

Give me all question answer

Altharva Bele <altharvale35@gmail.com>
To: Altharva Bele <altharvale35@gmail.com>, Prince Makari <princemakari007@gmail.com>

ADC Complete Answers (Q1 - Q45)

Q1: Analyze a situation where a trigger might cause a performance bottleneck. What factors contribute to this issue?

A trigger can slow down your database when it does too much work every time data changes.

- Situation:** A trigger on a highly-used `Orders` table executes a complex calculation or joins several large tables for every single new order. This constant, heavy processing slows down all new insertions.
- Factors Contributing to Bottleneck:**
 - Complex Logic:** The trigger code involves complicated joins, loops, or multiple DML operations (e.g., updating five other tables).
 - High Frequency:** The table it is attached to is modified very often (high volume of `INSERT` / `UPDATE`).
 - Locking:** The trigger may cause database locks on other tables, forcing different user transactions to wait for it to complete.

Q2: Analyze the advantages and disadvantages of using GOTO in PL/SQL programs.

The GOTO command forces the program to jump directly to another labeled line of code. It is generally discouraged.

Feature	Advantages (Pros)	Disadvantages (Cons)
Use Case	Allows you to quickly break out of deeply nested loops or complex IF structures to reach a single error-handling block.	Creates "Spaghetti Code", making the program flow chaotic, difficult to read, and hard to follow.
Maintenance	N/A	Hard to Debug: It is very difficult to trace the execution path, increasing the likelihood of errors during maintenance.
Structure	N/A	Breaks the rules of structured programming, leading to logical errors and instability.

Q3: Analyze the differences between a function and a procedure in PL/SQL.

Feature	Function	Procedure
Return	To calculate and return a single value (e.g., calculating tax).	To perform an action (e.g., inserting data, updating records, or printing a report).
Run Value	MUST return a value using the RETURN keyword.	Does not have to return a value. It returns data using OUT or IN OUT parameters.
SQL Usage	Can be called directly inside SQL statements (e.g., in a SELECT statement).	Cannot be called directly in SQL statements. Must be executed in a separate block (e.g., EXECUTE).

Q4: Analyze the differences between a function and a procedure in PL/SQL. In what situations would a function be preferred over a procedure?

(Combines Q3 and adds preference scenarios)

- Need a Single Result:** When the primary purpose is to compute and return one specific value (e.g., finding the average score, calculating a distance).
- Use within SQL:** When you need the logic to be part of a `SELECT`, `WHERE`, or `ORDER BY` clause within a standard SQL query.
- Code Reusability:** When you want to replace a complicated, repetitive calculation with a simple, clear function call in your application or SQL code.

Q5: Analyze the pros and cons of using triggers to enforce business rules compared to application-level validation.

Method	Advantages (Pros)	Disadvantages (Cons)
Trigger (Database-Level)	100% Guaranteed: The rule is enforced no matter how data is changed, ensuring perfect data integrity.	Can cause a performance hit on the database because it runs implicitly for every data change.
Application-Level	Better User Experience: Errors are caught instantly (e.g., in a web form) before reaching the database, providing fast feedback.	Easily Bypassed: Direct SQL updates or another application could bypass the rule and corrupt the data.
Application-Level	Better Performance: Saves the database server from doing the validation work.	Duplication of Logic: The same rule must be coded into every single application (web, mobile, desktop).

Q6: Analyze the role of cache and no cache in sequence.

A sequence automatically generates unique numbers.

Attribute	Role of CACHE	Role of NO CACHE
What it does	Stores a block of future sequence numbers in the database's memory for fast access.	Reads the next sequence number directly from disk (system files) every single time.
Performance	Very fast and ideal for high-volume applications.	Slower due to the need for I/O (disk access) for every number request.
Sequence Gaps	If the database crashes, any unused numbers in the cache are lost (skipped), creating gaps.	No numbers are lost on a crash, ensuring a gap-free, consecutive sequence.

Q7: Analyze the role of cycle and no cycle in sequence.

