SUMMER – 19 EXAMINATION

Subject Name: Database Management System **Model Answer** **Subject Code: 22319**

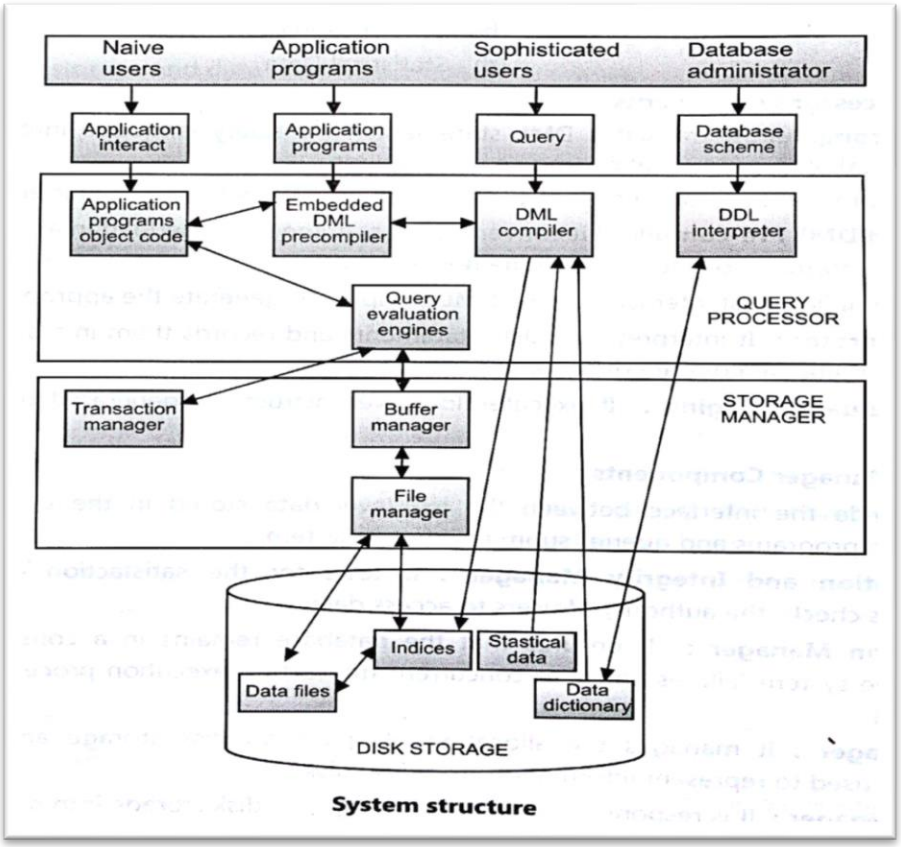
Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1.		**Attempt any FIVE of the following:**	**10 M**
	a	**Define :** **(i) Instance (ii) Schema**	**2 M**
	Ans	**(i) Instance:** The data stored in database at a particular moment of time is called instance of database. **(ii) Schema:** Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.	1 M for each Definition
	b	**List any four advantages of DBMS.**	**2 M**
	Ans	• Controlling Redundancy • Maintaining Integrity • Inconsistency can be avoided • Data can be shared • Restricting unauthorized access • Providing Backup and Recovery • Concurrency Control • Better security.	(½ M for any advantage)
	c	**State any two E.F. Codd's rule for RDBMS.**	**2 M**
	Ans	**1. The Information rule**: All information in an RDBMS is represented logically in just one way - by values in tables.	½ M for each rule , ½ M each

