Name - Lalit Devidas Patil

College Name - Jijabai College of Arts & Commerce Nashik

Subject Relational Database

Roll No - 106 Div - B

Std - FY BBA CA Sem-2

Assignment

Q.1 Attempt any Eight of the following (out of Ten):

a) What is difference between DBMS and RDBMS ?

→

RDBMS	DBMS
RDBMS is a relational database Manggement System	DBMS is a database Management System
RDBMS follows normalization Concept	DBMS does not follow the normalization
RDBMS has the major difference of Solving the queries easily as they are Stored in table format and use Many functional keys in Solving the querise.	DBMS is Mainly a Storage area and it does not employ any tables for Storing the data or does not use any Special Function keys or foreign keys for the retrieval of the data.
RDBMS Supports Client Server Architecture	DBMS does not Support Client / Server Architecture
RDBMS allows Simultaneous access of Users to the database.	Only One User Can access the database at a time in DBMS.
It treats data as tables Internally	It treats data as files Internally.

RDBMS	DBMS
RDBMS Stand for Relational Database Management System. This is the most Common form of DBMS. Invented by E.F Codd, the Only way to View the data is as a Set of tables. Because there Can be relation Ships between the tables.	DBMS Stands for Data Management System which is a general term for a Set of Software dedicated to Controlling the Storage of data.
It is used to establish the relationship Concept between two database Object, i.e tables.	In DBMS no relation Ship Concept.
It requires High Soft Were and hardwere requirement.	It requires low Soft Were and hard were requirement.
Examples: i) SQL Server ii) Oracle	Examples: i) Foxpro ii) IMS

b) What is timestamp ?

→ A timestamp is data type that Stores a unique Value that represents a Specific point in time. With each transaction T_i in the System, a fixed Value Called timestamp is associated, denoted by $TS(T_i)$. This timestamp is assigned by database System before T_i Starts execution.

If transaction T_i is assigned a timestamp $T_s(T_i)$ and a new transaction T_j enters the System, then $T_s(T_i) < T_s(T_j)$. There are two Simple Methods for implementing timestamp Scheme.

1) Use the Value of System Clock as the timestamp i.e transaction's time stamp is equal to the Value of System Clock, When the transaction enters the System.

2) Use a logical Counter that is incremented after a new timestamp has been assigned. i.e a transaction's timpstamp is equal to the Value of the Counter When the transaction enters the System.

Timestamp of transaction determines the Serializability Order.

If $T_s(T_i) < T_s(T_j)$ then the System Must ensure that the resultant Schedule is equivalent to Serial Schedule in Which T_i appears before T_j.

To implement this Scheme, two timestamp Values are associated With each data item Q.

W-Timestamp (Q): It denotes the largest timestamp of any transaction that executed Write (Q) Success fully.

R-Timestamp (Q) It denotes the largest timestamp of any transaction that executed read (Q) Success fully.

Whenever, a new read (Q) or write (Q) instruction is executed, these timestamps are updated.

C] Define Transaction.

→ In a Relational Database Management System, a transaction refers to logical unit of work that Consists of One or more database Operations, Such As inserting, updating, or deleting data. A transaction is a way to ensure the atomicity Consistency, isolation, and durability of database Operations, Commonly Known as the acid properties.

1] Atomicity: Atomicity property ensure that at the end of the transaction, either no Changes have Occurred to the database Or the database has been Changed in a Consistent manner. At the end of a transaction, the updates Made by the transaction Will be accessible to other transactions and processes Outside the transaction.

2] Consistency: Consistency property of transaction implies that if the database was in Consistent State before the Start of a transaction, then On termination of a transaction, the database Will also be in a Consistent State.

3] Isolation: Isolation property of transaction indicates that action performed by a transaction Will be hidden from Outside the transaction Untill the transaction terminates. Thus each transaction is

Unaware of Other Transactions executing Concurrently in the System.

4) Durability :- Durability property of a transaction ensures that Once a transaction Completes Successfully (commits), the Changes it has Made to the database persist even if there are System failures.
These four properties are Often Called ACID Atomicity, Consistency, Isolation and Durability properties of transaction.

---

d) What is Cursor ?

→ A Cursor is a temporary Work area used to Store the data retrieved from the database and Manipulate this data.

Declaring a Cursor - Cursors are defined Within a DECLARE Section of a PL/SQL block with DECLARE Command.

Syntax -

Cursor Cursor_name [(parameters)] [RETURN return type] Is SELECT query.

- The Cursor is defined by the CURSOR keyword followed by the Cursor identifier (cursor_name) and then the Select Statement.
- Parameters and return are optional part. When parameters are passed to Cursor it is Called as

Parameterized Cursor.

for - example -

```
Declare
    Cursor C-deptno Is Select ename,
Sal, deptno From EMP;
```

Opening Cursor - Cursor are opend With the Open Statement. This populates the Cursor With data.

Syntax -

```
Open Cursor-name [parameters];
```

for - example.

```
Declare
    Cursor C-deptno Is Select ename
Sal, deptno From Emp;
    Begin
        Open C-deptno;
    End;
```

Parameters is an Optional part. It is Used in parameterized Cursor.

---

e) Define Serializability.

→ For a transaction always a Serial Schedule result a Consistent database and not every Concurrent Schedule Can result in Consistent database. But a Concurrent Schedule result in a Consistent State if its result is equivalent to a

Serial Schedule results in a Consistent State. Such Concurrent Schedule is known as Serializable.

A Serializable Schedule is defined as:
Given (an interleaved execution) a Concurrent Schedule for n transactions: the following Conditions hold for each transaction in the Set.

1) All transactions are correct i.e if any One of the transactions is executed On a Consistent database, the resulting database is also Consistent.

2) Any Serial execution of the transactions is also Correct and preserves the Consistency of the data base.

