SUMMER– 18 EXAMINATION

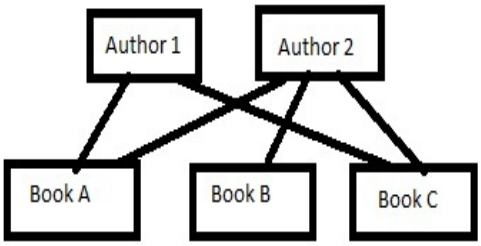
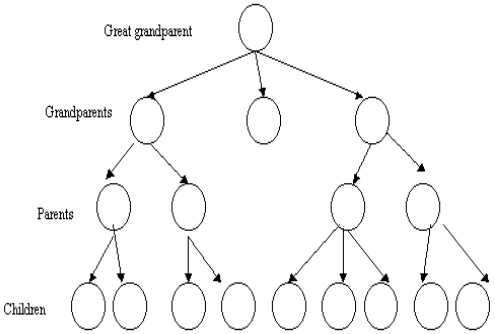
Subject Name: Relational Database Management System **Model Answer** **Subject Code:** 17332

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.	Answers	Marking Scheme
1.	**(A)**	**Attempt any SIX of the following:**	**12 Marks**
	(a)	**Define RDBMS.**	**2M**
	Ans:	RDBMS is Relational Database Management System which is an environment where data is represented in the form of relations, with enforced relationships between the tables.	**(Definition :2 marks)**
	(b)	**Compare Network model and Hierarchical model. (Min. 2 points)**	**2M**
	Ans:		**(Any 2 Points of differences : 1 mark each)**

Network Model	Hierarchical Model
The Network model replaces the hierarchical tree with a graph thus allowing more general connections among the nodes.	Hierarchical Model is like a structure of a tree with the records forming the nodes and fields forming the branches of the tree.
Reflects M:N (Many to many) relationship	Reflects 1:N (One to many) relationship
It allows a record to have more than one parent	There can be only one node at the parent level

		Example:  Example:  	
	c)	**Define Normalization.**	**2M**
	Ans:	Normalization can be defined as process of decomposition/division of database tables to avoid the data redundancy.	**(Definition : 2 marks)**
	d)	**List logical operator.**	**2M**
	Ans:	Logical operators are **1)** AND **2)** OR **3)** NOT	**(Any 2 Operator Listing : 2 marks)**
	(e)	**State any four DDL command.**	**2M**
	Ans:	DDL commands : 1) Create 2) Alter 3) Drop 4) Truncate 5) Rename 6) Desc	**(Any four commands : 1/2 mark each)**
	(f)	**Define sequence.**	**2M**
	Ans:	A sequence refers to a database object that is capable of generating unique and sequential integer values. These numbers can be ascending or descending order. It provides intervals between numbers.	**(Definition : 2 marks)**
	(g)	**List out the types of cursor.**	**2M**
	Ans:	Types of cursor are : 1) Implicit cursor 2) Explicit cursor	**(Each Type Name : 1 mark)**

(h) Define foreign key. — **2M**

Ans: A foreign key is a column (or group of columns) whose values are derived/referred from the 'primary key' of some other table / parent table. — **(Definition : 2 marks)**

(B) Attempt any TWO of the following : — **8 Marks**

(a) Differentiate between RDBMS & DBMS. (Min. 4 points) — **4M**

Ans: — **(Any relevant 4 points of differentiation:1 mark each)**

DBMS	RDBMS
Stores data as file	Stores data in tabular form.
There is no relationship between data stored in DBMS.	There exist a relationship between data values stored in form of table.
Does not support Distributed Database	Supports Distributed Database
Not all Codd rules are satisfied.	All 12 Codd rules are satisfied.
Suitable for low volume of data	Suitable for large volume of data
Security is less.	More security measures provided
Example : Forxpro, dbaseIII plus	Example : Oracle, SQL Server

(b) Explain order by clause with example. — **4M**

Ans: — **(Explanation : 2 marks, Example : 2 marks)**

Order by clause:

- The ORDER BY keyword is used to sort the result-set by one or more columns.
- The ORDER BY keyword sorts the records in ascending order by default.
- To sort the records in a descending order, you can use the DESC keyword.
- Syntax:

```
SELECT column_name,column_name
FROM table_name
ORDER BY column_name,column_name ASC|DESC;
```

Example:

1. SELECT * FROM Customers ORDER BY Country;
2. SELECT * FROM Customers ORDER BY Country DESC;

(c) Describe sequential control in PL/SQL with neat diagram. — **4M**

Ans: — **(Description : 3 marks, Diagram : 1 mark)**

{{ Note: Any other relevant diagram shall be considered**}}**

Sequential Control in PL/SQL :

- Label can be used to give a label or name to the piece of code. And once the name is given to it, user can call that piece of code with that label whenever and wherever required in the program.
- Syntax for label is as follows:

```
<<label_name>>
Statements;
```

- The GOTO statement is used to transfer the control or to change the sequence of the instructions.
- Syntax for GOTO statement is as follows:

GOTO<<label_name>>

OR

Sequential Control Statements
PL/SQL provides a GOTO statement and a NULL statement to aid in sequential control operations.