Attribute	Role of CYCLE	Role of NO CYCLE (Default)
What it does	Once the sequence reaches its maximum value, it starts over from its minimum value.	Once the sequence reaches its maximum value, it stops and returns an error if another number is requested.
Use Case	Used when you need unique IDs only for a short period and can safely reuse them later (e.g., batch numbers).	The standard, safer choice for creating permanent, unique primary keys that should never be repeated.

Q8: Compare and contrast %TYPE and %ROWTYPE attributes with examples. When would you use each?

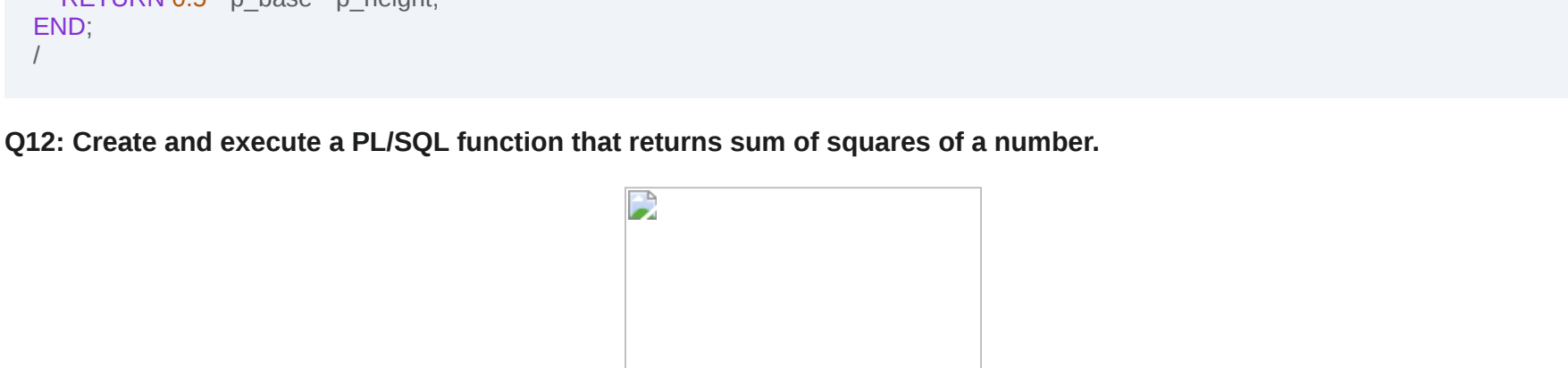
These make variable declarations flexible based on the database structure.

Attribute	Description	Use When...
%TYPE	Declares a variable to have the exact same data type and size as a single table column.	You need a variable to hold the value of a single column, like an employee's name or salary.
%ROWTYPE	Declares a record (like a structure) that can hold all columns (the entire row) from a table or cursor.	You need to process the entire data record of a table or query at once, handling all its columns.

Q9: Compare and contrast while loop and exit when in basic loop.

Loop Type	Key Feature	Control Mechanism
WHILE Loop	Condition at Start (Pre-test): The condition is checked before the loop code runs.	If the condition is initially FALSE, the code inside the loop never runs.
Basic Loop with EXIT WHEN	Runs at Least Once (Post-test): The loop code runs at least one time before the exit condition is checked.	The loop runs forever until the EXIT WHEN condition is met; the exit condition can be placed anywhere inside the loop.

Q10: Create and execute a PL/SQL function that returns area of a circle.