1) Conflict Serializability - Consider that T_1 and T_2 are two transactions and S is a Schedule for T_1 and T_2. I_i and I_j are two instructions. If I_i and I_j refer to different data items, then I_i and I_j Can be executed in any Sequence.
But if I_i and I_j refer to Same data items then the Order of two instructions May matter Here I_i and I_j Can be a read or Write Operation Only Hence, following 3 Conditions are possible.

i) I_i = read (A)
I_j = read (A)

The Order of I_i and I_j does not matter because both are reading the data.

iii) T_i = read (A) T_j = Write (A)
T_j = Write (A) T_j = read (A)

Here, if read (A) is executed before Write (A) then it will read the Original Value of A otherwise it will read that Value of A which is written by Write (A). Hence, the Order of T_i and T_j matters.

iii) T_i and = Write (A) T_j = Write (A)

Here Order of T_i and T_j does not affect either T_i or T_j But the database is Changed, and it Makes difference for next read.

We Say that T_i and T_j Conflict if they are Operated by different transactions on the Same data item and at least One of them is write Operation. i.e Only in Case 1, T_i and T_j do not Conflict.

2) View Serializability - Consider two Schedules S and S', where Same Set of transactions participate in both Schedules. The Schedules S and S' are Said to be View equivalent if the following three Condition are Satisfied.

1) For each data item Q if transaction T_i reads the initial Value of Q in Schedule S, then transaction T_i in Schedule S' Must also read the initial Value of Q.

2] For each data item Q if transaction T_i executes read (Q) in Schedule S, and that value was produced by transaction T_j (if any), then transaction T_i in Schedule S', Must also read the value of Q that was produced by T_j transaction.

3] For each data item Q, the transaction that performs the final Write (Q) Operation in Schedule S must perform the final Write (Q) Operation in Schedule S'.

A Schedule S is View Serializable if it View equivalent to a Serial Schedule.

Example of View Serializable Schedule:

T_3	T_4	T_6
read (A)		
	Write (A)	
Write (A)		Write (A)

This Schedule is View equivalent to Serial Schedule
$< T_3, T_4, T_6 >$

f] What is deadlock ?

→ A Deadlock Occurs When two or more transactions are blocked, each Waiting for the Other to release a resource, Such as a database table or a row within a table, that is currently locked by the Other transaction. When a deadlock Occurs, the transactions involved Cannot proceed and the RDBMS is effectively Stuck in Loop. The only Way to resolve a deadlo

To to force one or more transactions to rollback, which means undoing the changes they have made so far and allowing the other transactions to proceed.

Deadlock can occur in any system that uses locking to manage concurrent access to shared resources, and they can be particularly problematic in database system because transactions can involve multiple resources and can be long-running. To minimize the occurrence of deadlocks, RDBMSs typically use algorithms that detect and resolve deadlocks automatically, or provide tools for administrators to manually intervene and resolve deadlocks.

Example- We have two transaction, T_1 and T_2 that are concurrently accessing the same two tables, Table A and table B in the following.

1) Transaction T_1 acquires an exclusive lock on row in Table A.

2) Transaction T_2 acquires an exclusive lock on row in Table B.

3) Transaction T_1 tries to acquire an exclusive lock on a row in Table B but has to wait for Transaction T_2 to release its lock on Table B.

4) Transaction T_2 tries to acquire an exclusive lock on a row in Table A but has to wait for

Transaction T1 to release its lock on Table A.

Now both transactions are stuck waiting for each other to release a lock on the other table, resulting in a deadlock. Neither transaction can proceed until the other one releases the lock, which creates a situation where both transactions are blocked and cannot make progress. To prevent deadlock, RDBMSs use techniques such as locking, timeouts, and deadlock detection algorithms to detect and resolve such situations.

9] What is ~~autmicity~~ atomicity?

→ Atomicity property ensures that at the end of the transaction, either no changes have occurred to the database or the database has been changed in a consistent manner. At the end of a transaction, the updates made by the transaction will be accessible to other transactions and processes outside the transaction.

Example - Transfer money from one bank account to another

1) Deduct the transfer amount from the source account.
2) Add the transfer amount to the destination account.

These two operations must be performed as a single transaction to ensure atomicity. If either of these operations fails, the entire transaction should be rolled back to its original state before the transaction began.

For instance, let's say you are transferring ₹ 100 from Account A to Account B. If the operation to deduct ₹ 100 from Account A succeeds, but Operation to add ₹ 100 to account B fails, the entire transaction must be rolled back, and Account A should remain unchanged.

Atomicity ensures that the data in the database remains Consistent even in event of failures or errors.

h) What is procedure ?

→ Procedures are simply a named PL/SQL block, that executes certain task. A procedure is completely portable among platforms in which Oracle is executed. Procedure is similar to a procedure in other programming language. A procedure has a header and a body. The header Consists of the name of the procedure and the parameters or Variables passed to the procedure. The body Consists of declaration Section, execution Section and exception Section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it named for repeated Usage.

Syntax -

Create [Or Replace] PROCEDURE procedure_name.
[(argument [in/out/in out] datatype [, argument [in/out/in out] datatype....])]
{IS/AS}
[Variable declaration] {PL/SQL block};

Example - Employees and One called "Department. The Employees table Contains the.

Employees (emp-id) (first-name, last-name, email, phone-number, hire-date, job-id, Salary, Commission-pct and Manager-id)

Department (dept-id, dept-name, Manager-id, Loc-id)

```
Create Procedure Sp-get-avg-Salary-by-dept
As
Begin
   Select d.dept-name, AVG(e.Salary) as avg-Salary
   From Employee
   Join Department d ON e.department-id = dept-id
   Group by d.dept-name.
END;

Exec- Sp-get-avg-Salary-by-dept.
```

ii) What is trigger?

→ A trigger is a pl/SQL block structure which is fired when a DML statements like Insert, Delete, Update, is executed on a database table. A trigger is triggered automatically when an associated DML statement is executed.

