પ્રશ્ન
પેટા પ્રશ્ન
ક્રમાંક

[ ] + [ ] + [ ] + [ ] + [ ] = [ ]

BCA Semester : ~~3~~

Roll Number : 30

Student Name :
Jat Kishanbhai Laxmanbhai

Subject : ~~Data Structure~~ ADBMS

Subject Code : ~~301~~ 302

પ્રશ્ન પેટા પ્રશ્ન ક્રમાંક

[ ] + [ ] + [ ] + [ ] + [ ] = [ ]

Q 1. Explain database concurrency in detail.

A → Concurrency in terms of database means allowing multiple users to access the data within a database at the same time.

- Concurrent execution of transaction is essential for good DBMS performance.

1> It improves throughput and resource utilization be because it is important to keep the CPU running by working several transactions concurrently.

2> It reduces waiting time in a serial processing.
- A short transaction may have to wait for a long transaction to complete,
- in the concurrent execution it reduces average response time and the average time for a transaction to be completed.

-
- If the concurrent access is not managed by DBMS, so that simultaneous operations may interfere with.

one another.

- Problems can occur when various transaction are working together. interleaved their operations and resulting an in inconsistent database.

# Concurrency Control

- Concurrency control is the process of managing simultaneous execution of transactions [~~Suc~~ such as queries, updates, inserts, detetes and so on] in a multiprocessing database system without having them interfere with one another.

*- Such type of problem is known as Lost update problem as update made by one transaction is lost here.

- This property of DBms allows many transactions to access the same database at the same time without interfecing with each other.

- The primary goal of concurrency is to ensure the atomicity of the execution of transactions in a multi-user database environment.

# Problem of Concurrency Control:

- When concurrent transactions are executed in an uncontrolled manner, several problems can occur.

The concurrency control has the following three main problems:

1. Lost Update Problem
2. Uncommitted Dependency problem [Dirty read]
3. ~~Incon~~ Inconsistent Analysis problem [Unrepeatable read or inconsistent retrievals]

## 1. Lost update problem

- When two transactions that access the same database items contain their operations in a way that makes the value of some database item incorrect. then the lost update problem occure.

- If two ~~tras~~ transactions $T_1$ and $T_2$ read a record and then update it, then the effect of updating of the first record will be overwritten by the second update.

Example:

| Transaction-x | Time | Transaction-y |
|---|---|---|
| — | t1 | — |
| Read A | t2 | — |
| — | ~~t3~~ t3 | Read A |
| Update A | t4 | — |
| — | t5 | Update A |
| — | t6 | — |

Here,

- At time t2, transaction-x reads A's value.
- At time t3, transaction-y reads A's value.
- At time t4, transaction-x writes A's value on the basis of the value seen at time t2.
- At time t5, transaction-y writes A's value on the basis of the value seen at time t3.
- So at time t5, the update of transaction-x is lost because transaction y overwrites it without looking at its current value.
- Such type of problem is known as Lost update problem as update made by one transaction is lost here.

## 2. Uncommitted Dependency Problem (Dirty Read)

- The dirty read occurs in the case when on transaction updates an item of the database, and then the transaction fails for some reason.
- The updated database item is accessed by another transaction before it is changed back to the original value.

Example:

| Transaction-x | Time | Transaction-y |
|---|---|---|
| — | t1 | — |
| — | t2 | Update A |
| Read A | t3 | — |
| — | t4 | Rollback |
| — | t5 | — |

- At time t2, transaction-y writes A's value.
- At time t3, transaction-x reads A's value.
- At time t4, transaction-y rollbacks, so, it changes A's value back to that of prior to t1.
- So transaction-x now contains a value which has never become part of the stable database.
- Such type of problem is known as Dirty Read Problem, as one transaction reads a dirty value which has not been committed.

## 3. Inconsistent Analysis Problem (Unrepeatable read or Inconsistent Retrievals Problem)

- Inconsistent Retrievals Problem is also known as unrepeatable read
- When a transaction calculates some summary function over a set of data while the other transaction are updating the data, then the Inconsistent Retrivals problem occurs.