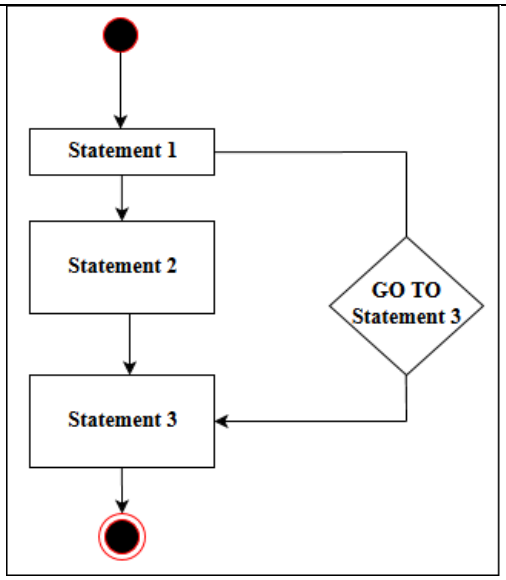
GOTO
The GOTO statement performs unconditional branching to a named label. You should only rarely use a GOTO. At least one executable statement must follow the label (the NULL statement can be this necessary executable statement). The format of a GOTO statement is:

GOTO <<label_name>>;
There are a number of scope restrictions on where a GOTO can branch control. An example of GOTO statement is :

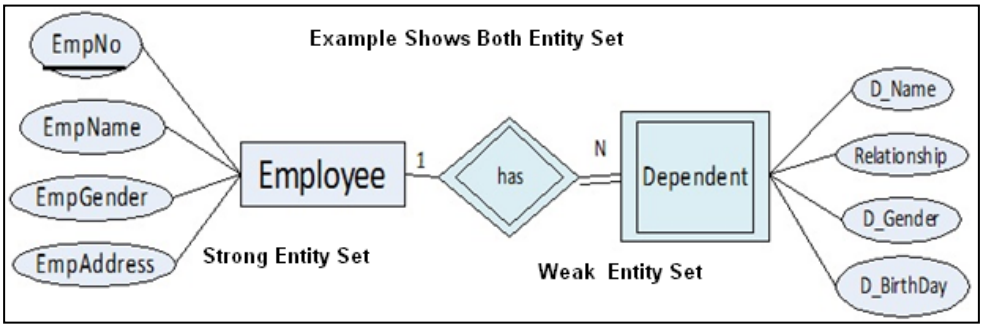
```
BEGIN
  GOTO second_output;
  DBMS_OUTPUT.PUT_LINE('This line will never execute.');
  <<second_output>>
  DBMS_OUPUT.PUT_LINE('We are here!);
END;
```

NULL
The NULL statement is an executable statement that does nothing. It is useful when an executable statement must follow a GOTO label or to aid readability in an IF-THEN-ELSE structure.
For example:

```
IF : report.selection = 'DETAIL' THEN
  exec_detail_report;
ELSE
  NULL;
END IF;
```



2.		**Attempt any FOUR of the following :**	**16 Marks**
	(a)	**Explain weak entity and strong entity set with example.**	**4M**
	Ans:	**Weak entity :** • It does not have sufficient attribute to form a primary key on its own. • The relationship between one strong and a weak entity set is represented by a double diamond symbol. (Known as identifying relationship). • It is represented by double rectangle. The line connecting weak entity set with the identifying relationship is double. **Strong entity:** • It has its own primary key. • The relationship between two strong entity set is represented by a diamond symbol • The member of strong entity set is called as dominant entity set. • It is represented by rectangle. • The line connecting strong entity set with the relationship is single.	**(Weak entity with example:2 marks, Strong entity with example:2 marks)**