		2. The Guaranteed Access rule: Each item of data in an RDBMS is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name. **3. The Systematic Treatment of Null Values rule**: Null values (distinct from an empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing **4. The Dynamic Online Catalog Based on the Relational Model rule**: The database description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational database. **5. The Comprehensive Data Sublanguage rule:** A relational system may support several languages and various modes of terminal for data definition, view definition, data manipulation etc. **6. The View Updating rule**: All views of the data which are theoretically updatable must be updatable in practice by the DBMS. **7. The High-level Insert, Update, and Delete rule:** The capability of handling a base relation or a derived relation as a single database to perform all DML operations. **8. The Physical Data Independence rule**: Application programs and terminal activities remain logically unchanged whenever any changes are made in either storage representations or access methods. **9. The Logical Data Independence rule:** Application programs and terminal activities remain logically unchanged when information preserving changes of any kind are made to the base tables. **10. The Integrity Independence rule:** Integrity constraints must be definable in the RDBMS sub-language and stored in the system catalogue and not within individual application programs. **11. The Distribution Independence rule:** An RDBMS has distribution independence. Distribution independence implies that users should not have to be aware of whether a database is distributed. **12. The No subversion rule**: If the database has any means of handling a single record at a time that low-level language must not be able avoid the integrity rules which are expressed in a higher-level language that handles multiple records at a time.	proper statement
	d	**List DCL commands.**	**2 M**
	Ans	**DCL is Data Control Language:** **1.** GRANT **2.** REVOKE	1 M for each command
	e	**Define Normalization and list its types.**	**2 M**
	Ans	**Normalization** is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.	1 M for definition, 1 M for the types

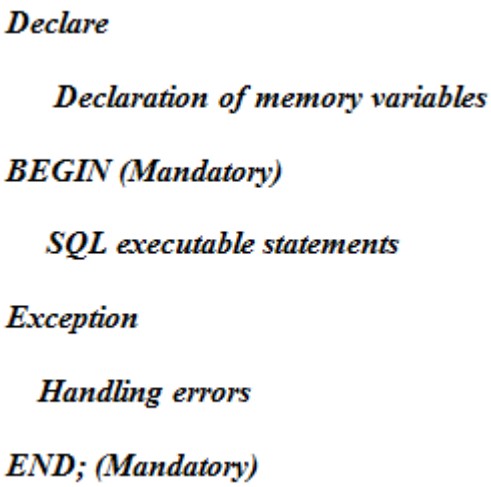
		Types of normalization are : • First normal form(1NF) • Second normal form(2NF) • Third normal form(3NF) • Boyce & Codd normal form (BCNF) • Fourth normal form(4NF)	
	f	**Write syntax for creating synonyms with example**	**2 M**
	Ans	**Syntax to create synonym:** CREATE SYNONYM SYNONYM_name FOR Table_name; **Example to create synonym:** CREATE SYNONYM offices FOR locations;	1 M for correct syntax, 1 M for correct example
	g	**State any four PL/SQL data types.**	**2 M**
	Ans	1. NUMBER or NUMBER(P,S) 2. PLS_INTEGER 3. CHAR 4. RAW 5. ROWID 6. VARCHAR2 7. DATE	½ M for each data type
2		**Attempt any THREE of the following:**	**12 M**
	a	**Explain overall structure of DBMS with the help of diagram.**	**4 M**
	Ans	**Components of DBMS structure are classified in 3 categories as:** **1. Query processor :** **Embedded DML pre compiler:** It converts DML statements embedded in application. Program to normal procedural calls in host language. **DML Compiler:** It translates DML statements of high level language into low level instruction that a query evaluation engine understands. **DDL interpreter:** It interprets DDL statements and records them in a set of tables containing metadata. **Query evaluation Engine:** It executes low level instructions generated by DML compiler and issued by query processor to select efficient ways to execute query. DDL interpreter. It has following components,	2 M for correct diagram, 2 M for correct explanation

		2. Storage Manager Components : **Transaction manager:** It ensures that the database remains in consistent state despite of the system failure and that concurrent transaction execution proceeds without conflicting. **File Manager:** It manages the allocation of space on disk storage and data structures used to represent information stored on disk **Buffer Manager:** It is responsible for fetching data from disk storage into main memory and deciding what data to cache memory. **3. Disk storage :** Data files: It stores the database. Data Dictionary: It stores metadata that hold particular values. Indices: Provide fast access to data items that hold particular values. **Statistical data:** It stores statistical information about the data in the database.  **System structure**	
	b	**Explain difference between delete and truncate command with example.**	**4 M**
	Ans	**DELETE Command :** • It is DML (Data Manipulation Language) command.	