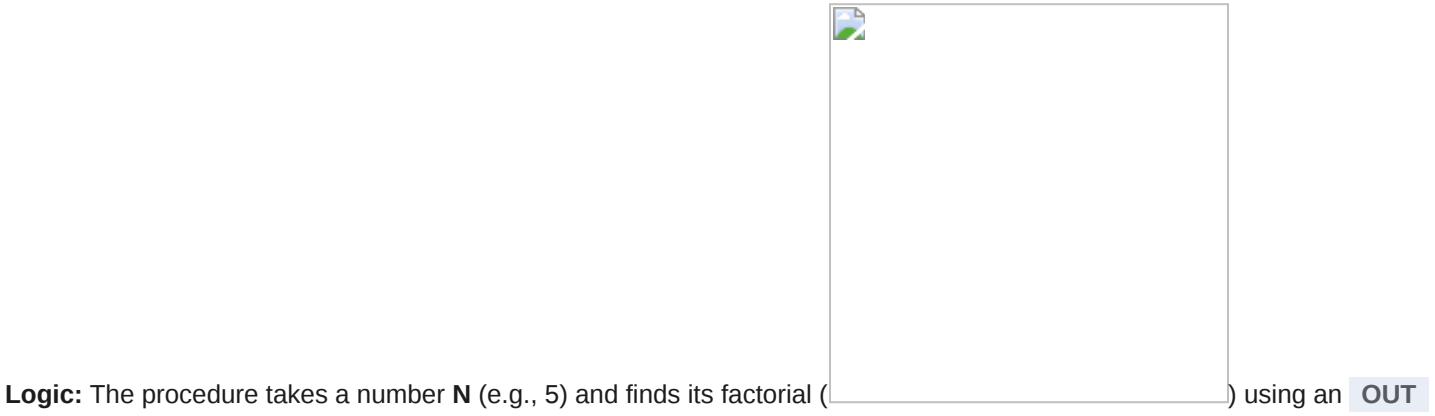


Logic: The function takes the radius () as an input and returns the calculated area ().

```
SQL

CREATE OR REPLACE FUNCTION get_circle_area (
    p_radius IN NUMBER
)
RETURN NUMBER
IS
    c_pi CONSTANT NUMBER := 3.14159;
BEGIN
    RETURN c_pi * p_radius * p_radius;
END;
```

Q11: Create and execute a PL/SQL function that returns area of a triangle.

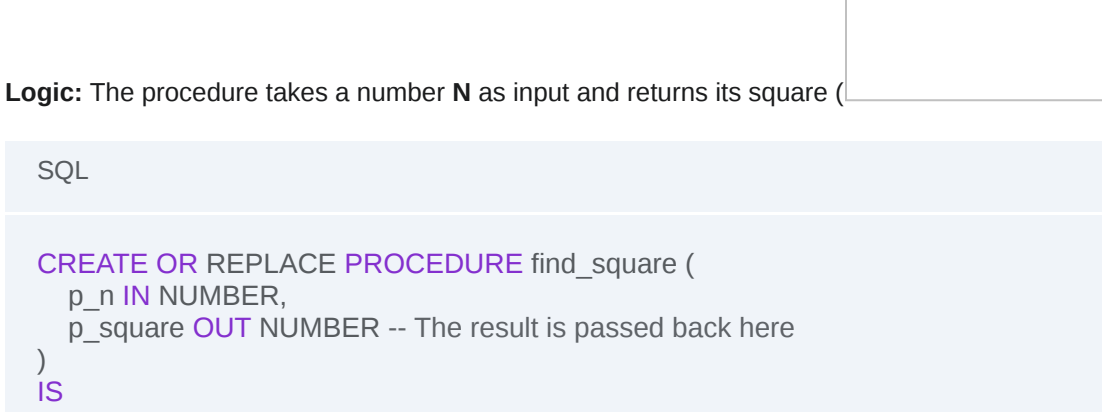


Logic: The function takes the base and height as inputs and returns the calculated area ().

```
SQL

CREATE OR REPLACE FUNCTION get_triangle_area (
    p_base IN NUMBER,
    p_height IN NUMBER
)
RETURN NUMBER
IS
BEGIN
    RETURN 0.5 * p_base * p_height;
END;
```

Q12: Create and execute a PL/SQL function that returns sum of squares of a number.