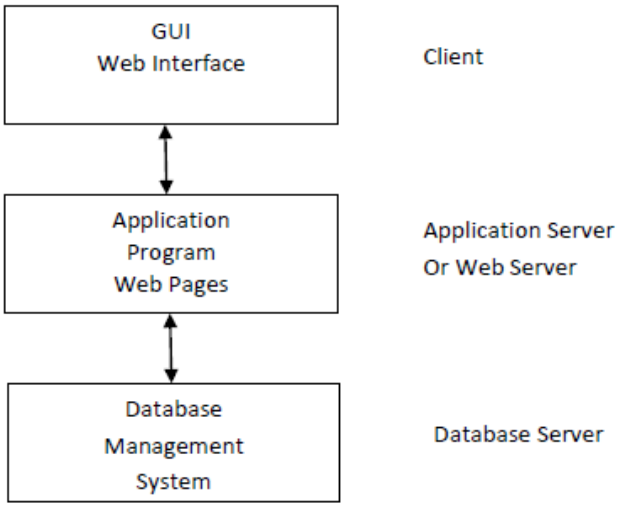


	(b)	**Describe domain relational calculus.**	**4M**
	Ans:	Domain relational calculus uses domain variable that takes values from attributes domain rather than values for entire tuple. Domain relational calculus uses list of attribute to be selected from the relation based on the condition. There are different symbols with specific meaning which can be used to write domain calculus expression;- 1. ε- belong to 2. ∃-There exits 3. ∀ – for all 4. ¬ – not 5. => implies 6. ^ -and 7. ∨ - or An expression is of the form $\{< x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$ where the $x_i, 1 \leq i \leq n$, represent attributes, and P is a **predicate**. Example : Find branch name, loan number, customer name and amount for loans of over $1200. $\{< b, l, c, a > \mid < b, l, c, a > \in borrow \wedge a > 1200\}$ Where 'b' represents domain of branch name, 'l' represents domain of loan number, 'c' represents domain of customer name and 'a' represents domain of amount from table 'borrow'	**(Explanation : 4 marks)**
	(c)	**Demonstrate the use of COMMIT Command with example.**	**4M**
	Ans:	**{{**Note: Any other relevant example shall be considered**}}** The COMMIT command is used to save changes invoked by a transaction to the database. The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command. The syntax for COMMIT command is as follows: SQL> COMMIT; **Example :** Insert into emp (101,'abc','clerk',5000); select * from emp; commit;	**(Explanation : 2 marks, Example : 2 marks)**

<table>
<tr><td></td><td>(d)</td><td>Explain dropping sequence with example.</td><td>4M</td></tr>
<tr><td></td><td>Ans:</td><td>The DROP SEQUENCE command is used to remove the sequence from database.
Syntax:
DROP SEQUENCE<SequenceName>;
Example :
Assuming there is a sequence already created as num_seq, then to drop it, we can write a statement as
Drop sequence num_seq;</td><td>(Explanation : 2 marks, example : 2 marks)</td></tr>
<tr><td></td><td>(e)</td><td>Discuss shared & exclusive rock.</td><td>4M</td></tr>
<tr><td></td><td>Ans:</td><td>The two commonly used locks are 1) Shared and 2) Exclusive
1) Shared Lock: It can lock the transaction only for reading. This lock opens a table/database in read mode.
2) Exclusive Lock: It can lock the transaction for reading as well as writing. This lock opens a table/database in write mode.

The compatibility table for these two locks can be given as:

<table><tr><td></td><td>Shared</td><td>Exclusive</td></tr><tr><td>Shared</td><td>Compatible</td><td>Not compatible</td></tr><tr><td>Exclusive</td><td>Not Compatible</td><td>Not Compatible</td></tr></table>
These are mainly used in two-phase locking protocol to ensure serializibilty.
Syntax: lock table table_name in{share |exclusive} mode</td><td>(Each type : 2 marks each)</td></tr>
<tr><td></td><td>(f)</td><td>Elaborate on three tier architecture with neat diagram.</td><td>4M</td></tr>
<tr><td></td><td>Ans:</td><td>Three Tier Architecture: In this the communication takes place from client to application server and then application server to database system to access the data. The application server or web server is sometimes called middle layer or intermediate layer. The middle layer which processes applications and database server processes the queries. This type of communication system is used in the large applications or the world web applications. On WWW all clients requests for data and server serves it. There are multiple servers used like fax server, proxy server, mail server etc.</td><td>(Explanation : 2 marks, diagram : 2 marks)</td></tr>
</table>



3.		**Attempt any FOUR of the following:**	**16 Marks**
	(a)	**Consider following database EMP (empno, ename, mgr, hiredate, sal, Deptno)** **(i) Select ename, sal, deptno from EMP.** **(ii) List all empno, ename, hiredate from EMP where sal < 5000 AND deptno = 20** **(iii) Write a query to add (city char (20)) to EMP** **(iv) Insert values in EMP** **(7788, 'Asha', Manager 7566, '10 July 1999', 30000, 20)**	**4M**

	Ans:	i. select ename, sal, deptno from EMP;
		ii. select empno, ename, hiredate from EMP where sal < 5000 AND deptno = 20;
		iii. alter table EMP add city char(20);
		iv. insert into EMP values(7788, 'Asha' , 'Manager 7566' , '10-Jul-1999', 30000 , 20);
			(Each correct query:1 mark each)

	(b)	**Explain PL SQL block structure.**	**4M**
	Ans:	**PL/SQL block has following structure:** ***Declare*** (Optional) --Use for declaring variables ***Begin*** (Mandatory) --Use for writing executable code; ***Exception*** (Optional) --Use to write exceptions to be catch during run time. ***End***; (Mandatory) --To terminate PL-SQL block/ code.	**(Correct Explanation:4 marks)**

The Declaration section : Code block start with a declaration section , in which memory variables, constants, cursors and other oracle objects can be declared and if required initialized.

The Begin section: Consist of set of SQL and PL/SQL statements, which describe processes that have to be applied to table data. Actual data manipulation, retrieval, looping and branching constructs are specified in this section.

The Exception section: This section deals with handling errors that arise during execution data manipulation statements, which make up PL/SQL code block. Errors can arise due to syntax, logic and/or validation rule.

The End section: This marks the end of a PL/SQL block.

(c) Describe second normalization form with example. **4M**

Ans: **{{**Note: Any other relevant example shall be considered**}}** **Description :2 marks, Example : 2 marks)**

Second Normal Form (2NF): A relation is said to be in the second normal form if it is in first normal form and all the non key attributes are fully functionally dependent on the primary key.