		• It is used to remove all or specific records of table. • WHERE clause can be used to remove specific records. • **Syntax:** DELETE FROM Table_name; OR DELETE FROM Table_name WHERE Condition; • **Example:** DELETE FROM Employees WHERE Emp_id=100; • ROLLBACK command can be used to get deleted record. **TRUNCATE Command :** • It is a DDL(Data Definition Language) command • It is used to remove all records permanently. • WHERE clause can be used as it removes all records. • **Syntax:** TRUNCATE TABLE Table_name; • **Example**: TRUNCATE TABLE Employees; • ROLLBACK command cannot be used to get records. • New records can be added into a table as structure remains intact. OR	(2 M for proper explanation of each command) or (any 4 differences)

DELETE	TRUNCATE
It is DML(Data Manipulation Language) command	It is a DDL(Data Definition Language) command
It is used to remove all or specific records of table.	It is used to remove all records permanently.
WHERE clause can be used to remove specific records.	WHERE clause can be used as it removes all records.
Syntax: DELETE FROM Table_name; OR DELETE FROM Table_name WHERE Condition;	Syntax: TRUNCATE TABLE Table_name;
Example: DELETE FROM Employees WHERE Emp_id=100;	Example: TRUNCATE TABLE Employees;
ROLLBACK command can be used to get deleted record.	ROLLBACK command cannot be used to get records. New records can be added into a table as structure remains intact.

	c	**Write and explain syntax for creating view with example.**	**4 M**
	Ans	A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. **View has two types:** **1**. **Simple view**: The fields in a view are fields from one table in the database. **2. Complex view**: The fields in a view are fields from more than one table in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from different table. **CREATE VIEW Syntax** Create view view_name As	2 M for correct syntax, 1 M for explanation, 1 M for correct example

		Select column1, column2… From table_name Where condition ; **Example** Create view mumbai_customers AS Select customer_name,contact_name From customers Where city='Mumbai';	
	d	**Explain PL/SQL block structure with the help of diagram.**	**4 M**
	Ans.	**PL/SQL Block Strucure :**  *Declare* *Declaration of memory variables* *BEGIN (Mandatory)* *SQL executable statements* *Exception* *Handling errors* *END; (Mandatory)* **Explanation of PL/SQL Block Strucure:** **Declaration section** A block begins with declarative section where variables, cursors are declared. It is an Optional block. **Execution section** Executable SQL or PL/SQL Statements are needed to write here	PL/SQL block structure 2M, Explanation 2M

for the execution. It is mandatory block.

Exception section
It is used to handles the exceptions. It is an Optional block.

End statement
It is used to indicate termination of PL/SQL block. It is mandatory.

3		**Attempt any THREE of the following:**	**12 M**
	a	**State and explain 2NF with example.**	**4 M**
	Ans	A table is said to be in 2NF if both the following conditions hold: • Table is in 1NF (First normal form) • No non-prime attribute is dependent on the proper subset of any candidate key of table. • San attribute that is not part of any candidate key is known as non-prime attribute. • **Example**: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.	State : 1M Explanation with example: 3M

teacher_id	Subject	teacher_age
111	Math's	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

CandidateKeys:{teacher_id,subject}

Non-prime attribute: teacher_age The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non-prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table "To make the table complies with 2NF we can break it in two tables like this:
teacher details tab

teacher_id	teacher_age
111	38
222	40
333	40

teacher_subject Table:

Teacher_id	Subject
111	Math's
111	Physics
222	Biology
333	Physics
333	Chemistry

	b	**Explain any four aggregate functions with example.**	**4 M**
	Ans	An aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning. Aggregate functions are : 1) Count() 2) Sum() 3) Avg() 4) Min() 5) Max() 1. **Count ()** - 1) It returns number of rows from the given table if no attribute is mentioned. 2) If some attribute is mentioned, it gives total number of not null values	Any 4 aggregate functions with example : 1M each

		for that attribute. Eg :Select count(*) from emp; Returns total number of records from emp table. 1) Select count(telephone) from emp; Returns total number of employees having telephone numbers. 2. **Sum()** - It give total of all values from a numeric attribute of the given table, Eg :Select sum(salary) from emp; Returns total salary drawn of all employees from the emp table. 3. **Avg ()** - It gives average of all the numeric values of the given attribute from the table. Eg :Select Avg(salary) from emp; Returns average salary of employees from emp table. 4. **Min ()** - It gives minimum of all the values of the numeric given attribute from the table. Eg :Select Min(salary) from emp; Returns minimum salary value from emp table, 5. **Max ()** - It gives maximum of all the values of the numeric given attribute from the table. Eg :Select Max(salary) from emp; retunes maximum salary value from emp table,	
	c	**Explain exception handling in PL/SQL with example.**	**4 M**
	Ans	**Exception handling in PL/SQL:** An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions − • System-defined (built in) exceptions	Explanation : 2M , example :2M

