Lab on Applied Database Management Systems

Cursor Management and Triggers

SELF LEARNING MATERIAL

SEM - I (106)

MCA

UNIT-5 CURSOR MANAGEMENT AND TRIGGERS

Scan QR Code to Apply

TABLE OF CONTENTS

Learning Objectives

- To define the meaning of cursors and triggers
- To learn about the usage of cursors and triggers
- To understand the parts and types of triggers

5.1 Introduction

Cursors and triggers serve different purposes and are used in different scenarios within a database. Cursors allow you to iterate through the rows of a result set or a specific subset of data. This is useful when you need to perform operations on individual rows sequentially, such as performing calculations, updating values, or applying complex business logic. Cursors provide mechanisms to control the position of the cursor, move forward or backward through the rows, skip rows, or reposition the cursor based on specific conditions.

Triggers can be used to enforce business rules and data integrity constraints by validating the data before it is inserted, updated, or deleted from a table. This ensures that only valid and consistent data is stored in the database. Triggers allow you to implement complex business rules or calculations that need to be automatically executed when specific data manipulation events occur. This eliminates the need to manually perform these operations in application code, ensuring consistency and reducing redundancy.

NOTES

5.2 Explicit & Implicit Cursor

Explicit Cursors:

Explicit cursors are cursors that are explicitly declared and utilized by developers to retrieve and manipulate data from a result set in a controlled manner. Developers have direct control over explicit cursors and can define their behaviour. Explicit cursors offer developers precise control over traversing the result set and enable them to perform row-level processing and manipulation. They are commonly employed in procedural languages like PL/SQL (used in Oracle) or T-SQL (used in Microsoft SQL Server).

STUDY NOTE

With explicit cursors, developers have fine-grained control over traversing and processing the result set. They can fetch and manipulate rows individually, allowing for complex row-level operations.

The steps involved in using an explicit cursor generally include:

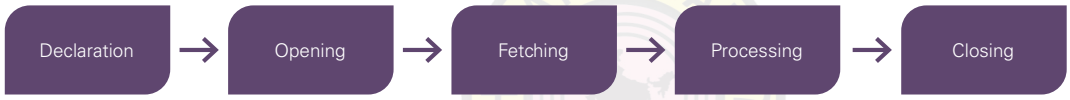

Fig 1: Steps involved in using Explicit Cursor

- **Declaration:** The cursor is defined and associated with a specific SQL query.
- **Opening:** The query is executed, and the result set is populated.
- **Fetching:** Rows are retrieved one at a time from the result set.
- **Processing:** Operations are performed on each fetched row individually.
- **Closing:** The resources associated with the cursor are released.

Implicit Cursors:

In contrast, implicit cursors are cursors that are created and managed automatically by the database system. They are utilized implicitly when executing SQL statements in a program, without the need for explicit declaration by the developer. When an SQL statement is executed without the use of an explicit cursor, the database system automatically generates an implicit cursor and associates it with the statement. The result set is accessed through this implicit cursor.

Implicit cursors are typically used for single-row queries or situations where developers do not require fine-grained control over the result set. They are well-suited for scenarios where only a single row is expected as the output, such as retrieving a specific value from a table or performing simple data manipulation operations.

Some definitions of cursors by different authors are given below:

"Cursors can be seen as a mechanism for processing query results one row at a time."

Tom Kyte

"Cursors are a powerful database programming technique that allows you to loop through a set of rows and perform operations on each row individually"

Richard Walsh

CHECK YOUR PROGRESS

1. Implicit cursors eliminate the need for _____ cursor declaration and management.
2. Explicit cursors are commonly used in _____ languages like PL/SQL or T-SQL.
3. Developers can pass _____ to explicit cursors, allowing for dynamic queries and result set filtering based on specific conditions or criteria.
4. With explicit cursors, developers have fine-grained control over traversing and processing the result set. They can _____ and _____ rows individually, allowing for complex row-level operations.

5.3 Declaring Cursor Variables

Declaring cursor variables is a feature available in database systems, allowing for dynamic manipulation of cursors.