Example:

Example: Supplier(SNO,SNAME,LOCATION,PNO,QTY)

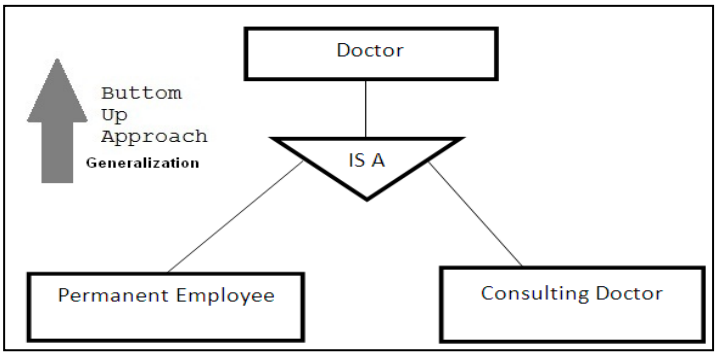
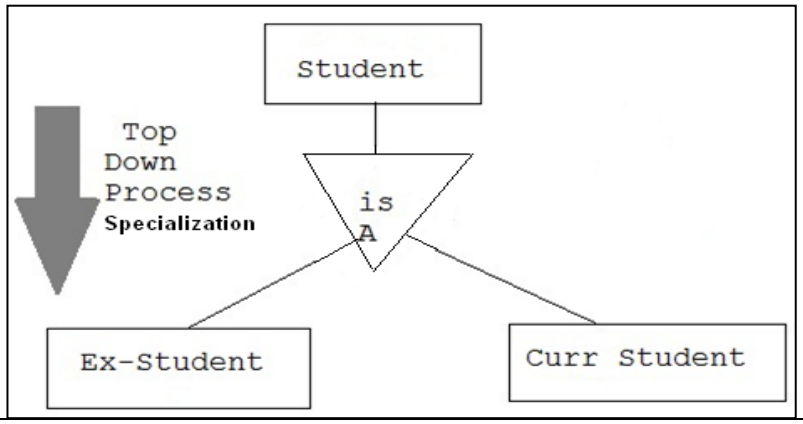
SNO	SNAME	LOCATION	PNO	QTY
S1	Abc	Mumbai	P1	200
S2	Pqr	Pune	P2	300
S3	Lmn	Delhi	P1	400

In the above relation SNAME, LOCATION depends on SNO and QTY on (SNO, PNO) so the table can be split up into two tables as *Supplier(SNO,SNAME,LOCATION)* and *SP(SNO,PNO,QTY)* and now both the tables are in second normal form

SNO	SNAME	LOCATION
S1	Abc	Mumbai
S2	Pqr	Pune
S3	Lmn	Delhi

SNO	PNO	QTY
S1	P1	200
S2	P2	300
S3	P1	400

	(d)	**Paraphrase simple unique indexes with syntax and example.**	**4M**
	Ans:	**1) Simple Index:** An index created on single column of a table is called a Simple Index. **Syntax:** Create index index_name on <tablename><column name>; **Example:** Create index idx on employee (empno); **2) Unique Index:** A unique index is created automatically when you define a primary key or unique constraint in a table definition. It is used not only for performance, but also for data integrity. A unique index does not allow any duplicate values to be inserted into the table. **Syntax :** Create unique index index_name on table_name(column_name); **Example:** Create unique index index_empno on emp(empno);	**(Each index with explanation & example: 2 marks each)**
	(e)	**Explain implicit and explicit locking strategy.**	**4M**
	Ans:	**Implicit locking:** Implicit locks are generally placed by the DBMS automatically. Most DBMS allow the developer or the application to issue locks which are referred to as explicit locks. The default locking is done by the oracle server implicitly by creating deadlock situation when the transaction is done on the same database object (table) in different sessions. It is also called as implicit locking or automatic locking. This lock held till the transaction is completed. **Explicit locking:** It is placed by application program. When locking is done by the user with the help of SQL statement, it is called as explicit locking. There two types of Explicit locking: • Share • Exclusive	**(Each locking strategies: 2 marks each)**
	(f)	**State any four advantages of DBMS over file processing system.**	**4M**
	Ans:	1. **Centralized Management and Control over data:** DBA is a person having central control over the system. 2. **Reduction in Redundancy**: Centralized control of data by DBA avoids unnecessary duplication of data. 3. **Avoiding Inconsistency**: As the redundancy is reduced inconsistency is avoided. 4. **Maintaining Integrity:** Centralized control ensures that adequate checks are incorporated in the database to provide data integrity thus, accuracy is maintained.	**(Any four Advantages :1 mark each)**

		5. **Sharing of data**: DBMS allows sharing of data under its control by any number of application programmer and users. . 6. **Enforcement of Security**: DBA can ensures that proper access procedures are followed, including proper authentication schemas for access to the DBMS and additional checks before permitting access to sensitive data 7. **Conflict Resolution**: DBA resolves the conflicting requirement of various users and applications 8. **Data Independence:** DBA can modify the structure of data record. This modification do not affect other applications.	
4.		**Attempt any FOUR of the following :**	**16 Marks**
	(a)	**Explain specialization & Generalization.**	**4M**
	Ans:	**{{**Note: Diagram is optional**}}** **Generalization:** Generalization is super class in DBMS which holds common properties/ attributes for more than one entities (Specialized class). In generalization, the higher level entity can also combine with other lower level entity to make further higher level entity.  **Specialization:** Specialization holds special attributes of entities which are distinguished from other entities of same type. Specialization is opposite to Generalization. In specialization, some higher level entities may not have lower-level entity sets at all.  	**Specialization: 2 marks, Generalization: 2 marks)**