- User-defined exceptions

The general syntax for exception handling is as follows :

```
DECLARE
<declarations section>
BEGIN
<executable command(s)>
EXCEPTION
<exception handling goes here >
WHEN exception1 THEN
exception1-handling-statements
WHEN exception2  THEN
exception2-handling-statements
……
……..
END;
```

Raising Exceptions

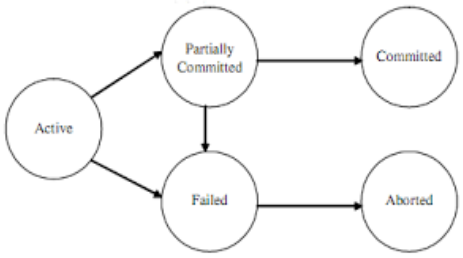
Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command **RAISE**. Following is the simple syntax for raising an exception

```
DECLARE
exception_name EXCEPTION; BEGIN
IF condition THEN
RAISE exception_name;
END IF;
EXCEPTION
WHEN exception_name THEN
statement;
END;
```

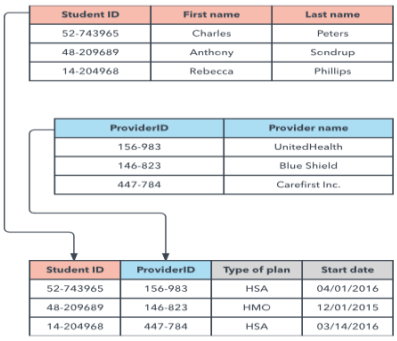
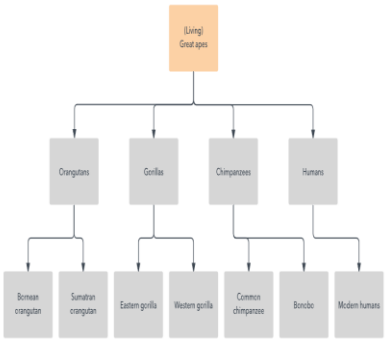
You can use the above syntax in raising the Oracle standard exception or any user-defined exception.

Example :

```
DECLARE
        A number:=20;
        B number:=0;
        C number;
BEGIN
        dbms_output.put_line('First Num : '||A);
        dbms_output.put_line('Second Num : '||B);
```

<table>
<tr><td></td><td></td><td>C:= A / B;
--Raise built in Exception if B is 0
dbms_output.put_line(‘ Result ’ || C);-- and then Result will not be displayed
EXCEPTION
WHEN ZERO_DIVIDE THEN
dbms_output.put_line(‘ Trying to Divide by zero :: Error ’);
END;</td><td></td></tr>
<tr><td></td><td>d</td><td>Explain states of transaction with the help of diagram.</td><td>4 M</td></tr>
<tr><td></td><td>Ans</td><td> 