Example: Suppose two transactions operate on three accounts.

| Acc1 | Acc2 | Acc3 |
|---|---|---|
| Balance=40 | Balance=50 | Balance=30 |

| Transaction-Tx | Time | Transaction-y |
|---|---|---|
| Read Acc1 Sum=40 | t1 | — |
| Read Acc2 Sum=90 | t2 | — |
| — | t3 | Read Acc3 30 |
| — | t4 | update Acc3 30→20 |
| — | t5 | Read Acc1 40 |
| — | t6 | UpdateAcc1 40-50 |
| — | t7 | Commit |
| Read Acc3 Sum=110 not 120 | t8 | |

- here Transaction-y is chang data base during Tx is read. that Problem to retrive result wrong.

પ્રશ્ન
ન પ્રશ્ન
ક્રમાંક

[ ] + [ ] + [ ] + [ ] + [ ] = [ ]

Q 2 Explain string and numeric functions in detail.

A → #Oracle Numeric ~~Funcit~~ Functions:

- ABS(n) – Return the absolute value of n.
  if n is 0 or a positive integer, return → n
  otherwise, n is multiplied by -1 → n*-1

  SELECT ABS(-15)"Absolute" from dual;
  Output: 15 as Absolute fild.

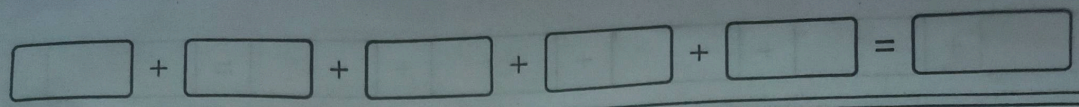
- POWER(m,n) – Returns the value of n raised to the power of m.

  SELECT POWER(3,2) from dual;
  Output: 9

- ROUND(n, m) – Returns a number n, rounded to the m specified decimal place.
  Unary (one argument) – drops the decimal portion of the input.

Ex SELECT ROUND(8.2) from dual; → 8
SELECT ROUND(8.7) from dual; → 9

Binary (two argument) – (the number to be rounded and a positive or negative integer that allows you to set the number of spaces at which the number is rounded.) The binary version



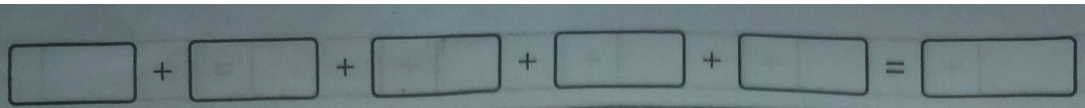
always returns a double.

SELECT ROUND(123.4567,3) from dual;
→ 123.457
SELECT ROUND(123.4,-3) from dual;
→ 0
SELECT ROUND(1234.56,-3) from dual;
→ 1000

- SQRT(n) - Returns the non-negative square root of n.
  SELECT SQRT(9) from dual; → 3
- GREATEST(exp1, exp2, ... expn)
  - Returns the greatest value in a list of expressions.

  SELECT GREATEST(392, 36, 81, 125) from dual; → 125
- LEAST(exp1, exp2, ... expn)
  - Returns the smallest value in a list of expressions.

  SELECT LEAST(30, 2, 26, 81, 225) from dual; → 2
- MOD(m,n) - Modulo. Returns the remainder of m divided by n.



SELECT MOD(10,3) from dual; → 1

- TRUNC(n, m) - Returns the number n truncated to m decimal places.
  - If m is 0, the result has no decimal point or fractional part.

Unary (one argument) - drops the decimal portion of the input.

Ex. SELECT TRUNC(3.14159265) from dual;
→ 3

Binary (two argument) - set the number of spaces at which the number is truncated.