To declare a cursor in database programming, the syntax typically involves specifying the name of the cursor and associating it with a select statement.

```
DECLARE Cursor_Name CURSOR FOR Select_Statement;
```

When using the DECLARE keyword, you define a cursor by providing a unique name for the cursor. Following the cursor name, you can specify the select statement that determines the result set associated with the cursor.

Cursor variables provide flexibility by enabling dynamic construction and execution of queries. They allow developers to assign SQL statements to the cursor variable at runtime, facilitating the manipulation of result sets. Cursor variables can also be passed between program units or stored procedures, offering a powerful means for further processing.

NOTES

Types of cursors:

Constrained and unconstrained cursor variables are two distinct types of cursor variables that exhibit different behaviours and characteristics.

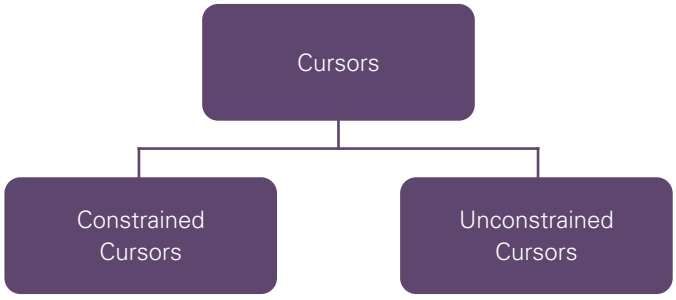

Fig 2: Types of Cursors

Constrained Cursor Variables:

Constrained cursor variables are cursor variables that are bound to a specific cursor type upon declaration. This type determines the structure and format of the result set that the cursor variable can accommodate.

- The cursor type can be explicitly specified using a cursor declaration or implicitly inferred from a strongly typed REF CURSOR type.
- Constrained cursor variables possess a fixed structure and can only hold result sets that align with their designated cursor type.
- They are advantageous in situations where the result set structure is known and predictable, ensuring type safety and minimizing potential errors during compilation or runtime.

Examples of constrained cursor variables include those based on strongly typed REF CURSOR types or cursors declared using explicit SELECT statements with specific column lists.

Unconstrained Cursor Variables:

Unconstrained cursor variables, alternatively known as weakly typed cursor variables or generic cursor variables, lack an association with any particular cursor type upon declaration. They offer greater flexibility but come with fewer compile-time checks.

- Unconstrained cursor variables have the ability to hold result sets with diverse structures or column lists.
- They are not restricted to a specific cursor type, thereby permitting the dynamic accommodation of result sets with varying structures.
- The actual structure of the result set is determined at runtime when the cursor variable is assigned or opened using a query.
- Unconstrained cursor variables provide increased flexibility but may introduce potential runtime errors if the result set structure deviates from the expected usage.
- They prove useful in scenarios where the result set structure is subject to change or remains unknown at the time of declaration.

CHECK YOUR PROGRESS

5. Cursor variables are commonly used when the _____ structure is unknown or can vary.
6. Cursor variables can only be declared within stored procedures or functions, not in standalone SQL statements. [True/False]
7. Unconstrained cursor variables can only be used in procedural languages and not in other programming paradigms. [True/False]
8. Constrained cursor variables ensure type safety and prevent potential runtime errors related to result set structure. [True/False]
9. You are working on a reporting module that generates various types of reports. The report structures are well-defined and consistent. Which type of cursor variable would be more appropriate for this scenario?

Activity

You are developing a batch processing job that involves performing sequential operations on a large dataset. The operations require iterating through the dataset, applying business rules, and updating records in the database. Facilitate a class discussion on which database construct would handle this efficiently. Justify your answers.

5.4 Working with Cursors

Opening a cursor:

When working with cursors in database programming, opening a cursor involves executing a query and associating the result set with the cursor. It is the second step in the cursor lifecycle.

Syntax:

```
OPEN Cursor_Name;
```

Fetching a cursor:

The third step in the cursor life cycle is to fetch the rows for performing the insertion, deletion, and updation operations on the currently active tuple in the cursor. Use a fetch statement, such as FETCH or FETCH INTO, to retrieve a row from the cursor's result set and store it in the declared variables.