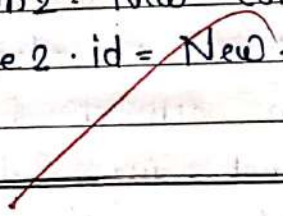
Syntax -

```
Create [Or Replace] Trigger trigger-name
{ Before | After | Instead Of}
{ Delete | Insert | Update [Of Column [, Column ..]]
```

[OR { Delete / Insert / update [OF Column
[, Column ...]]}
ON { Table / View }
For Each { Row / Statement }
[When (Condition)]
PL/SQL block.

Example -

```
Create Trigger update_another_table After Insert
ON table1.
For each row
Begin
    update table2
    Set Column1 = New.Column1,
        Column2 = New.Column2
    Where table2.id = New.id;
End;
```



ij What are Operators in plSQL ?

→ PLSQL Operators are either unary or binary. Binary Operators act on two Values. An example of binary Binary Operators is the addition Operators, which adds two numbers together. Unary Operators Only Operate On One Value.

1) Arithmetic Operators - Arithmetic Operators are used for Mathematical Computations.

Ex - +, -, /, *, **, +,

for example -

```
SQL >
SQL > Set Serveroutput On
SQL > Begin
        dbms_output.put_line (4*2); Multiplication
        dbms_output.put_line (8/4); division
        dbms_output.put_line (4+4); addition
        dbms_output.put_line (16-8); Subtraction.
      end;
      /
```

2) Comparison Operators - Comparison Operators are Used to Compare One Value Or expression to another. All Comparison Operators return a boolean result.

Ex - =, <>, != , ~= , <, >, >=, <=, Like, Between, In, Is Null.

for example :- greater than Use.

```
Declare
    v_emp_count Integer;
Begin
    Select Count(*) Into v_emp_count from employee
    where Salary > 50,000;
    DBMS_output.put_line ('There are' || v_emp_count
    || 'employees with Salary greater than 50,000.')
END;
```

3) Relational Operators :- Relational Operators are used to Compare two values and return a Boolean value of True or false based on the Comparison result.

1) Equal (=) - This Operator Checks ~~that~~ Whether two values are equal or not
E.g. 5=5 would return True.

2) Not equal to (!= or <>) - This Operator Checks Whether two values are not equal.
E.g. 5!=6 would return True.

3) Less than or equal to (<=) - This Operator Checks Whether the left Operand is less than or equal to the right Operand.
E.g. 5<=5 would return True.

4) Greater than (>) - This Operator Checks Whether the left Operand is greater than the right Operand.
E.g. 7>5 would return True.

5) Less than (<) - This Operator Checks Whether the left Operand is less than the right Operand
E.g. 3<5 would return True.

6) Greater than or equal to (>=) - This Operator Checks Whether the left Operand is greater than or equal to right Operand.
E.g. 7>=7 would return True.

Example:

```
Declare
    x Integer := 5
    y Integer := 7
    z Boolean;
Begin
    z := (x > y); -- Assigns False to z
    z := (x <= y); -- Assigns True to z
END;
```

4) Logical Operators - Logical operators are used to Combine Multiple Conditions to form a more Complex Condition that can be evaluated as true or false

E.g. - AND, OR, NOT

Example.

```
Declare
    x Number := 10;
    y Number := 5;
Begin
    If (x > 5 AND y < 10) OR NOT (x = 10) Then
    dbms_output.put_line ('The Condition is True');
Else.
    dbms_output.put_line ('The Condition is false');
END If
END;
```

5] String Operators - String Operators are used to manipulate character strings. These are like Operator and Concatenation (||) Operator.

1) Concatenation Operator (||) - The Concatenation Operator is used to join two or more strings together.

Example.

```
Declare
        first_name Varchar2(20) := 'lalit';
        last_name Varchar2(20) := 'patil';
        full_name Varchar2(40);
Begin
        full_name := first_name || ' ' || last_name;
        dbms_output.put_line(full_name);
END;
```

2) Like Operator (%) - The like Operator is used to match a string pattern.

Example.

```
Declare
        name Varchar2(20) := 'lalit patil';
Begin
        If name like 'L%' Then
        DBMS_Output.put_line('Start L');
END If
END;
```

Q.2 Attempt any four of the following.

a) Explain %type and %rowtype with an example.

→ Assign the same type to variable as that of the relation column declared in database. If there is any type mismatch, variable assignment and comparisons may not work the way you expect, so instead of hard coding the type of a variable, you should use the %type operator.

Example -

```
Declare
    My_name emp.ename%Type;
```

gives PL/SQL variable My_name whatever type was declared for the ename column in emp table.

The %type attribute provides the datatype of a variable or database column. This is particularly useful when declaring variables that will hold database values.

%rowtype - A variable can be declared with %rowtype that is equivalent to a row of a table i.e record with several fields. The result is a record type in which the fields have the same names and types as the attributes of the relation.

Example.

```
Declare
    Emp_rec emp%rowtype;
```

This Makes Variable emp-rec be a record with fields name and Salary, assuming that the relation has the Schema emp1 (name, Salary).

The initial Value of any Variable, regardless of its type, is Null. In pL-SQL, records are used to group data. A record Consists of a number of related fields in which data Values Can be Stored.

b) List and explain properties of transaction.

→

1) Atomicity - Atomicity property ensures that at the end of the transaction, either no changes have occurred to the database or the database has been Changed in a Consistent manner. At the end of a transaction, the updates Made by the transaction will be accessible to other transaction and processes outside the transaction.

2) Consistency - Consistency property of transaction implies that if the database was in Consistent State before the Start of a transaction, then On termination of a transaction, the database will also be in a Consistent State.

3) Isolation - Isolation property of transaction indicates that action performed by a transaction will be hidden from outside the transaction until the transaction terminates. Thus each transaction is unaware of other transactions executing Concurrently in the System.

4] Durability - Durability property of a transaction ensures that Once a transaction Completes Successfully, the Changes it has Made to the data base persist even if there are System failures.

C] What is deadlock ? Explain Methods to prevent deadlock.