Ex. SELECT TRUNC(3.14159265, 3) from dual;
→ 3.141

- CEIL(n) - Returns the smallest integer value not less than n (integer).

SELECT CEIL(24.8), CEIL(13.15) from dual;
→ 24.25, 14

- FLOOR(n) - Returns the largest integer value not greater than n.

SELECT FLOOR(24.8), FLOOR(13.15) from dual;
→ 24, 13

# String Functions:

- ASCII(character_expression)
  - Returns an ASCII code value of a character.
  SELECT ASCII('A') from dual;
  → 65

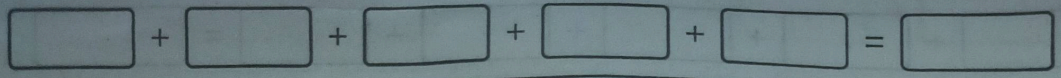
- CHR(numeric_expression) - Converts a numeric value to its corresponding ASCII character.

  SELECT CHR('65') from dual;
  → 'A'

- INITCAP(char) - Converts the first character in each word in a specified string to uppercase and the rest to lowercase.
  SELECT INITCAP('hi there') from dual;
  → 'Hi There'

- LENGTH(word) - Return the number of characters (or length) of a specified string.
  SELECT LENGTH('Palanpur') from dual;
  → 8

- LOWER(char) - Return a string with all characters converted to lowercase.
  SELECT LOWER('Abc') from dual; → 'abc'



- LPAD(char1, n, char2) - Return a string that is left-padded with the specified characters to a certain length.
  SELECT LPAD('ABC', 5, '*') from dual;
  → '**ABC'

- LTRIM(char, set) - Remove spaces or other specified characters in a set from the left end of a string
  SELECT LTRIM(' ABC') from dual;
  → 'ABC'

- SUBSTR(String, Start_Position, End_Position)
  - Extract a substring from a string
  - SELECT SUBSTR('Oracle substring', 1, 6);
    → 'Oracle'

- TRANSLATE(String, Exist_String, Replace_String)
  - Replace all occurrences of characters by other characters in a string.
    SELECT TRANSLATE('12345', '123134', 'bx' from dual; → 'bzx5'

- UPPER(char) - convert all characters in a specified string to uppercase.

  SELECT UPPER('Abc') from dual;
  → 'ABC'

Q. 3 Explain cursor in dbms in detail.

A → A Oracle creates a memory area, known as the cotext area, for ~~Pross~~ processing an SQL statement, which contains all the information needed for processing the statement.
- for examples, the number of rows processed, etc.

- A cursor is a ~~memory area, tt~~ pointer to this context area.
- PL/SQL controls the context area through a cursor.
- A curor holds the rows (one or more) returned by a SQL statement.
- The set of rows the cursor holds is refered to as the active set.

■ Use of Cursor:

- The major function of a cursor is to retrieve data

- cursors are used when the user needs to update records in a singleton fashi
- or in a row by row manner, in a database table.

- Oracle DBMS has another predefined area in the main memory set.

There are two types of cursors -

- Implicit cursors - If the Oracle engine opend a cursor for its internal processing it is known as an Implicit cursor.
  - It is created "automaticaly" for the user by Oracle when a ~~qua~~ query is executed and is simpler to code.

- Explicit cursors - A cursor can also be opend for processing data through a PL/SQL block.
  -on demand, such a user-defined cursor is known as Explicit cursors.

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

- Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an

implicit cursor is associated with this statement.

- for INSERT operations, the cursor holds the data that needs to be inserted
- for UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

- SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND and %ROWCOUNT

- Any SQL cursor attribute will be accessed as SQL%attribute_name as shown below in the example.

Example -

- We will be using the CUSTOMERS table we had created and used in the previous chapters.