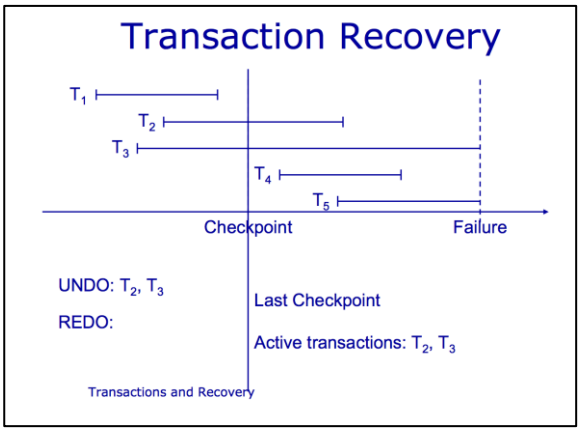
Active –the initial state; the transaction stays in this state while it is executing
Partially committed –after the final statement has been executed.
Failed - after the discovery that normal execution can no longer proceed.
Aborted – after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted: restart the transaction - can be done only if no internal logical error kill the transaction Committed –after successful completion.</td><td>diagram : 1M, explanation : 3M</td></tr>
<tr><td>4</td><td></td><td>Attempt any THREE of the following:</td><td>12 M</td></tr>
<tr><td></td><td>a</td><td>State difference between relational and hierarchical model.</td><td>4 M</td></tr>
<tr><td></td><td>Ans</td><td><table><tr><th>Relational model</th><th>Hierarchical model.</th></tr><tr><td>A database model to manage data as tuples grouped into relations(tables)</td><td>A structure of data organized in a tree like model using parent child relationships.</td></tr><tr><td>Arranges data in tables</td><td>Arranges data in tree like structure</td></tr></table></td><td>Any 4 differences : 1M each</td></tr>
</table>

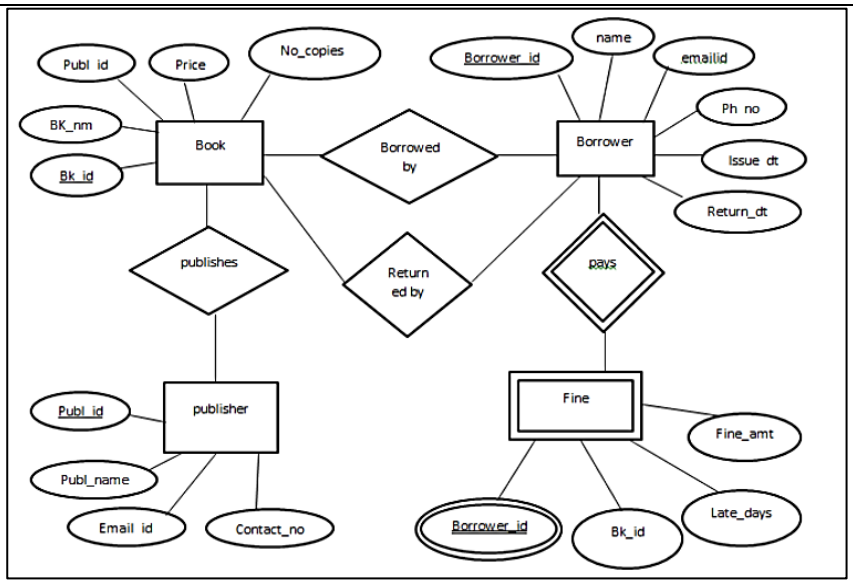
		Represents both "one to many" and"many to many" relationships.	Represents "one to many" relationship	
		Easier to access data	Difficult to access data	
		Flexible	Less flexible	
		Example :  	Example :  	
	b	**List the SQL operations and explain range searching operations between and pattern matching operator 'like' with example.**		**4 M**
	Ans	**Types of SQL operators** : 1) SQL Arithmetic Operators 2) SQL Comparison Operators 3) SQL Logical Operators **Arithmetic operators** are used to perform arithmetic operations on numbers. They are +,-,*, / and %. **Comparison operators** are used in between two variables to compare their values. They are <,>,<=,>=,=,!=or <>,!< and !>.' **Logical operators** are used for the Boolean results in sql queries for comparison of values from the attributes of the tables. Eg: Any, Exists, All, Like, Between, In etc. **Between operator:** The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value inclusive of both the limits. Eg: select * from emp where salary between 40000 and 50000; This will results in rows from emp table where salary falls in the range of 40000 to 50000.		List of operators : 2M, between operator : 1M, Like operator : 1M