→ A System is in deadlock State if there exists a Set of transactions Such that every transaction in the Set is waiting for another transaction in the Set. If $\{T_0, T_1, \ldots T_n\}$ is the Set of transaction Such that T_0 is waiting for a data item that is held by T_1 and T_1 is waiting for a data item that is held by T_2 ... and T_{n-1} is waiting for a data item that is held by T_0.
T_n is waiting for a data item that is held by T_0. Hence, none of the transactions Can Make progress in Such Situation. That is the System is in deadlock State.

There are 3 Methods for prevent deadlocks.

1] Deadlock prevention.
2] Time - out based Schemes.
3] Deadlock detection and deadlock recovery.

1] Deadlock Prevention - It ensures that Cyclic Waits Can be avoided by Ordering the request for locks or requiring all locks to be acquired together. It performs transaction rollbacks instead of waiti-

ng for a Lock, whenever the wait could poten tially result in a deadlock.

1) It is often hard to predict, before the transaction what data items need to be locked.
2) Data item utilization may be very low. Since many of the data items may be locked but unused for a long time.

11 Wait - Die - Scheme is based on non-preemptive technique. When a transaction T_i requests a lock on a data item currently held by T_j, T_i is allowed to wait only if it has a timestamp smaller than that of T_j. Otherwise T_i is rolled back (die)

Example. $TS(T_i) = 5$
$TS(T_j) = 10$
$TS(T_k) = 15$

If T_i requests a lock on a data item held by T_j then T_i will since $TS(T_i) < TS(T_j)$. If T_k requests a lock on a data item held by T_j then T_k will be rolled back since $TS(T_k) \geq TS(T_j)$.

2) Wound - wait - This scheme is based on preemptive technique. When a transaction T_i requests a lock on a data item, currently held by T_j, T_i is allowed to wait only if it has a timestamp larger than that of T_j. Otherwise T_j is rolled back (T_j is

Wounded by Ti).

Example - Con - The timestamps given in previous example for Ti, Tj and Tk transactions.

If Ti requests a Lock on data item held by Tj then Tj will be rolled back.

$$Ts(Ti) < Ts(Tj)$$

If Tk requests a Lock on a data item held by Tj then Tk will Since.

$$Ts(Tk) > Ts(Tj)$$

2] Time-out Based Schemes - In this approach deadLock handling is based on Lock time-outs. A transaction that has requested a lock waits for at most a Specified amount of time. If the Lock has not been granted within that time, the transaction is said to time Out and it rolls back itself and restarts.

If there was a deadLock, One or more transactions involved in deadLock will time-out and rollback allowing others to Complete their execution.

3] DeadLock Detection - This is One method for dealing with deadlock. It allows the System to enter a deadLock State, and then try to recover Using deadLock detection and dead-Lock recovery Scheme.

If the probability that System enters deadLock State is relatively Low, this Method is efficient.

An algorithm that examines the State of the System is executed periodically to determine Whether a deadLock has Occurred. If it has occurred then the System must attempt to recover from the dead lock.

4) DeadLock Recovery - The most Common Solution to recover from deadLock is to roll back One or more transactions to break the deadLock.

i) Selection of Victim - Determine which transaction to roll back. Those transactions that will incur the minimum Cost will be rolled back. The Cost of rollback One or more be decided by following factors.

i) How long the transaction has Computed and how Much longer the transaction will Compute before it Completes the assigned task?

ii) How Many data items it has used?

iii) How Many more data items it needs to Complete

iv) How Many transactions will be involved in the rollback?

ii) Roll back — One method to rollback a transaction is to abort transaction and restart it. The other more effective Method is to roll back the transaction Only as far as necessary to break the deadLock

But this require a additional information about all the running transactions.

iii) Starvation - It may happen that the same transaction is always selected as victim. This result in Starvation. The most common solution is to include the Number of rollback in the Cost factor.

Q) Explain RDBMS packages in detail.

→ Relational Database Management System which is a Software package that manages data stored in a relational database. RDBMS packages have become popular because they provide an efficient way of managing large amount of data in a highly organized and structured manner.

i) Database engine - The database engine is the core component of an RDBMS package. It manages the storage, retrieval, and updating of data in the database. It also enforces the rules and constraints that govern the relationship between the different tables in the database.

ii) Query language - RDBMS packages support a query language that allows users to retrieve data from the database. The most common query language used by RDBMS packages is SQL. SQL allows users to select, insert, update, and delete data from the database.

iiij Data Modeling tools- RDBMS packages typically provide data Modeling tools that allow Users to design and create the database Schema. These tools help Users to define the structure of the database and the relationships between the different tables.

iv) Security features- RDBMS packages provide a range of Security features to protect the data Stored in the database. These feasures include User authentication, access, Control, and encryption.

v) Backup & recovery- RDBMS packages provide backup and recovery to ensure that data is not Lost in the event of a System Failure. These tools allow Users to create backups of the database at rugular intervals and to restore the database to a previous State if necessary.

vij Performance Optimization- RDBMS packages provide a range of performance Optimization features to ensure that the database Operates efficiently. These features include indexing, caching and query Optimization.

Example- RDBMS packages include Oracle database, Micro Soft SQL Server, MySQL, and postgreSQL. These packages are used by Organizations of all Sizes to manage their data and Support a wide range of applications and business processes.

Q.3 Attempt any four of the following

a) What is function ? Explain with an example.

→ A function is a named PL/SQL Block which is similar to a procedure. The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

Syntax-

```
Create [Or Replace] Function function_name [para]
Return return_datatype.
{ is | AS}
Declaration - Section
Begin
Execution - Section
Return return_Variable;
Exception
exception Section
Return return_Variable.
END;
```

Example - "Employer_details_func"

```
Create or Replace Function employer_details_func
Return Varchar (20);
Is
emp_name Varchar (20);
Begin
Select first_name INto emp_name.
```

From emp-tbl where emp Id = '100';
Return emp-name;
END;
/

b) What is exception handling ? Explain predefined exception
→ PL/SQL provides a feature to handle the Exceptions which occur in a PL/SQL Block known as Exception Handling. Using Exception Handling, we can test the Code and avoid it from exiting suddenly. When an exception occurs a message which explains its cause is received.