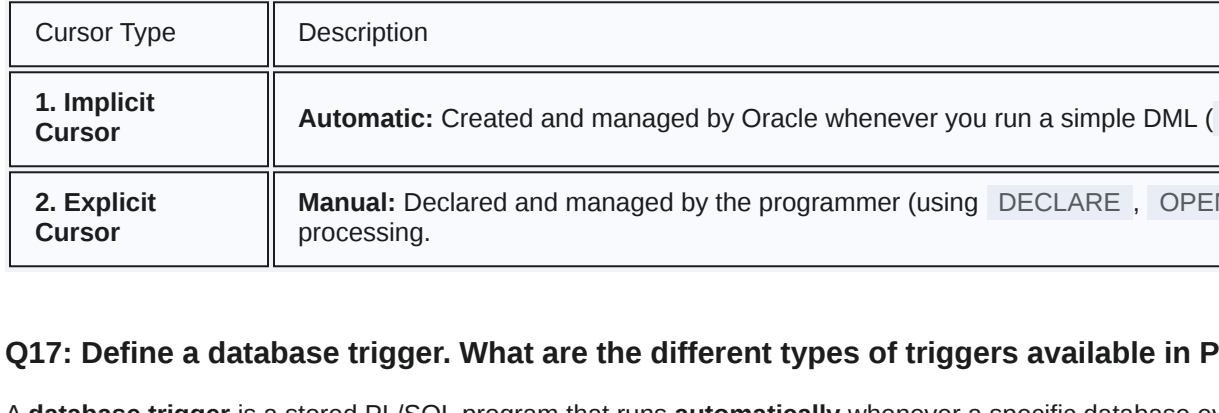
Assuming the sum of squares up to a given number N (e.g., N=5).

Logic: The function uses a FOR loop to iterate from 1 up to N and accumulates the squares.

```
SQL

CREATE OR REPLACE FUNCTION sum_of_squares (
    p_n IN NUMBER
)
RETURN NUMBER
IS
    v_total_sum NUMBER := 0;
BEGIN
    FOR i IN 1..p_n LOOP
        v_total_sum := v_total_sum + (i * i);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Sum of squares up to ' || p_n || ': ' || v_total_sum);
END;
```

Q13: Create and execute a PL/SQL procedure to find the factorial of a number.

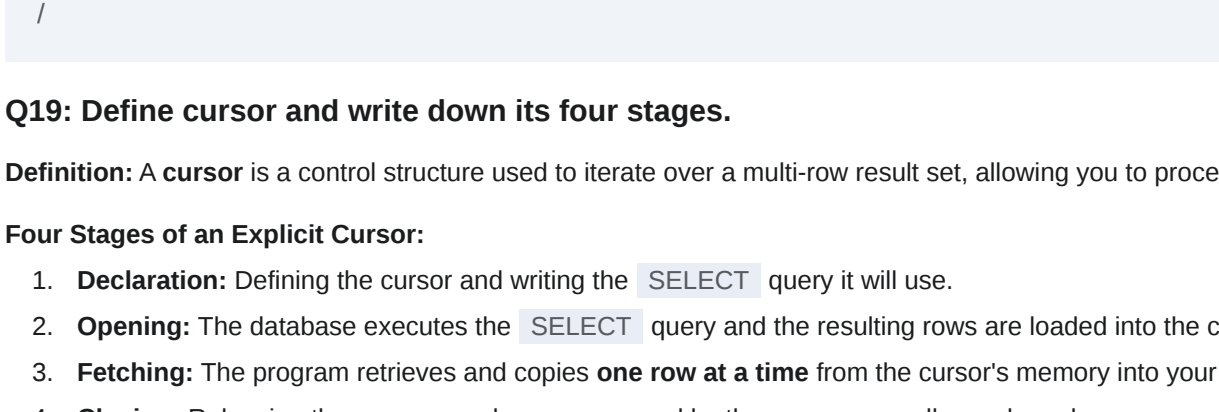


Logic: The procedure takes a number N (e.g., 5) and finds its factorial () using an OUT parameter for the result.

```
SQL

CREATE OR REPLACE PROCEDURE find_factorial (
    p_n IN NUMBER,
    p_result OUT NUMBER -- The result is passed back here
)
IS
BEGIN
    p_result := 1;
    FOR i IN 1..p_n LOOP
        p_result := p_result * i;
    END LOOP;
END;
```

Q14: Create and execute a PL/SQL procedure to find the square of a given number.



Logic: The procedure takes a number N as input and returns its square () using an OUT parameter.

```
SQL

CREATE OR REPLACE PROCEDURE find_square (
    p_n IN NUMBER,
    p_square OUT NUMBER -- The result is passed back here
)
IS
BEGIN
    p_square := p_n * p_n;
END;
```

Q15: Create and execute a PL/SQL procedure to find the sum of a first five numbers.

Assuming the sum of the first five natural numbers: 1, 2, 3, 4, 5 (Total = 15).