Declaring and fetching is typically done in a loop, allowing you to fetch each row of data one at a time until the entire result set has been processed.

NOTES

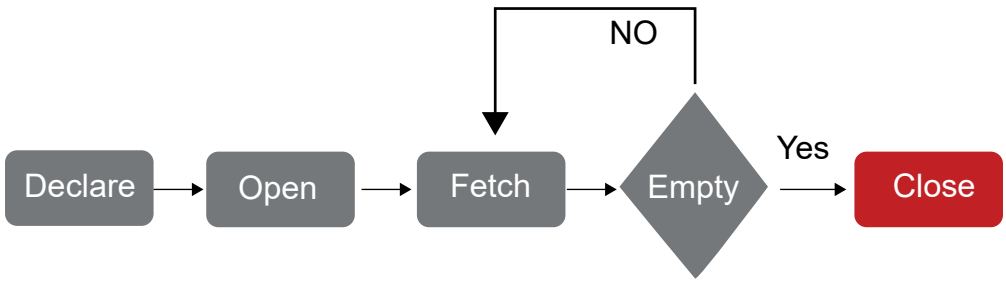

Fig 3: Fetch statement in Cursor lifecycle

Closing a cursor:

Closing a cursor is an essential step in the cursor process, typically performed when the cursor is no longer needed. To close a cursor in SQL, you can use the following query:

```
CLOSE Cursor_Name;
```

By executing this query, you signal the database system to release the resources associated with the cursor, freeing up memory and potentially releasing any locks held by the cursor.

All the steps together are used as:

```
DECLARE
  my_cursor SYS_REFCURSOR; -- Declaration of a cursor variable
BEGIN
  OPEN my_cursor FOR
    SELECT * FROM employees; -- Opening the cursor variable
with a query

  -- Perform operations on the cursor variable...

  CLOSE my_cursor; -- Closing the cursor variable
END;
```

In the given code snippet, a cursor variable named 'my_cursor' is declared using the 'SYS_REFCURSOR' type.

By utilizing the 'OPEN' statement, the cursor variable is associated with a specific query, which retrieves data from the 'employees' table.

Consequently, the cursor variable is populated with the result set. Various operations can be performed on the cursor variable within the code block.

Finally, the cursor variable is closed using the 'CLOSE' statement to free up any associated resources.

CHECK YOUR PROGRESS

10. Opening a cursor involves _____ a query and associating the result set with the cursor.

11. Closing a cursor is optional and does not affect the system resources or memory usage. [True/False]

12. Fetching a cursor into variables is done using a loop construct to fetch each row sequentially until all rows are processed. [True/False]

13. You are developing an inventory management system for a retail store. The system needs to generate a report of all products that have low stock levels. Why would you choose to use a cursor in this inventory management system scenario?

5.5 Cursor For Loops and Parametric Cursors

Cursor FOR loops are a construct used in programming languages like PL/SQL to iterate over the result set of a cursor. It simplifies the process of fetching and processing data from a cursor.

Declare a Cursor:

First, you declare a cursor variable and associate it with a SELECT statement that retrieves the desired data from a database table.

Cursor FOR Loop Syntax:

The Cursor FOR loop syntax is as follows:

```
FOR variable_name IN cursor_name LOOP
    -- Statements to process each row of data
END LOOP;
```

In this syntax, `variable_name` is a variable that holds the current row's data during each iteration of the loop. `cursor_name` refers to the previously declared cursor.

The Cursor FOR loop automatically opens the cursor, fetches rows from the result set, and assigns them to the `variable_name`. It then executes the statements within the loop for each row in the result set. Inside the loop, you can perform operations and calculations on the fetched data using the `variable_name`. Once all the rows have been processed, the Cursor FOR loop automatically closes the cursor, releasing resources and freeing up memory.

NOTES

Using a Cursor FOR loop simplifies the iteration process and eliminates the need for explicit cursor opening, fetching, and closing operations. It provides a concise and readable way to process data from a cursor in a database.

Parametric cursors are a type of cursor in database programming that allows for dynamic and flexible query execution. Unlike standard cursors that have a fixed query defined at declaration, parametric cursors enable you to change the query during runtime based on varying conditions or user input.