	(b)	**List user defined exception and explain with suitable example.**	**4M**

	Ans:	**1) User defined exception:**
		It must be declare by the user in the declaration part of the block where the exception is used. It is raised explicitly in sequence of statements using:
		Steps to be followed to use user-defined exceptions:
		1) **They should be explicitly declared in the declaration section.**
		The syntax for declaring an exception is:
		DECLARE
		My_exception EXCEPTION;
		2) They should be explicitly raised in the Execution Section.
		Raise_application_error(EXCEPTION NAME);
		3) They should be handled by referencing the user-defined exception name in the exception section.
		Example to throw user defined exception if number is less than 0.
		declare
		out_of_stock exception;
		number_on_hand NUMBER:=0;
		begin
		if number_on_hand<1 then
		raise out_of_stock; --raise an exception that we defined
		end if;
		exception
		when out_of_stock then
		dbms_output.put_line('Encountered out-of-stock error');
		end;
		/
			(Explanation: 2 marks , Example : 2 marks)

	(c)	**Describe transaction properties.**	**4M**
	Ans:	**Transaction Properties:** **Atomicity:** Either all operations of the transaction are reflected properly in the database, or none are. **Consistency:** Consistency ensures that the database is in consistent state prior and after the execution of transaction. Execution of a transaction needs to take place in isolation. It helps in reducing complications of executing multiple transactions at a time and preserves the consistency of the database.	**(Each property: 1 mark)**

Isolation: Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions Ti and Tj, it appears to Ti that either Tj finished execution before Ti started, or Tj started execution after Ti finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.

Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

(d) Write any five string function with syntax and example of each. **4M**

Ans:

Sr. No	Function (syntax)	Description
1.	Initcap(str)	Converts first letter of string to capital letter. Example: Select initcap('rdbms') from dual;
2.	Lower(char)	Converts a string to all lowercase characters. Example: Select lower('RDBMS') from dual;
3.	Upper(char)	Converts a string to all uppercase chracters. Example: Select upper('rdbms') from dual;
4.	Length(char)	Find outs the length of given string. Example: Select length('RDBMS') from dual;
5.	Ltrim(char,set)	It trim from the left of character string. Example: Select Ltrim('manas khan', 'manas') from
6.	Rtrim(char,set)	It trim from the Right of character string. Example: Select Rtrim('manas khan', 'khan') from dual;
7.	Lpad(char1,length, char2)	It returns char1, left-padded to given length with the sequence of characters in char2. Example: Select Lpad('SKY', 8, '*') from dual;
8.	Rpad(char1,length ,char2)	It returns char1, right-padded to given length with the sequence of characters in char2. Example: Select Lpad('SKY', 8, '*') from dual;
9	Translate(char, from string, to string)	It returns expr with all occurrences of each character in from_string replaced by its corresponding character in to_string.

(Any five string functions with syntax & example:4 marks)

					Example: Select translate(Hickory,'H','D') from dual	
			10	Replace(char,searchstring,[repstring])	It returns character string with each occurrences of searchstring replaced with [repstring] Example: Select replace('Tick and Tock','T','Cl') from dual;	
			11	Substr(char,m,n)	It returns substring of character string that stack at m character and is of length n Example: Select substr(Triangle'4,5) from dual;	
			12	Concat (str, str2)	It merges two or more strings or a string and a data value together Example: select concat('summer ','18') from dual;	
			13	Chr(n)	Returns a character binary equivalent of n. select chr(65) from dual;	
			14	Asci(char)	Returns a decimal representation of a character. select ascii('A') from dual;	
	(e)	**Memorize different features of sequences.**				**4M**
	Ans:	**Sequence:** 1. Sequence is database object that generate/produce integer values in sequential order. 2. It automatically generates primary key and unique key values. 3. It may be ascending or descending order. 4. It can be used for multiple tables. 5. Sequence numbers are stored and generated independently of tables. 6. It saves a time by reducing application code. 7. It is used to generate unique integers. 8. It is used to create an auto number field. 9. Useful when you need to create a unique number to act as a primary key. 10. Oracle provides an object called as a Sequence that can generate numeric values. The value generated can have maximum of 38 digits. 11. Provide intervals between numbers.				**(Any 4 features : 1 mark each)**