Logic: The procedure calculates the sum using a simple loop and displays the result using DBMS_OUTPUT.

```
SQL

CREATE OR REPLACE PROCEDURE sum_first_five
IS
    v_total NUMBER := 0;
BEGIN
    FOR i IN 1..5 LOOP
        v_total := v_total + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Sum of the first five numbers is: ' || v_total);
END;
```

Q16: Define a cursor in PL/SQL. List different types of cursors and briefly describe each.

A cursor is a temporary, private memory area that acts as a pointer to the set of rows returned by a SELECT statement, allowing a program to process the data one row at a time.

Cursor Type	Description
1. Implicit Cursor	Automatic: Created and managed by Oracle whenever you run a simple DML (INSERT, UPDATE, DELETE) or a single-row SELECT INTO. You only check its status.
2. Explicit Cursor	Manual: Declared and managed by the programmer (using DECLARE, OPEN, FETCH, and CLOSE). Used when a SELECT statement is expected to return multiple rows for processing.

Q17: Define a database trigger. What are the different types of triggers available in PL/SQL?

A database trigger is a stored PL/SQL program that runs automatically whenever a specific database event occurs (like a change to data or a system action).

Different Types of Triggers:

- DML Triggers (Data Manipulation):** Fire on INSERT, UPDATE, or DELETE commands on a table. Can be Row-Level (runs per row) or Statement-Level (runs once per command).
- Timing Triggers:** Based on when they fire: BEFORE (before the change, good for validation) or AFTER (after the change, good for auditing).
- DOL Triggers (Data Definition):** Fire on structure changes like CREATE, ALTER, or DROP of database objects.
- Database Event Triggers:** Fire on system events like database STARTUP, SHUTDOWN, or user LOGON / LOGOFF.

Q18: Demonstrate how to use IF-THEN-ELSE statement to compare two variables and print the larger one.

```
SQL

DECLARE
    v_a NUMBER := 50;
    v_b NUMBER := 60;
    v_max NUMBER;
BEGIN
    IF v_a > v_b THEN
        v_max := v_a;
    ELSE IF
        v_max := v_b;
    END IF;
    DBMS_OUTPUT.PUT_LINE('The maximum number is: ' || v_max);
END;
```

Q19: Define cursor and write down its four stages.

Definition: A cursor is a control structure used to iterate over a multi-row result set, allowing you to process each row individually.

Four Stages of an Explicit Cursor:

- Declaration:** Defining the cursor and writing the SELECT query it will use.
- Opening:** The database executes the SELECT query and the resulting rows are loaded into the cursor's memory area.
- Fetching:** The program retrieves and copies one row at a time from the cursor's memory into your local variables. This happens inside a loop.
- Closing:** Releasing the memory and resources used by the cursor once all rows have been processed.

Q20: Define sequence in PL/SQL. Explain the key attributes of a sequence.

Definition: A sequence is a database object used to automatically generate unique, consecutive numbers, typically used for creating Primary Key values.

Key Attributes:

- START WITH :** Sets the first number that will be generated.
- INCREMENT BY :** Specifies the step value by which the sequence number increases (or decreases).
- MAXVALUE / MINVALUE :** Sets the absolute highest or lowest number the sequence can generate.
- CYCLE / NOCYCLE :** Determines if the sequence should start over (CYCLE) after hitting its limit or stop (NOCYCLE).
- CACHE / NOCACHE :** Determines if a block of numbers should be stored in memory for faster access (CACHE).

Q21: Distinguish between actual and formal parameters in procedures/functions.

Feature	Formal Parameters (The Blueprint)	Actual Parameters (The Real Data)
Location	Declared in the procedure/function definition.	Passed to the procedure/function when it is called (executed).
Purpose	Placeholders that define the data type and direction (IN, OUT).	The real values or variables that are passed into or returned from the program.
Example	In PROCEDURE calc(P_AMT IN NUMBER), P_AMT is formal.	In EXEC calc(1000), 1000 is actual.

Q22: Differentiate between SQL and PL/SQL.