Select * from customers;

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Khilan | 28 | Delhi | 1500.00 |
| 3 | Kaushik | 23 | Kota | 2000.00 |
| 4 | Chaitali | 25 | Mumbai | 6500.00 |
| 5 | Hardik | 27 | Bhopal | 8500.00 |

# Explicit Cursors

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

The syntax for creating an explicit cursor is -

```
CURSOR cursor_name IS
    select_statement;
```

- Working with an explicit cursor includes the following steps -
  - Declaring the cursor for initializing the memory.
  - Opening the cursor for allocating the memory.
  - Fetching the cursor for retrieving the data
  - Closing the cursor to release the allocated memory

Syntax:

```
DECLARE variables;
        records;
        create a cursor;
BEGIN
        OPEN cursor;
        FETCH cursor;
        process the records;
        CLOSE cursor;
END;
```

- Declaring the Cursor
  - Declaring the cursor defines the cursor with a name and the associated SELECT statement. For
  - For example:

```
(1) CURSOR c_customers IS
        SELECT id, name, address
        FROM customers;
```

- Opening the cursor
  - Opening the cursor allocates the memory for the cursor and makes it ready for fetching.

```
(2) OPEN c_customers;
```

- Fetching the cursor
  - Fetching the cursor involves accessing one row at a time.

```
(3) FETCH c_customers INTO c_id, c_name, c_addr;
```

- Closing the Cursor
  - Closing the cursor means releasing the allocated memory.

(E) CLOSE C_customers;

Complete example

```
DECLARE
    C_id customers.id%.type;
    C_name customers.name%.type;
    C_addr customers.addr%.type;
    CURSOR C_customers IS
        SELECT id, name, address FROM
        FROM customers;
BEGIN
    OPEN C_customer;
    LOOP
    FETCH C_customers into C_id, C_name, C_addr;
        EXIT WHEN C_customer%.notfound;
        dbms_output.put_line(c_id||''||C_name||''
            || C_addr);
    END Loop;
    CLOSE C_customers;
END;
```

Q4. Explain all database objects in detail.

A → # Database objects:-

- An object set
- A database object is any defined object in a database that is used to store or reference data.

- Anything which we make from create command is known as Database object.
- It can be used to hold and manipulate the data.

# List of Database objects:

- Tables: The Building Blocks of Data storage.

- At the core of any database system lies the concept of tables.
- Oracle's tables serve as the foundation for data storage, each comprising rows and colums

- A column defines a specific data type and constraints as fild. allowing for structured data representation.

- Tables define the schema, or structure, of the data, and their well-designed organization is essential for optimal performance.

- **Views: Abstraction and Controlled Access.**

  - views are virtual tables generated by querying existing tables.

  - They provide a layer of abstraction, shielding users from the complexities of the underlying table structures.

  - views are especially useful in scenarios where certain subsets of data need to be presented to different user groups.

  - By offering controlled access and tailored presentations, views enhance security and simplify data retrival.

- **Stored Procedures and Functions: Logic Encapsulation**

- Oracle's support for store - procedures and functions enables the encapsulation of business logic within the database.

- These units of code, - written in PL/SQL,
  - promote code reusability,
  - enhance security by centralizing logic.
  - and minimize network traffic

  by executing operations directly ~~with~~ within the database.

- **Triggers: Automated Event Handling**

  - Triggers are powerful mechanisms for automating actions based on specific events.

  - In Oracle, triggers are crefted using PL/SQL and are executed automatically in response to events sush as data manipulation. ~~They~~

  - They enforce business rules, enable auditing of changes, and maintain data consistency.

- by ensuring that predefined action are taken in response to defined events.

- Sequences : Generating Unique Identifiers.

  - Maintaining data integrity often involves the need for unique identifiers.
  - Oracle's sequences offer a systematic way to generate such ~~iden~~ identifiers, particularly for primary key fields.

  - Their role in preventing duplicate values and ensuring data accuracy in essential, especially in high-concurrency environments.

~~In the~~

In the realm of Oracle databases, data objects are the building blocks that enable efficient data storage, retrieval, and manipulation.

As we've ex, these database objects collectively contribute to the effectiveness and reliability of Oracle's database management capabilities.