	(f)	**Explain the functions of database administrator.**	**4M**
	Ans:	**Functions of DBA:** **1. Schema Definition:** The Database Administrator creates the database schema by executing DDL statements. Schema includes the logical structure of database table (Relation) like data types of attributes, length of attributes, integrity constraints etc. **2. Storage structure and access method definition:** The DBA creates appropriate storage structures and access methods by writing a set of definitions which is translated by data storage and DDL compiler. **3. Schema and physical organization modification:** DBA writes set of definitions to modify the database schema or description of physical storage organization. **4. Granting authorization for data access:** The DBA provides different access rights to the users according to their level. Ordinary users might have highly restricted access to data, while you go up in the hierarchy to the administrator, you will get more access rights. Integrity constraints specifications: Integrity constraints are written by DBA and they are stored in a special file which is accessed by database manager while updating data. **5. Routine Maintenance:** Some of the routine maintenance activities of a DBA is given below. • Taking backup of database periodically • Ensuring enough disk space is available all the time. • Monitoring jobs running on the database. • Ensure that performance is not degraded by some expensive task submitted by some users.	**(Any four functions : 1mark each)**
5.		**Attempt any FOUR of the following:**	**16 Marks**
	(a)	**Explain instance and schema.**	**4M**
	Ans:	**Instance:** The collection of information stored in the databases at a particular moment is called as an instance. The value of various database objects at a moment of time is called the instance of that database. **Schema:** *The overall design of the database/table is known as schema.* The database schemas are partitioned at different level of abstractions. Database schema defines the variable declarations in tables that belong to a particular database;	**(Instance 2: marks, Schema: 2 marks)**

	(b)	**Describe views and join in detail.**	**4M**
	Ans:	**{{**Note:Syntax / Types are optional**}}** **View:** Views are created for security reasons. View is a logical copy of physical table. It doesn't exist physically. With the help of view, we can give restricted access to users. When view is used, underlying table is invisible, thus increasing security. Views can be used to see, insert, update and delete data from base table. **Syntax for creating view:-** Create [OR Replace][Force /Noforce] view <viewname> [alias name ….] As subquery [with CHECK OPTION[CONSTRAINT]] [with READ ONLY]; **Join:** A SQL join is an instruction to combine data from two sets of data (i.e. two tables). A JOIN clause is used to combine rows from two or more tables, based on a related column between them. **Different Types of SQL JOINs** There are the different types of the JOINs in SQL: (INNER) JOIN: Returns records that have matching values in both tables. LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table. RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table. FULL (OUTER) JOIN: Return all records when there is a match in either left or right table.	**(Views: 2 marks, Join: 2 marks)**
	(c)	**Define following terms with example :** **(i) declaring** **(ii) opening** **(iii) closing a cursor**	**4M**
	Ans:	**Declaring:** This term is used to declare a cursor so that memory initialization will take place. A cursor is declared by defining the SQL statement that returns a result set. Example: Declare *CURSOR* Summer_18 *IS* SELECT seat_no, stu_name, percentageFROM candidate;	**(Declaring: 2 marks, Opening: 1 mark, Closing cursor: 1 mark)**

		OR
		
		Declare: Declare statement is use to declare a variable in PL/SQL block.
		Declare
		x number(2);
		y varchar(10);
		begin
		..
		end;
		
		Opening: A Cursor is opened and populates data by executing the SQL statement defined by the cursor.
		Example: -
		Open Summer_18;
		Closing a Cursor: This forces cursor for releasing the allocated memory assigned/ occupied by cursor.
		CLOSE Summer_18;	

	(d)	**Explain following terms with example :** **(i) Procedure** **(ii) Function**	**4M**
	Ans:	**Procedure:** A procedure is named PL/SQL block which perform one or more specified task. Procedure does not return any values. Example *CREATE OR REPLACE PROCEDURE* SUMMER_RDB *[AS]* *BEGIN* dbms_output.put_line('This is Procedure!'); *END;* **Function:** Function is a logically grouped set of SQL and Pl/SQL statements that perform a specific task. A function is same as a procedure except that it returns a value. A function is created using the CREATE FUNCTION statement. Example *CREATE OR REPLACE FUNCTION* Success_cnt RETURN number IS cnt number(7) := 0; BEGIN SELECT count(*) into cnt FROM candidate where result='Pass'; RETURN cnt; END;	**(Procedure: 2 marks, Function: 2 marks)**

	(e)	**Summarize multivalue dependency.**	**4M**
	Ans:	Multivalued dependencies occur when the presence of one or more rows in a table implies the presence of one or more other rows in that same table. OR A multivalued dependency (MVD) X -->> Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, where we use Z to denote (R 2 (X Y)): t3[X] = t4[X] = t1[X] = t2[X]. t3[Y] = t1[Y] and t4[Y] = t2[Y]. t3[Z] = t2[Z] and t4[Z] = t1[Z]. Example 1: For example, imagine a car company that manufactures many models of car, but always makes both red and blue colors of each model. If you have a table that contains the model name, color and year of each car the company manufactures, there is a multivalued dependency in that table. If there is a row for a certain model name and year in blue, there must also be a similar row corresponding to the red version of that same car. **Example 2:** Subject | Text | Teacher MATHEMATICS | ALGEBRA | Mr. John MATHEMATICS | GEOMETRY | Mr. Jack MATHEMATICS | ALGEBRA | Mr. Steve MATHEMATICS | GEOMETRY | Mr. Matt COMPUTER | DATABASEMANAGEMENT | Mrs. Headley COMPUTER | VB.NET | Mrs. Tom In the above relation Text and Teacher are multivalued dependent on Subject. There are two multivalued dependencies in this. {Subject} $\twoheadrightarrow$ {Text} and {Subject}→ {Teacher}	**(Description: 4 marks)**
	(f)	**Explain serializability with example.**	**4M**
	Ans:	**{{**Note: Any other relevant description/ example shall be considered**}}** In concurrent execution of transaction, if the consistency level of the concurrent schedule is same as the consistency level after serial schedule of the same schedule, then that concurrent schedule is called as serializable concurrent schedule and this property of schedule is called as serializability. Serializability ensures consistency of database.	**(Description: 2 marks, Example: 2 marks.)**