PL/SQL Exception Message.

i) Type of expeption.
ii) An Error Code.
iii) A Message.

Syntax -

```
Declare
    Declaration Section
Begin
    Exception Section
Exception
When ex-name1 Then
  - Error handling Statements.
When ex-name2 Then
  - Error handling Statements
When Others Then
```

- Error handling Statements

END ;

General PL/SQL Statements can be used in the Exception Block.

Predefined Exception. The two most common erros originating from a Select Statement occur when it returns no rows when No-Data-Found or more than One row remember that is not allowed in PL/SQL Select Command. If no rows are selected from Select Statement then when No-Data-Found exception is used and for more then One row when Too-Many-rows exception is used.

Predefined Exceptions are.

i) No-Data-Found - When no rows are returned.
ii) Cursor-Already-open - When a cursor is opened in advance
iii) Storage-Error - If Memory is demaged.
iv) Program Error Internal problem in PL/SQL.
v) Zero-Divide - divide by zero
vi) Invalid-Cursor - If a cursor is not Open and you are trying
vii) Login-Denied - Invalid user name. or password.
viii) Invalid-Number - If you are inserting a String datatype for a Number datatype which is already. declared.
ix) Too Many-rows - If more row are returned by Select Statement.

Predefined exceptions are raised implicitly by the

runtime System. e.g. If you try to divide a numb er by zero, PL/SQL raises the predefined exception Zero_Divide automatically.

Syntax - No_Data_Found.

```
Exception
    When No_Data_Found Then
```

Example.

```
Declare
    V_name Varchar2(50);
Begin
    Select emp_name Into V_name From empl
    yees where emp_id = 106;

    DBMS_OutPUT.PUT_LINE ('Employee name
    is ' || V_name);
Exception
    When No_Data_Found Then
        DBMS_OUTPUT.PUT_LINE ('No Emplo
        ee found with emp_id 106');
END;
```

Q] Explain PLSQL block in detail.

→ The basic unit in PL/SQL is a block. All PL/SQ programs are made up of blocks, which can be nested within each other. Each block performs a logical action in the program. Each program consists of SQL and Statements which are from a PL/SQL.

A PL/SQL Block Consists of 3 Sections:

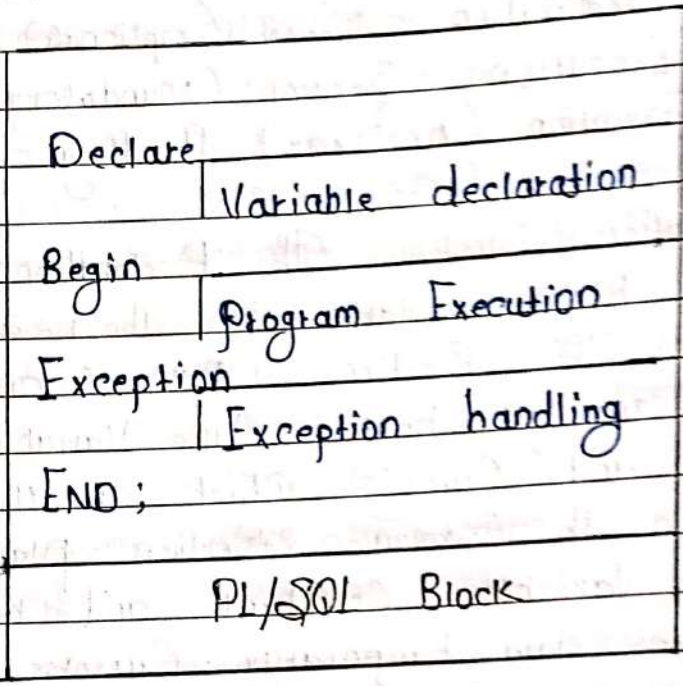
1) The Declaration Section (optional)
2) The Execution Section (Mandatory)
3) The Execution (or Error) Handling Section (optional)

1) Declaration Section - The Declaration Section of a PL/SQL Block Starts with the reserved keyword Declare. This Section is Optional and is used to declare any placeholders like Variables, Constants, records and Cursors, which are used to Manipulate data in the execution Section. placeholder May be any of Variables, Constants and records, which Stores data temporarily. Cursors are also declared in this Section.

2) Execution Section - The Execution Section of a PL/SQL Block Starts with the reserved keyword Begin and ends with END. This is Mandatory Section and is the Section where the program logic is written to perform any tasks. The programmatic Constructs like Loops, Conditional Statements and SQL Statement Form the part of execution Section.

3) Exception Section - The Exception Section of a PL/SQL Block Starts with the reserved keyword Exception. This Section is Optional. Any errors in the program Can be handled in this Section So that the PL/SQL Block terminates gracefully If the plSQL Block Contains exception that

Cannot be handled, the Block terminates abnormally with errors.



Example:

```
Declare
    V_roll No  Student.rollno %type ;
begin
  -- get roll number of the Student who
     joined most recently
     Select Max (rollno) into V_roll no;
  -- Insert a new row into payments table
     Insert into payment Values (V_roll No;
     Sysdate, 1000);
  -- Commit transaction
     Commit ;
end ;
/
```

e] What is log ? Explain log based recovery.

→ log is a structure used to store the database Modifi, cations. It is a sequence of log records and Main tains a record of all the Update activities in the database. Several types of log records Significant events during transaction processing.

1] Start of transaction :
denoted as : $\langle T_i \text{ Start} \rangle$

2] update log.
It describes a single database write and it is denoted as :

$$\langle T_j, X_j, V_1, V_2 \rangle$$

Where.

T_i - Transaction identifier.
X_j - Data item identifier.
V_1 - Old Value of X_j
V_2 - New Value of X_j after the write Operation.

3] Transaction Commits :
denoted as : $\langle T_i \text{ Commit} \rangle$

4] Transaction abort :
denoted as : $\langle T_i \text{ abort} \rangle$
log is Stored in Stable Storage.

log - based Recovery - It assumes that transactions are executed Serially . i.e Only One transaction is active at a time . It uses a Structure

Called log to Store the database Modifications. There are two techniques for using log to achieve the recovery and ensure atomicity in Case of Failures:

1) Deferred database Modification.
2) Immediate database Modification.

