Cursors in SQL offer a powerful way to process data row-by-row, providing greater control and flexibility compared to traditional set-based operations.

However, managing complex cursor operations can become challenging.

This is where nested and parameterized cursors come in handy.

1. Nested Cursors

Nested cursors allow you to define and use one cursor within another cursor.

This is useful when you need to iterate through data in a hierarchical structure or perform multiple levels of data processing.

For example,

Consider a table of orders where each order includes a nested table of order items. A nested cursor can be utilized to iterate through all orders and, for each order, iterate through its associated order items.

Example syntax for a nested cursor

```
DECLARE
  order_cursor CURSOR FOR SELECT * FROM orders;
  item_cursor CURSOR FOR SELECT * FROM order_items WHERE order_id =
order_cursor%ROWTYPE.order_id;

BEGIN
  OPEN order_cursor;
  LOOP
```

```
    FETCH order_cursor INTO order_record;
    EXIT WHEN order_cursor%NOTFOUND;

    OPEN item_cursor FOR SELECT * FROM order_items WHERE order_id =
order_record.order_id;
    LOOP
      FETCH item_cursor INTO item_record;
      EXIT WHEN item_cursor%NOTFOUND;

      -- Process each order item
    END LOOP;
    CLOSE item_cursor;
  END LOOP;
  CLOSE order_cursor;
END;
```

---

2. Parameterized Cursors

Parameterized cursors enable the passing of dynamic values as parameters during cursor opening. This enhances flexibility and adaptability to various scenarios, eliminating the necessity to modify the cursor definition for each distinct query.

For example,

A parameterized cursor can retrieve employees based on a department ID passed as a parameter. This facilitates the effortless retrieval of employees from various departments without the need to modify the cursor code.

Example syntax for a parameterized cursor

```
DECLARE
  employee_cursor CURSOR (department_id NUMBER) IS
  SELECT * FROM employees WHERE department_id =
employee_cursor.department_id;

BEGIN
  OPEN employee_cursor(10); -- Open the cursor with department ID 10
  LOOP
    FETCH employee_cursor INTO employee_record;
    EXIT WHEN employee_cursor%NOTFOUND;

    -- Process each employee record
  END LOOP;
  CLOSE employee_cursor;

  OPEN employee_cursor(20); -- Open the cursor with department ID 20
and repeat the process
END;
```

Benefits of using nested and parameterized cursors

- Improved code modularity and organization: Complex logic can be broken down into smaller, reusable units.
- Enhanced flexibility and adaptability: Dynamically handle different data sets and scenarios.
- Reduced code duplication and maintenance: Avoid repetitive code for similar tasks.

- Greater control and efficiency: Process data row-by-row for specific needs.

Drawbacks

- Increased complexity: Nested and parameterized cursors can be more difficult to understand and debug.
- Performance considerations: Cursor operations can be less performant than set-based operations for large datasets.

Related Posts:

1. SQL Functions
2. History of DBMS
3. Introduction to DBMS
4. Introduction to Database
5. Advantages and Disadvantages of DBMS
6. SQL | DDL, DML, DCL Commands
7. Domain
8. Entity and Attribute
9. Relationship among entities
10. Attribute
11. Database Relation
12. DBMS Keys
13. Schema
14. Twelve rules of CODD
15. Normalization

16. Functional Dependency
17. Transaction processing concepts
18. Schedules
19. Serializability
20. OODBMS vs RDBMS
21. RDBMS
22. SQL Join
23. SQL Functions
24. Trigger
25. Oracle cursor
26. Introduction to Concurrency control
27. Net 11
28. NET 3
29. NET 2
30. GATE, AVG function and join DBMS | Prof. Jayesh Umre
31. GATE 2014 DBMS FIND Maximum number of Super keys | Prof. Jayesh Umre
32. GATE 2017 DBMS Query | Prof. Jayesh Umre
33. Data types
34. Entity
35. Check Constraint
36. Primary and Foreign key
37. SQL join
38. DDLDMLDCL
39. Database applications
40. Disadvantages of file system data management
41. RGPV DBMS Explain the concepts of generalization and aggregation with appropriate examples

42. RGPV solved Database approach vs Traditional file accessing approach
43. Find all employees who live in the city where the company for which they work is located
44. Concept of table spaces, segments, extents and block
45. Triggers: mutating errors, instead of triggers
46. Dedicated Server vs Multi-Threaded Server
47. Distributed database, database links, and snapshot
48. RDBMS Security
49. SQL queries for various join types
50. Oracle exception handling mechanism
51. Stored Procedures and Parameters