Example:

Transaction T1: Rs. 50 of A's account are transferred to B's Account. Transaction T2: 10% of A's balance is transferred to B's Account Consider initial amount as A=100, B=150 so initially A+B=250.

Serial Schedule appears as

T1	T2
Read(A); ……A=100	
A:=A-50;	
Write(A); …..A=50	
Read(B); …..B=150	
B:=B+50;	
Write(B);…...B=200	
	Read(A); …..A=50
	Temp:=A*0.1;
	A:=A-temp;
	Write(A); …..A=45
	Read(B); ……B=200
	B:=B+temp;
	Write(B); ……B=205

At the end of serial schedule, A+B=250; Concurrent schedule will appear as:

T1	T2
Read(A); ……A=100	
A:=A-50;	
Write(A); …..A=50	
	Read(A); …..A=50
	Temp:=A*0.1;
	A:=A-temp;
	Write(A); …..A=45
Read(B); …..B=150	
B:=B+50;	
Write(B); …...B=200	
	Read(B); ……B=200
	B:=B+temp;
	Write(B); ……B=205

		Here also A+B=250 at the end of schedule, so this concurrent schedule preserves consistency and hence it is a serializable schedule and shows serializability property	
6.		**Attempt any FOUR of the following :**	**16 Marks**
	(a)	**Elaborate on distributed database with example.**	**4M**
	Ans:	**Distributed Database:** Distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network. A Distributed Database Management System (DDBMS) consists of a single logical database that is split into a number of fragments. Each fragment is stored on one or more computers under the control of a separate DBMS, with the computers connected by a communications network. Each site is capable of independently processing user requests that require access to local data and is also capable of processing data stored on other computers in the network. **Example:** Using distributed database technology, a bank may implement their database system on a number of separate computer systems rather than a single, centralized mainframe. The computer systems may be located at each local branch office: for example, Mumbai, Pune and Nagpur. A network linking the computer will enable the branches to communicate with each other, and DDBMS will enable them to access data stored at another branch office. Thus a client living in Pune can also check his/her account during the stay in Mumbai or Nagpur	**(Description: 2 marks, Example: 2 marks)**
	(b)	**What is Referential Integrity Constraints? Explain with example.**	**4M**
	Ans:	**Referential Integrity Constraint:** It is a relational database concept in which multiple tables share a relationship based on the data stored in the tables, and that relationship must remain consistent. A value of foreign key is derived from primary key which is defined in parent table. Syntax: CREATE TABLE TABLE_NAME (COLUMN_NAME DATA_TYPE, COLUMN_NAME DATA_TYPE CONSTRAINT CONSTRAINT_NAME *REFERENCES* PARENT_TABLE_NAME (PARENT_TABLE_COL_NAME ON DELETE CASCADE, COLUMN_NAME DATA_TYPE); **Example:** CREATE TABLE DEPARTMENT (EMP_ID NUMBER(5) REFERENCES EMP(EMP_ID), DNO NUMBER(3)); After table creation the foreign key is added as: Alter table product add constraint fk_prod foreign key (EmpId) references Emp (EmpId);	**(Referential Integrity Constraint : 2 marks, Example: 2 marks)**

	(c)	**Write the syntax and example of CREATE and UPDATE command.**	**4M**
	Ans:	**CREATE COMMAND: -** Create Command is used to create database object like table, trigger, view, cursor, sequence etc. Syntax: - CREATE DATABASE_OBJECT OBJ_NAME (PROPERTIES); i.e. CREATE TABLE TABLE_NAME (ATTRIBUTE DATA_TYPES, ….); CREATE VIEW VIEW_NAME AS (SELECT STATEMENT); Example: CREATE TABLE SUMMER_18 (SEAT_NO NUMBER(6), PERCENTAGE NUMBER(4,3)); **UPDATE COMMAND: -** Update command is use to change/modify the existing data/value in table, view. Syntax: - UPDATE OBJ_NAME SET ATTRIBUTE = *value* WHERE (*condition*); UPDATE EMP SET deptid = 10 WHERE empid = 789;	**(Create Command : 2 marks, Update Command : 2 marks)**
	(d)	**Explain BCNF with example.**	**4M**
	Ans:	**BCNF:** Usually tables that are in Third Normal Form are already in Boyce Codd Normal Form. Boyce Codd Normal Form (BCNF) is considered a special condition of third Normal form. A table is in BCNF if every determinant is a candidate key. A table can be in 3NF but not in BCNF. This occurs when a non-key attribute is a determinant of a key attribute. OR A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds: • $\alpha \rightarrow \beta$ is a trivial functional dependency (that is, $\beta \subseteq \alpha$). • α is a super key for schema R. A database design is in BCNF if each member of the set of relation schemas that constitutes the design is in BCNF. **Example:** As an illustration, consider the following relation schemas and their respective • Customer-schema = (customer-name, customer-street, customer-city) customer-name → customer-street customer-city • Branch-schema = (branch-name, assets, branch-city) branch-name → assets branch-city • Loan-info-schema = (branch-name, customer-name, loan-number, amount) loan-number → amount branch-name One can entitle that Customer-schema is in BCNF. We note that a candidate key for the schema is customer-name. The only nontrivial functional dependencies that hold on	**(Description :2 marks, Example :2 marks)**

