.

(b) An anonymous block is a PL/SQL block that appears in the application but is not named or stored in the database. An anonymous block can appear wherever SQL statements can appear.

2. Structure of PL/SQL Block

It consists of the following sections:

```
HEADER — Header section (optional)
           <<label>> (optional)
DECLARE  -- Declarative section (optional)
           -- Declarations of local types, variables, & subprograms
 BEGIN     -- Executable section (required)
           -- Statements (which can use items declared in declarative part)
[EXCEPTION -- Exception-handling section (optional)
           -- Exception handlers for exceptions (errors) raised in executable
             part]
END;
```

(a) Header Section

- Header section is relevant only for subprograms. It includes the name, parameter list and RETURN clause for functions. it determines the way the subprogram must be called.

(b) Declaration Section

- Declaration section of a PL/SQL Block starts with the keyword DECLARE. Declare section is optional and is used to declare temporary placeholders like variables, constants, records and cursors, used to manipulate data in the execution section.

(c) Execution Section

- Execution section starts with the reserved keyword BEGIN and ends with END. This section is a mandatory and is the section where the program logic is written.

- Looping statements, conditional statement and SQL statements form a part of execution section.

(d) Exception Section

- The Exception section starts with the reserved keyword EXCEPTION. Errors in the program can be handled in this section, so that the PL/SQL Blocks terminates gracefully and not abruptly with errors.

(e) Comments can be used to document code. Every statement in the above three sections must end with a semicolon (;).

3. PL/SQL Subprograms

A PL/SQL subprogram is a named PL/SQL block that allows the calling program to supply parameters that can be input only, output only, or input and output values. A subprogram solves a specific problem or performs related tasks. It serves as a building block for modular, reusable and maintainable database applications. A subprogram is either a procedure or a function. Procedures and functions are identical except that functions always return a single value to the calling program, whereas procedures do not.

4. PL/SQL Anonymous Blocks

An anonymous block is an unnamed PL/SQL unit. It is a block of PL/SQL that is coded within a script. Some typical uses for anonymous blocks include:

- Managing control by nesting code within other PL/SQL blocks.
- Initiating calls to subprograms.
- Isolating exception handling
- Anonymous blocks cannot be reused unlike the stored subprograms.
- To execute a PL/SQL block you must code a front slash (I) after the END keyword. DBMS_OUTPUT.PUT_LINE is a function used to generate output on the screen.

For Example, to print a message "Hello world"

```
1 SQL>BEGIN
2 DBMS_OUTPUT.PUT_LINE ('Hello world');
3 END;
4 /
```

❑ ❑ ❑ ❑ ❑