STUDY NOTE

Parametric cursors provide flexibility in constructing queries based on varying conditions, allowing for dynamic retrieval and processing of data. They are particularly useful in scenarios where the query criteria depend on user input or runtime conditions.

- **Declaration**: To declare a parametric cursor, you define it with placeholders or variables in the query portion of the cursor declaration. These placeholders will be replaced with specific values at runtime.
- **Binding Values**: Before opening the parametric cursor, you need to bind specific values to the placeholders or variables used in the cursor query. The values can come from user input, variables, or other sources.
- **Execution**: After binding the values, you can open and execute the parametric cursor. The cursor will use the bound values to generate a dynamic query based on the provided input.
- **Fetching Data**: Once the parametric cursor is open, you can fetch data from the result set as you would with a standard cursor. The query executed by the cursor will reflect the specific values bound to the placeholders.
- **Closing the Cursor**: After processing the result set, you can close the parametric cursor to release resources and free up memory.

CHECK YOUR PROGRESS

14. Cursor FOR loops can only be used with explicit cursors declared using the DECLARE CURSOR statement. [True/False]

15. Cursor FOR loops are only supported in specific database systems and may not be available in all programming languages or database platforms. [True/False]

16. Parametric cursors provide a convenient way to handle exceptions and errors that may occur during query execution. [True/False]

17. Parametric cursors can only be used in procedural programming languages like PL/SQL and not in _____ queries directly.

18. Parametric cursors are useful when you need to fetch data from _____ tables simultaneously.

Activity

You are building a reporting system that generates sales reports for different regions. The report format and content vary based on the selected region. To handle this requirement, you plan to use parametric cursors. How can parametric cursors be useful in this scenario? Create the code to depict the same. Compare your code with the other students in the class and analyse the different approaches used.

5.6 Triggers

Triggers in databases are special types of stored procedures that are automatically executed in response to specific events or actions occurring in the database. Triggers are associated with database tables and are triggered by operations such as INSERT, UPDATE, DELETE, or other database-related events.

- **Event-based Execution:** Triggers are executed automatically when a predefined event occurs in the database. These events can include data manipulation operations like INSERT, UPDATE, or DELETE on a table, or even database-level events such as database startup or shutdown.
- **Trigger Types:** Triggers can be classified into two main types: "Before Triggers" and "After Triggers." Before triggers are executed before the associated event takes place, allowing you to modify or validate data before it is modified in the database. After triggers are executed after the associated event completes, enabling you to perform additional actions or enforce constraints.
- **Business Logic Enforcement:** Triggers are commonly used to enforce business rules, data integrity constraints, or complex validation logic that cannot be easily achieved through other means such as constraints or default values. They allow you to implement custom logic to control or modify data changes.
- **Data Auditing and Logging:** Triggers can be used to track and log changes made to specific tables or columns in the database. This is particularly useful for maintaining an audit trail or capturing historical data changes.
- **Cascading Actions:** Triggers can initiate cascading actions, where a trigger on one table can cause operations on other related tables. For example, an UPDATE trigger on a parent table can trigger updates on child tables to maintain data consistency.

STUDY NOTE

Triggers can help with auditing and logging changes by capturing information about the modified data or triggering events, providing an extra layer of transparency and accountability.

NOTES

- **Performance Considerations:** While triggers provide powerful functionality, they should be used judiciously due to their potential impact on database performance. Poorly designed triggers with complex logic or inefficient queries can lead to performance degradation.
- **Trigger Creation and Management:** Triggers are typically defined and managed using Data Definition Language (DDL) statements provided by the database system. They can be created, altered, enabled, disabled, or dropped as needed.

Syntax of a trigger statement:

```
CREATE TRIGGER Trigger_Name
[ BEFORE | AFTER ] [ Insert | Update | Delete]
ON [Table_Name]
[ FOR EACH ROW | FOR EACH COLUMN ]
AS
Set of SQL Statement
```

The trigger syntax starts with the declaration of a unique trigger name after the CREATE TRIGGER keyword. Subsequently, the BEFORE or AFTER keyword is used to specify the timing of the trigger event. Following this, the table name is specified, indicating the table on which the trigger will be applied.