		Customer-schema have customer-name on the left side of the arrow. Since customer-name is a candidate key, functional dependencies with customer-name on the left side do not violate the definition of BCNF. Similarly, it can be shown easily that the relation schema Branch-schema is in BCNF.	
	(e)	**What is join? Explain different types of join.**	**4M**
	Ans:	**JOIN:** A SQL join is an instruction to combine data from two sets of data (i.e. two tables). A JOIN clause is used to combine rows from two or more tables, based on a related column between them. SQL Join types are as follows: **1) INNER JOIN or EQUI JOIN:** A join which is based on equalities is called equi join. In equi join comparison operator "=" is used to perform a Join. Syntax: SELECT tablename.column1_name,tablename.column1_name FROM table_name1,table_name2 where table_name1.column_name=table_name2.column_name; Example: Select stud_info.stud_name, stud_info.branch_code, branch_details.location From stud_info, branch_details Where Stud_info.branch_code=branch_details.branch_code; **2) SELF JOIN:** The SQL SELF JOIN is used to join a table to itself, as if the table were two tables, temporarily renaming at least one table in the SQL statement. Syntax: SELECT a.column_name, b.column_name FROM table1 a, table1 bWHERE a.common_filed = b.common_field; Example: Select x.stud_name, y.stud_name from stud_info x, stud_info y Where x.leader= y.stud_id; **3) LEFT OUTER JOIN:** A left outer join retains all of the rows of the "left" table, regardless of whether there is a row that matches on the "right" table. Syntax: Select column1name,column2name from table1name any_alias1 ,table2name any_alias2 on any_alias1.columnname(+) = any_alias2.columnname; **OR** Select column1name,column2name from table1name left outer join table2name on table1name.columnname= table2name.columnname;	**(Definition : 2 marks, Any 2 types with description: 1 mark each)**

Example:
select last_name, department_name from employees e, departments d on e.department_id(+) = d.department_id;

OR

select last_name, department_name from employees left outer join departments on employees.department_id = departments.department_id;

4) RIGHT OUTER JOIN:

A right outer join retains all of the rows of the "right" table, regardless of whether there is a row that matches on the "left" table.

Syntax:

Select column1name, column2name from table1name any_alias1, table2name any_alias2 on
any_alias1.columnname =any_alias2.columnname (+);

OR

Select column1name, column2name from table1name any_alias1 right outer join table2 name any_alias2 on any_alias1.columnname =any_alias2.columnname;

Example:

Select last_name,department_name from employees e, departments d on e.department_id = d.department_id(+);

OR

Select last_name, department_name from employees e right outer join departments d on e.department_id = d.department_id;

5) NON EQUI JOIN:

Non equi joins is used to return result from two or more tables where exact join is not possible.

Syntax:

Select aliasname.column1name, aliasname.column2name from tablename alias where <condition using range>;

For example:

In emp table and salgrade table. The salgrade table contains grade and their low salary and high salary. Suppose you want to find the grade of employees based on their salaries then you can use NON EQUI join.

Select e.empno, e.ename, e.sal, s.grade from emp e, salgrade s where e.sal between s.lowsal and s.hisal;

	(f)	**What are the types of triggers? Describe any two in detail.**	**4M**
	Ans:	Types of Triggers: 1. Row-level trigger 2. Statement-level trigger 3. Before-trigger 4. After-trigger **Row-level triggers for data-related activities** • Row-level triggers execute once for each row in a transaction. • Row-level triggers are the most common type of triggers; they are often used in data auditing applications. • Row-level trigger is identified by the FOR EACH ROW clause in the CREATE TRIGGER command. **Statement-level triggers for transaction-related activities** • Statement-level triggers execute once for each transaction. For example, if a single transaction inserted 500 rows into the Customer table, then a statement-level trigger on that table would only be executed once. • Statement-level triggers therefore are not often used for data-related activities; they are normally used to enforce additional security measures on the types of transactions that may be performed on a table. • Statement-level triggers are the default type of triggers created and are identified by omitting the FOR EACH ROW clause in the CREATE TRIGGER command. **Before Triggers:** • Before triggers are commonly used to check the validity of the data before action is performed. For instance one can use before trigger to prevent deletion of row if deletion should not be allowed in the given case. **After Triggers:** • After triggers are fired after the triggering action is completed for example if after trigger is associated with insert command then it is fired after the row is inserted into the table.	**(List: 2 marks, Description of 2 types: 1 mark each)**