Deferred Modification technique ensures transaction atomicity by recording all database Modificatio in the log. but deferring (delaying) the executi n of all write Operations of transactions. Untill the transaction partially Commits.

The immediate update technique allows database Mod fications to be Output to the database While the transaction is Still in the active State. Data Modification Written by active transactio -s are Called uncommitted Modification. If a failure Occurs during execution, the System Mus Use the Old Value field of Log records.

Q.4 Attempt any four of the following

a) Consider following relational database.
Doctor (dno, dname, dcity)
Hospital (hno, hname, hcity)
Doct-Hosp (dno, hno)
Write a function to return Count of Number of hospitals located in Kolkatta City.

→ Create Or Replace function f-hcount return number as hcnt number;
Begin
Select Count(*) into hcnt from hospital Where hcity = 'Kolkatta';
return hcnt;
END f-hcount;
/

output-
SQL> Select f-count from dual;

f-Hcount

2

b) Consider the following transaction. Give two non-serial Schedules that the Serializable.

T_1	T_2
Read (A)	Read (B)
A = A - 1000	B = B + 100
Write (A)	Write (B)
Read (B)	Read (C)
B = B - 100	C = C + 100
Write (B)	Write (C)

→ i)

T_1	T_2
Read (A)	
A = A - 1000	
Write (A)	
	Read (B)
	B = B + 100
	Write (B)
Read (B)	
B = B - 100	
Write (B)	
	Read (C)
	C = C + 100
	Write (C)

ii)

T_1	T_2
Read (A)	
A = A - 1000	
	Read (B)
Write (A)	
	B = B + 100
	Write (B)
Read (B)	
B = B - 100	
	Read (C)
Write	
	C = C + 100
	Write (C)

C) Consider the following relational database.

Customer (cno, cname, city)

Account (ano, acc-type, balance, cno)

Define a trigger that restricts insertion or updation of account having balance less than 100.

→ Create or Replace trigger Customer-trig before insert or update On acc for each row.

```
begin
if updating then
    if :new.bal<100 then
        raise_application_error('-20010','Cannot Updat
        ! Balance Should be above 100');
    END if;
END if;
```

```
if inserting then
    if :New.bal < 100 then
    raise_application_error('-20010', 'Cannot
    Insert ! Balance Should be above 100');
  end if;
  End if;
End Customer_trig;
/
```

Output -

```
SQL > insert into acc Values (105, 'Savings', 50, 1);
Insert into acc Values (105, 'Savings', 50, 1);
                        *
Error at line 1:

ORA-20010: Cannot insert ! Balance Should be above 100
ORA-06512: at "Scott.Customer_Trig", Line 9
ORA-04088: error during execution of trigger 'Scott.
           Customer_Trig'

SQL > Insert into acc Values (105, 'Savings', 100, 1);
1 row Created.

SQL > update acc Set bal=50 Where ano=105;
update acc Set bal=50 Where ano=105
Error at line 1:
ORA-20010: Cannot update ! Balance Should by above
           100
```

ORA-06512: at "Scott.Customer_Trig", line 4
ORA-04088: error during execution of trigger 'Scott.Customer_Trig'

SQL > update acc set bal = 150 where ano = 105;

1 row updated.

d] Consider the following related database
Customer (cno, cname, city)
~~Account~~ (~~Cano~~, acc_type, ~~balance~~, Cno)
loan (lno, lamt, no_of_years, cno)

Write a procedure to display total loan amount from Delhi City.

→ Create or replace procedure p_disp_amt
as
t_loan number;
begin
Select Sum(lamt) into t_loan From loan
where cno in (select cno from Customer
where ccity = 'Delhi');
dbms_output.put_line('Total loan amount from Delhi ~~Pune~~ City is : ' || t_loan);
end p_disp_amt;
/

Output: SQL > execute p_disp_amt;
Total loan amount from Delhi city : 500000
PL/SQL procedure Successfully Completed.

Q.5. Write a short note on Any Two of the following.

a) Characteristics of RDBMS.

→

1) Data abstraction - Relational abstraction enhances program - data independence.

2) Self - describing data: Metadata describing structure of data stored together with data.

3) Concurrency - Supporting shared concurrent access.

4) Support for Multiple Views - External Users Can be provided with different views of the data.

5) Security - RDBMS offers different Levels of Security features such as Privacy / Confidentiality, Integrity, Availability and Accountability.

c) Concurrent Execution - The DBMS interleaves the actions of different transactions to improve performance, in terms of increased throughput to improved response times for short transactions, but not all interleaving should be allowed.
Ensuring transaction Isolation while permitting such Concurrent execution is difficult but is necessary for performance reasons.

ii) While One transaction. This waiting for a page to

be read in from disks, the Cpu Can process another transaction. This is because I/o activity Can be done in parallel with Cpu activity in a Computer. Overlapping I/o and Cpu activity reduces the amount of time disks and processors are idle, and Increases System throughput (the average number of transactions Completed in a given time)

iii) Interleaved execution of a short transaction with a long transaction usually allows the short transaction to Complete quickly. In Serial execution, a short transaction Could get Stuck behind a long transaction Leading to unpredictable delays in response time taken to Complete a transaction.

T_1	T_2
Read (A)	
Write (A)	
	Read (A)
	Write (B)
Read (C)	
Write (C)	

Problems of Concurrent Execution.

1) Lost update problem Occurs when Multiple transaction Select the same row and update the row based On the value Selected.

2] The uncommitted dependency problem / The Temporary update problem occurs when the second transaction selects a row which is updated by another transaction.

3] The non-Repeatable Read problem occurs when a second transaction is trying to access the same row several times and reads diffrent data each time.

4] The Incorrect Summary problem occurs when one transaction takes summary over the value of all the instances of a repeated data-item and second transactions updates few instances of that specific data-item. In that situation, the resulting summary does not reflect a correct result.