		Like operator : The LIKE operator is used to compare a value to similar values using wildcard operators. It uses two wild characters as '%' and '_' where '%' represents all characters of the pattern and '_' represents one single character from pattern. Eg : Select ename from emp where ename like 'S%'; This will return all employee names starting with 'S'. Select ename from emp where ename like '_a%; This will return all employee names whose second character is 'a'.	
	c	**Explain cursor with example.**	**4 M**
	Ans	A cursor is a temporary work area created in system memory when a SQL statement is executed. A cursor is a set of rows together with a pointer that identifies a current row. It is a database object to retrieve data from a result set one row at a time. It is useful when we want to manipulate the record of a table in a singleton method, in other words one row at a time. In other words, a cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set. Each cursor contains the followings 4 steps, 1. Declare Cursor: In this part we declare variables and return a set of values. 2. Open: This is the entering part of the cursor. 3. Fetch: Used to retrieve the data row by row from a cursor. 4. Close: This is an exit part of the cursor and used to close a cursor. 5. Eg: Declare enumemp.eno%type; enemp.ename%type; Cursor cur is select eno, ename from emp where jobname = "mgr"; Begin	Explanation : 2M, example : 2M

		Open cur; Loop Fetch cur into enum,en; Exit when cur%NOTFOUND; Dbms_output.put_line(„emp num ‟||enum||‟ emp name „||en); End loop; Close cur; End; / The example shows fetching multiple records using cursor. A cursor is a temporary work area created in system memory when a SQL statement is executed. A cursor is a set of rows together with a pointer that identifies a current row. In the example, the cursor is defined to hold the rows as defined by the select query. Once the cursor is defined, the next step is to open the cursor. When the cursor is opened, it is ready to retrieve the rows. This is done using the fetch statement. Since there are many rows, a loop is used to display the values of all the rows. Once the rows are fetched, the cursor should be closed.	
	d	**State the use of database trigger and also list types of trigger.**	**4 M**
	Ans	**Use of trigger** Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. A trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated. Triggers are written to be executed in response to any of the following events − A database manipulation (DML) statement (DELETE, INSERT, or UPDATE) Database definition (DDL) statements (CREATE, ALTER, or DROP). A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN). Triggers can be defined on the table, view, schema, or database with which the event is associated.	Use : 3M List of types : 1M

		Triggers can be written for the following purposes – ● Generating some derived column values automatically ● Enforcing referential integrity ● Event logging and storing information on table access ● Auditing ● Synchronous replication of tables ● Imposing security authorizations ● Preventing invalid transactions **Types of trigger** • DML Triggers • DDL Triggers • Logon Triggers	
	e	**Explain recovery techniques with example.**	**4 M**
	Ans	When recovering the database, it is must redo the effects of the previous transactions. This is called Rolling Forward or simple Forward Recovery**.** Not all but some active transaction that didn't complete successfully needs to rollback, when the disk drive crashed. Such kind of rollback is called Backward Recovery. **The Redo Log and Rolling Forward (REDO operation)** The redo log is a set of operating system files that record all changes made to any database buffer, including data, index, and rollback segments, whether the changes are committed or uncommitted. The redo log protects changes made to database buffers in memory that have not been written to the data files. The first step of recovery from an instance or disk failure is to roll forward, or reapply all of the changes recorded in the redo log to the data files. Because rollback data is also recorded in the redo log, rolling forward also regenerates the corresponding rollback segments. Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time. Rolling forward usually includes online redo log files and may include archived redo log files. After roll forward, the data blocks contain all committed changes as	Explanation : 3M, Example 1M

		well as any uncommitted changes that were recorded in the redo log. **Rollback Segments and Rolling Back (UNDO operation)** Rollback segments record database actions that should be undone during certain database operations. In database recovery, rollback segments undo the effects of uncommitted transactions previously applied by the rolling forward phase. After the roll forward, any changes that were not committed must be undone. After redo log files have reapplied all changes made to the database, then the corresponding rollback segments are used. Rollback segments are used to identify and undo transactions that were never committed, yet were recorded in the redo log and applied to the database during roll forward. This process is called rolling back.  (**Descriptive example can be considered**)	
5		**Attempt any TWO of the following:**	**12 M**
	a	**Draw an E-R diagram of library management system considering issue and return,** **Fine calculation facility. Consider appropriate entities.**	**6M**