Feature	SQL (Structured Query Language)	PL/SQL (Procedural Language/SQL)
Purpose	Used to manage data: retrieving, inserting, updating, and deleting data.	Used to program logic: performing calculations, handling errors, and using loops.
Nature	Non-Procedural (or Declarative): You tell the DB what data you want.	Procedural: You tell the DB how to process the data step-by-step.
Flow Control	No loops or IF/THEN/ELSE statements.	Yes, includes IF, WHILE, and FOR loops for complex logic.

Q23: Differentiate between stored procedure and package.

Feature	Stored Procedure	Package
Functionality	A single program unit designed to perform one specific task or action.	A container that groups related procedures, functions, variables, and other objects.
Components	Contains only the procedure logic.	Contains two parts: Specification (the list of objects) and Body (the code for those objects).
Use for	Executing a specific, standalone task (e.g., calculating an employee's bonus).	Organizing code, managing security, and maintaining a session state (variables keep their value).

Q24: Differentiate between stored procedures and embedded SQL with examples.

Feature	Stored Procedure	Embedded SQL
Location	The code is compiled and stored inside the database server.	SQL code is placed inside the source code of a host language (like Java, C#, or Python).
Execution	Runs entirely on the database server.	SQL statements are sent one by one from the host language to the server.
Focus	Database-centric complex business logic (e.g., transaction processing).	Application-centric data access and manipulation.

Q25: Explain the benefits of using stored procedures in PL/SQL applications with suitable examples.

- Improved Performance (Speed):** Procedures are pre-compiled and stored. This avoids the need to parse and optimize the code every time, leading to faster execution.
- Reduced Network Traffic:** Instead of sending many SQL statements, the application sends one single command to execute the procedure, cutting down on network chatter.
- Enhanced Security:** Users can be granted permission to run the procedure without having direct access to the underlying tables, protecting the raw data.
- Code Reusability:** Logic is written once in the procedure and can be called by any application (web, mobile, other procedures), ensuring consistency.

Q26: Explain the structure of a PL/SQL block with example.

Every PL/SQL program is organized into a block with three main sections:

- DECLARE (Optional):** Used to define variables, constants, and cursors that are local to this block. (The Definition section).
 - BEGIN (Mandatory):** The executable section where the main logic, loops, and SQL statements are written. (The Action section).
 - EXCEPTION (Optional):** Used to specify actions to be taken when a runtime error occurs. (The Error Handling section).
- The block always ends with the mandatory END; keyword.

Q27: Explain the structure of PL/SQL block with a neat diagram.

(Since I cannot draw a diagram, I will describe the structure as a flow chart.)

- A PL/SQL block is a logical, structured unit:
- [DECLARE]** (Variables, Cursors)
 - BEGIN** (then Code Logic, SQL, Loops, IF-THEN)
 - [EXCEPTION]** (WHEN Error THEN Handle Error)
 - END;**

This structure ensures clean separation between variable definition, execution logic, and error handling.

Q28: Explain the use of for loop with syntax and example.

Use: The WHILE loop is used when you know exactly how many times you want the loop to execute (i.e., you have a definite starting and ending range). It automatically handles the loop counter.

Syntax:

```
SQL

FOR counter IN (REVERSE) start_value..end_value LOOP
    -- Statements to be executed repeatedly
END LOOP;
```

Example: A loop that counts down from 5 to 1.

```
SQL

FOR i IN REVERSE 1..5 LOOP
    DBMS_OUTPUT.PUT_LINE('Countdown: ' || i);
END LOOP;
```

Q29: Explain the use of while loop with syntax and example.

Use: The WHILE loop is used when you don't know the exact number of times the loop will run, but you have a condition that must remain TRUE. The condition is checked before each iteration (Pre-test).

Syntax:

```
SQL

WHILE condition LOOP
    -- Statements to be executed repeatedly
    -- Must include a statement that changes the condition!
END LOOP;
```

Example:

```
SQL

v_count := 1;
WHILE v_count <= 3 LOOP
    DBMS_OUTPUT.PUT_LINE('Iteration: ' || v_count);
    v_count := v_count + 1; -- Increment counter
END LOOP;
```

Q30: Explain the various attributes of cursor.