B.C.A / B.B.A. (CA) (III Sem-) Examination, 2017
301 Relational Database Management System (RDBMS)
2013 pattern

5. Solve the following

a) Consider the following transaction :

T_1	T_2
Read (z)	Read (x)
z = z * 10	Read (z)
Write (z)	x = x + z
Read (y)	Write (x)
y = y + z	
Write (y)	

Give two non Serial Schedules that are Serializable.

i)

T_1	T_2
Read (z)	
z = z * 10	
Write	
	Read (x)
Read (y)	
y = y + z	
Write (y)	
	Read (z)
	x = x + z
	Write (x)

ii)

T_1	T_2
Read (z)	
z = z * 10	
	Read (x)
Write (z)	
Read (y)	
y = y + z	
Write (y)	
	Read (z)
	x = x + z
	Write (x)

b) Consider the following transactions -

T_1	T_2
Read (A)	Read (B)
A = A * 5	B = B * 5
Write (A)	Write (B)
Read (B)	Read (A)
Read (C)	A = A * 10
B = B * 10	Write (A)
Write (B)	
C = C * 5	
Write (C)	

Give two non-Serial Schedules that are Serializable.

i)

T_1	T_2
Read (A)	
A = A * 5	
Write (A)	
	Read (B)
	B = B * 5
	Write (B)
Read (B)	
Read (C)	
B = B * 10	
Write (B)	
C = C * 5	
Write (C)	
	Read (A)
	A = A * 10
	Write (A)

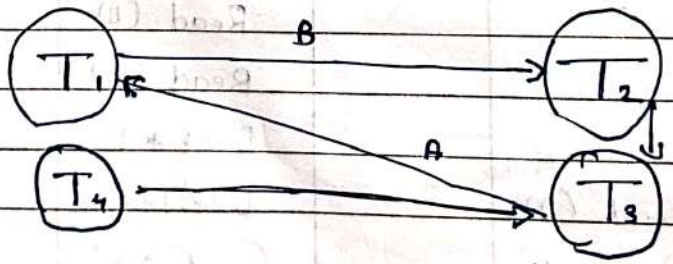
T_1	T_2
	Read (B)
	B = B * 5
	Write (B)
Read (A)	
A = A * 5	
Write (A)	
	Read (A)
	A = A * 10
	Write (A)
Read (B)	
Read (C)	
B = B * 10	
Write (B)	
C = C * 5	
Write (C)	

c) The following is the list of events in an inter-leaved execution of set T_1, T_2, T_3 and T_4 assuming 2pl. Is there a deadlock ? if yes. Which transaction are involved in deadlock ?

Time	Transaction	Code
t_1	T_1	Lock (A, x)
t_2	T_2	Lock (B, S)
t_3	T_3	Lock (A, S)
t_4	T_4	Lock (C, S)
t_5	T_1	Lock (B, S)
t_6	T_2	Lock (C, x)
t_7	T_3	Lock (D, x)
t_8	T_4	Lock (D, S)

Ans -

T_1	T_2	T_3	T_4
x(A)			
	S(B)		
		S(A)	
			S(C)
S(B)			
	x(C)		
		x(D)	
			S(D)



From the above wait-for graph, there is No Cycle.

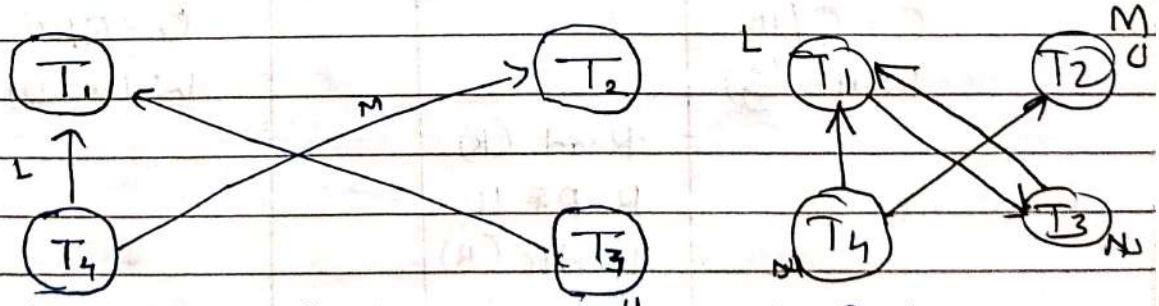
Page No.
Date

q) The following is the list of Events in an inter-leaved execution of set T_1, T_2, T_3 and T_4 assuming 2pl. Is there a deadlock? If yes, which transactions are Involved in deadlocks.

Time	Transaction	Code.
t_1	T_1	Lock (L, x)
t_2	T_2	Lock (M, x)
t_3	T_3	Lock (N, S).
t_4	T_4	Lock (N, S)
t_5	T_3	Lock (N, x)
t_6	T_3	Lock (L, S)
t_7	T_2	Lock (O, x)
t_8	T_4	Lock (M, S)

Ans.

T_1	T_2	T_3	T_4
x(L)			
	x(m)		
		S(N)	
			S(L)
x(N)			
		S(L)	
	x(O)		
			S(M)



From the above wait for graph, there No Cycle.

Apr-2018

BCA / BBA · (CA) (III) Semester Examination 2018

301 : Relational Database Management System (RDBMS)

2013 pattern

5 Attempt any four.

a) Consider the following transaction find out two non Serial Schedules that are Serializable.