	Ans		Correct entities: 2M, correct symbols: 2M, Correct relationships: 2M
	b	**Consider the table** **Student (name, marks, dept, age, place, phone, birthdate). Write SQL query for following.** **i)To list students having place as 'Pune' or 'Jalgaon'** **ii)To list students having same department(dept) as that of 'Rachana'** **iii) To change marks of 'Rahul' from 81 to 96.** **iv) To list student name and marks from 'Computer' dept.** **v) To list student name who have marks less than 40.** **vi)To list students who are not from 'Mumbai;**	**6M**
	Ans	select name from Student where place= 'Pune' or place='Jalgaon'; **(OR)** select name from Students where place in('Pune','Jalgaon'); ii)select name from Student where dept=(select dept from student where name='Rachana'); iii)update Student set marks=96 where name= 'Rahul'; v)select name,marks from Student where dept='Computer'; iv)select name from Student where marks<40; v)select * from Student where place != 'Mumbai';	Each Correct Query : 1M
	c	**Create simple and composite index. Write command to drop above index.**	**6M**
	Ans	**<u>Create simple index</u>** Syntax: Create index index_name on <tablename><column name>; **(OR)**	Simple index 2M, Composite

		E.g.: Create index idx_empno on employee (empno); **Create composite index:** Syntax: Create index index_name on <tablename><Column_name1, Column_name2>; **(OR)** E.g.: Create index idx_ename_eno on employee (ename, empno); **Drop Index:** Syntax: Drop index <index_name>; **(OR)** *E.g. (Assuming idx_empno created on employee table)* **Drop index idx_empno;**	index: 2M Drop index 2M (Note: Either syntax or example can be considered. Any other example allowed.)
6		**Attempt any TWO of the following:**	**12 M**
	a	**i) Write a command to create table student(RNO,name marks, dept) with proper datatypes and RNo as primary key** **ii) Write a command to create and drop sequence.**	**6M**
	Ans	i) create table student (RNO number(5) constraint student_RNO_pk primary key, name varchar2(20), marks number(4), dept varchar2(20)); **(OR)** create table student (RNO number(5) , name varchar2(20), marks number(4), dept varchar2(20), constraint student_RNO_pk primary key(RNO),); ii) **Create Sequence:** Create sequence <seq_name> Start with [initial value] Increment by [value] Minvalue [minimum value] Maxvalue [maximum value] [cycle/no cycle] [{cache value / No cache}] [{order / No order}];	Correct query: 3M Create sequence : 2M Drop sequence :1M (Note: For (ii) Either syntax or example can be considered. Any other example allowed)

		(OR) *(*Creating sequence for Employee number of emp table.*)* Create sequence emp_eno_seq start with 1 increment by 1 maxvalue 100 no cycle no cache; **Drop sequence:** Drop sequence<Sequence Name>; (OR) Drop sequence emp_eno_seq;	
	b	**Write a PL/SQL program to calculate factorial of a given number.**	**6M**

	Ans	DECLARE
		num number:=#
		fact number:=1;
		BEGIN
		
		while num!=0
		loop
		fact:=fact*num;
		num:=num-1
		end loop;
		dbms_output.put_line('Factorial ='||fact);
		END;
		/
		(OR)
		
		DECLARE
		
		num number:=#
		
		fact number:=1;
		
		i number;
		BEGIN
		for i in 1..num loop
		fact:=fact*i;
		end loop;
			Correct Syntax: 3M,Correct logic : 3M
			
			(Note: Any other logic can be considered**)**

		dbms_output.put_line('Factorial='||fact); END; /	
	c	**Write SQL command for following** **i)Create user** **ii) Grant privileges to user.** **Iii) Remove privileges from user.**	**6M**
	Ans	**i)Create user** CREATE USER <username> IDENTIFIED BY <password>; **(OR)** CREATE USER RAJ IDENTIFIED BY RAJ123; **ii) Grant privileges to user.** GRANT <privilege list> ON <relation name or view name> TO<user list>; **(OR)** *(*assuming table Employee for granting permissions to user *'RAJ' for select, insert,* update and delete privilege) GRANT SELECT, INSERT,UPDATE,DELETE ON EMPLOYEE TO RAJ; **Iii) Remove privileges from user.** REVOKE <privilege list> ON <relation name or view name > FROM <user list>; **(OR)** (assuming table Employee for revoking permissions to user 'RAJ) REVOKE SELECT, INSERT,UPDATE,DELETE ON EMPLOYEE FROM RAJ;	Each correct command: 2M (Note: Either syntax or example can be considered. Any other example allowed)