- Cursors attributes are status flags that give information about the most recently executed explicit or implicit cursor.
- %ISOPEN :** TRUE if the cursor is currently open; FALSE otherwise.
 - %FOUND :** TRUE if the last FETCH operation successfully returned a row.
 - %NOTFOUND :** TRUE if the last FETCH operation did not return a row (used to exit loops).
 - %ROWCOUNT :** The total number of rows that have been processed or affected by the cursor so far.

Q31: Identify the differences between BEFORE and AFTER triggers.

These are timing types of DML triggers that fire on a table.

Feature	BEFORE Trigger	AFTER Trigger
Fires When	Before the change is written to the database.	After the change is successfully written to the database.
Primary Use	Validation (checking data) and pre-processing (setting a default value).	Auditing (logging the final change) and cascading operations (updating summary tables).
Can Stop DML?	Yes. It can raise an error and prevent the operation from completing.	No. The original DML has already happened.

Q32: List any five advantages of using PL/SQL in Oracle databases.

- Performance:** It reduces network traffic because a single PL/SQL call executes many SQL statements internally.
- Error Handling:** It provides the EXCEPTION section, allowing developers to write reliable code that gracefully manages runtime errors.
- Procedural Capabilities:** It adds standard programming features (loops, conditionals, modularity) to the SQL environment.
- Integration with SQL:** All valid SQL statements (DML, DDL, TCL) can be executed directly and easily within a PL/SQL block.
- Security:** Using stored procedures allows you to control access to data by granting users permission to run the logic without direct table access.

Q33: What are the benefits of using a cursor in PL/SQL?

- Row-by-Row Processing:** Cursors allow you to loop through a set of results and process each row individually, which is essential for complex procedural logic.
- State Preservation:** A cursor maintains its position in the result set, allowing you to fetch one row, do other work, and then return to fetch the next row in order.
- Flexibility:** Cursors can be used for reading data and for making targeted, row-specific UPDATE or DELETE operations using the FOR UPDATE clause.

Q34: What are the benefits of using triggers in PL/SQL?

- Data Integrity Enforcement:** Triggers are the last line of defense for data, enforcing complex business rules regardless of the application used.
- Automatic Auditing and Logging:** They are perfect for automatically tracking changes (who, when, what was changed) to sensitive data.
- Centralized Logic:** Ensures that a rule is implemented consistently across all applications by storing the rule once in the database.
- Cascading Actions:** Allows the database to automatically perform related tasks, such as updating summary tables when detail data changes.

Q35: What are the features of PL/SQL?

- Block Structure:** Code is organized into logical, readable blocks (DECLARE, BEGIN, EXCEPTION).
- Procedural Capabilities:** Supports standard programming features like IF statements, WHILE / FOR loops, and reusable program units.
- Integration with SQL:** Allows seamless execution of all SQL statements directly within the code.
- Transaction Control:** Provides full control over transactions using COMMIT and ROLLBACK for multi-step data operations.
- Error Handling (Exceptions):** The EXCEPTION section allows for robust and graceful management of runtime errors.

Q36: What are the three main types of loops available in PL/SQL? Explain any one loop with syntax and example.

The three main types of loops are: Basic (Simple) Loop, WHILE Loop, and FOR Loop.

Explanation of FOR Loop:

- Definition:** The FOR loop is used when you know the exact starting and ending point of your iterations. It is simple because the counter variable and exit condition are managed automatically.
- Syntax:**

```
SQL

FOR counter IN start_value..end_value LOOP
    -- Statements to repeat
END LOOP;
```

- Example (Counting from 1 to 5):**

```
SQL

BEGIN
    FOR i IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE('Current number is: ' || i);
    END LOOP;
END;
```

Q37: What is a sequence in PL/SQL? List the key attributes of a sequence.

Definition: A sequence is a database object used to automatically generate unique numbers, often used to create primary key values.

Key Attributes:

- START WITH :** The first number generated.
- INCREMENT BY :** The step value for the sequence.
- MAXVALUE / MINVALUE :** The highest (or lowest) number the sequence can reach.
- CYCLE / NOCYCLE :** Whether the sequence restarts after reaching its limit.
- CACHE / NOCACHE :** Whether a block of numbers is pre-loaded into memory for speed.