T_1	T_2
Read (P)	Read (Q)
P = P * 10	Q = Q + 10
Write (P)	Write (Q)
Read (Q)	Read (R)
Q = Q/10	R = R * 10
Write (Q)	Write (R)

i)

T_1	T_2
Read (P)	
P = P * 10	
Write (P)	
	Read (Q)
	Q = Q + 10
	Write (Q)
Read (Q)	
Q = Q/10	
Write (Q)	
	Read (R)
	R = R * 10
	Write (R)

T_1	T_2
Read (P)	
P = P * 10	
Write (P)	
	Read (Q)
Read (Q)	
	Q = Q + 10
	Write (Q)
Q = Q/10	
Write (Q)	
	Read (R)
	R = R * 10
	Write (R)

b) Consider the following transaction. find out two non-Serial Schedules that are Serializable :

T_1	T_2
Read (x)	Read (x)
x = x + 1000	x = x - 1000
Write (x)	Write (x)
Read (y)	Read (y)
Read (z)	y = y - 2000
y = y + 2000	Write (y)
Write (y)	
z = z + 3000	
Write (z)	

i)

T_1	T_2
Read (x)	
x = x + 1000	
Write (x)	
	Read (x)
	x = x - 1000
	Write (x)
Read (y)	
Read (z)	
y = y + 2000	
Write (y)	
	Read (y)
	y = y - 2000
	Write (y)
Write (y)	
z = z + 3000	
Write (z)	

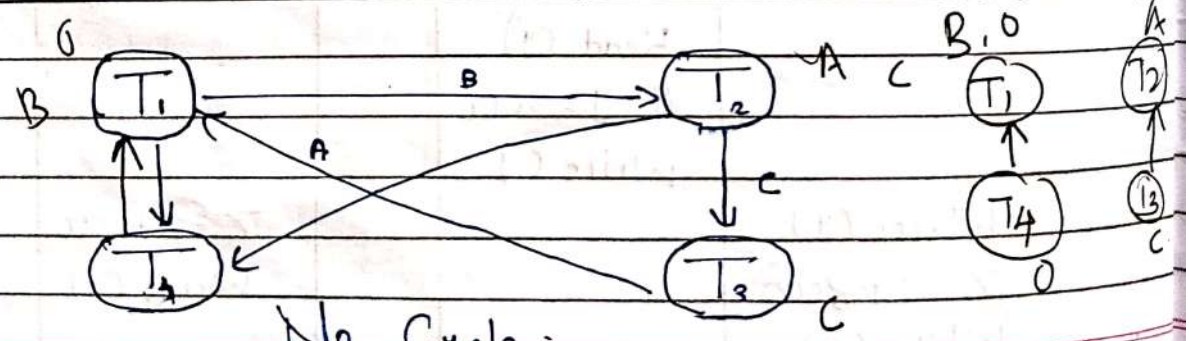
T_1	T_2
Read (x	
x = x + 1000	
Write (x)	
	Read (x)
Read (y)	
Read (z)	
	x = x - 1000
	Write (x)
y = y + 2000	
Write (y)	
	Read (y)
	y = y - 2000
	Write (y)
z = z + 3000	
Write (z)	

c] The following is the list representing the sequence of event in an interleved execution of set T_1, T_2, T_3 and T_4 assuming two-phase locking protocol. Is there a deadlock? If yes, which transaction are involved in deadlock.

Time	Transaction	Code.
t_1	T_1	Lock (B, S)
t_2	T_2	Lock (A, X)
t_3	T_3	Lock (C, S)
t_4	T_4	Lock (B, X)
t_5	T_1	Lock (D, S)
t_6	T_2	Lock (C, S)
t_7	T_3	Lock (A, S)
t_8	T_4	Lock (D, S)

Ans.

T_1	T_2	T_3	T_4
S(B)			
	X(A)		
		S(C)	
			X(B)
S(D)			
	S(C)		
		S(A)	
			S(D)



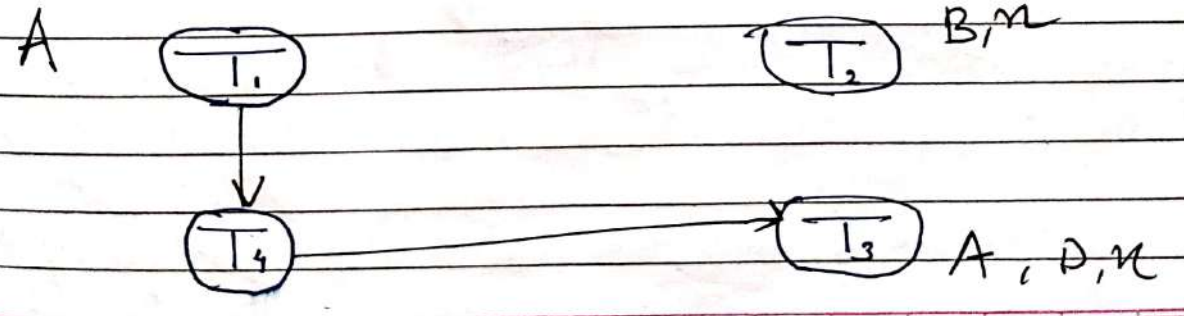
Page No.
Date

Q1 The following is the list representing the sequence of events in an interleaved execution of set T_1, T_2, T_3 and T_4 assuming two phase locking protocol. Is there a deadlock? If yes, which transactions are involved in deadlock.

Time	Transaction	Code.
t_1	T_1	Lock (A, X)
t_2	T_2	Lock (B, S)
t_3	T_3	Lock (A, S)
t_4	T_4	Lock (C, S)
t_5	T_1	Lock (C, X)
t_6	T_2	Lock (B, X)
t_7	T_3	Lock (D, X)
t_8	T_4	Lock (D, S)

Ans.

T_1	T_2	T_3	T_4
X(A)			
	S(B)		
		S(A)	
			S(C)
X(C)			
	X(B)		
		X(D)	
			S(D)



C

Page No.
Date

Oct-2018

B.C.A./B.B.A.(CA) (III Semester) Examination 2018
301 : Relational Database Management System
2013 Pattern

5. Attempt any Four:

a) Consider the following transaction. find out two non-Serial Schedules Serializable to Serial Schedule.

T_1	T_2	T_3
Read (A)	Read (C)	Read (B)
A = A + 500	Read (B)	Read (C)
Write (A)	B = B + C	B = B + 450
Read (B)	Write (B)	Write (B)
B = B - 500	Read (A)	C = C + B
Write (B)	A = N - C	Write (B)
	Write (A)	