Q38: What is a Trigger? Write the advantages of using trigger.

Definition: A trigger is a stored PL/SQL program that automatically runs (fires) when a specific database event occurs, such as an INSERT, UPDATE, or DELETE.

Advantages of Using Triggers:

- Guaranteed Data Integrity:** Enforces complex rules at the database level that cannot be bypassed by any application.
- Automatic Auditing:** Simplifies tracking changes to important data by automatically logging actions.
- Centralized Logic:** The business rule is stored in one place, ensuring consistency across all applications.
- Event-Driven Action:** Allows the database to immediately perform related tasks (e.g., updating stock) when an event occurs.

Q39: What is function? Write the advantages of function.

Definition: A function is a reusable PL/SQL program that must calculate and return a single value back to the calling program or SQL statement.

Advantages of Using Functions:

- Direct Use in SQL:** Functions can be called directly from standard SQL statements (SELECT, WHERE clauses).
- Code Reusability:** Complex calculations are written once and used many times across multiple applications.
- Modularity:** Functions break down large programs into smaller, testable, and manageable units.
- Focus:** Their clear purpose is to return one value, which makes them easy to integrate into expressions.

Q40: What is the syntax for creating a simple procedure in PL/SQL? Give an example.

Syntax for Creating a Function:

```
SQL

CREATE OR REPLACE FUNCTION function_name (
    parameter1 IN datatype
)
RETURN return_datatype
IS/AS
BEGIN
    -- Logic to calculate the return value
    RETURN calculated_value;
END function_name;
```

Example (Function to find the cube of a number):

```
SQL

CREATE OR REPLACE FUNCTION calculate_cube (
    p_number IN NUMBER
)
RETURN NUMBER
IS
BEGIN
    RETURN p_number * p_number * p_number;
END calculate_cube;
```

Q41: What is the syntax for creating a simple procedure in PL/SQL? Give an example.

Syntax for Creating a Procedure:

```
SQL

CREATE OR REPLACE PROCEDURE procedure_name (
    parameter1 [IN|OUT|IN OUT] datatype
)
IS/AS
BEGIN
    -- Logic to perform an action (e.g., INSERT, UPDATE, or printing)
END procedure_name;
```

Example (Procedure to print a greeting):

```
SQL

CREATE OR REPLACE PROCEDURE greet_user (
    p_name IN VARCHAR2
)
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello, ' || p_name || '!');
END greet_user;
```

Q42: Write a PL/SQL block using an IF-THEN-ELSE statement to check if a number is even or odd.

Logic: Uses the MOD function to check if the remainder when dividing by 2 is 0.

```
SQL

DECLARE
    v_number NUMBER := 17;
BEGIN
    IF MOD(v_number, 2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE('v_number || ' is an EVEN number.');
```

Q43: Write a PL/SQL block using an IF-THEN-ELSE statement to check if a number is positive, negative, or zero.

Logic: Uses IF, ELSEIF, and ELSE for the three possible conditions.

```
SQL

DECLARE
    v_test_number NUMBER := -10;
BEGIN
    IF v_test_number > 0 THEN
        DBMS_OUTPUT.PUT_LINE('The number is POSITIVE.');
```

Q44: Write the application of PL/SQL.

- Transaction Processing:** Used to write complex Stored Procedures that handle multi-step business transactions (e.g., money transfer).
- Data Validation:** Logic within Triggers enforces complex business rules, ensuring data is correct before it is saved.
- Reporting and Analytics:** PL/SQL Functions are used within SQL queries to calculate metrics and format data for reports.
- Web and Application Backends:** Procedures act as a secure, fast API for external applications to perform complex database operations.

Q45: Write the syntax for creating and dropping a sequence in PL/SQL.

1. Syntax for Creating a Sequence:

```
SQL

CREATE SEQUENCE sequence_name
INCREMENT BY number
START WITH number
NOCYCLE;
-- Other optional clauses like MAXVALUE, CACHE
```

2. Syntax for Dropping (Deleting) a Sequence:

```
SQL

DROP SEQUENCE sequence_name;
```

I hope this complete, consolidated list is helpful for your studies! Let me know if you need any of these topics explained further.

noname

2025-10-02 14:35:35