The trigger can be defined as either row-level or statement-level, determining whether it operates on individual rows or the entire result set of the triggering event. Finally, the trigger body consists of SQL statements that define the actions to be performed when the trigger event occurs.

"BEFORE" triggers are executed before the triggering event, "AFTER" triggers are executed after the triggering event, and "INSTEAD OF" triggers are executed instead of the triggering event for views.

The trigger body consists of the SQL statements or procedural code that defines the actions to be performed when the trigger is executed. It can include queries, data manipulation statements (e.g., INSERT, UPDATE, DELETE), conditional statements, or other database operations.

Within the trigger body, there are certain operations that are restricted or not allowed. These restricted parts include:

Fig 4: Restrictive operations on triggers

- **Modifying the Triggering Table:** It is generally advised to avoid modifying the table on which the trigger is defined within the trigger statement to prevent recursive triggers and potential infinite loops. It is recommended to perform any necessary modifications before or after the trigger execution.
- **Data Manipulation Language (DML) Statements:** Some database management systems (DBMSs) impose restrictions on specific data manipulation operations within a trigger. For instance, an UPDATE trigger may be limited from performing additional UPDATE statements on the same table. This restriction helps maintain data integrity and prevent conflicts or inconsistencies.
- **Transaction Control Statements:** Transaction control statements like COMMIT or ROLLBACK may have limited or no functionality within a trigger. This is because triggers are typically executed within the context of a larger transaction, and modifying the transaction state within a trigger can lead to unpredictable outcomes.
- **Recursive Trigger Execution:** Recursive triggers occur when a trigger action on a table triggers another trigger on the same table. Most DBMSs have mechanisms in place to control or prevent recursive triggers to avoid infinite loops and potential performance issues.

CHECK YOUR PROGRESS

19. Triggers can be nested, with one trigger invoking another trigger.

[True/False]

20. Triggers can be used to perform _____ checks and cancel the execution of the triggering event.

21. Triggers can _____ and _____ data in other tables not directly involved in the triggering event.

Activity

Search any schema freely available on online resources. Create a specific scenarios or requirements for which triggers will be helpful. Design and write appropriate trigger that satisfies the given scenario. Present your trigger implementation to the class. Facilitate a discussion by asking questions and clarify the doubts or confusion related to triggers.

NOTES

5.7 Types of Triggers

Triggers are classified into two categories:

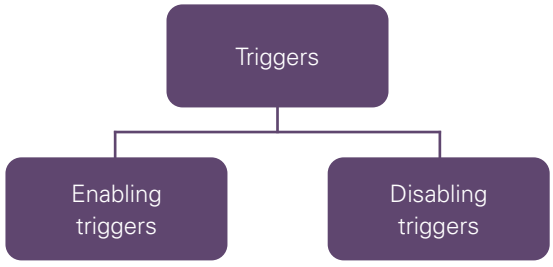

Fig 5: Types of Triggers

Enabling triggers:

Enabling triggers involves activating or allowing a trigger to be triggered by a specific event in a database management system (DBMS). Enabling a trigger makes it active and causes it to execute when the associated event occurs.

The process of enabling triggers may vary depending on the DBMS used, but it typically involves executing a specific command or modifying the trigger's status.

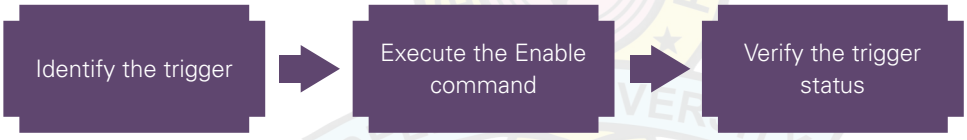

Fig 6: Process of enabling triggers

- **Identify the trigger:** Determine the name of the trigger that needs to be enabled.
- **Execute the Enable Command:** Utilize the appropriate syntax or command provided by your specific DBMS to enable the trigger. The exact command may differ based on the DBMS in use. For example, in Oracle, the ALTER TRIGGER statement with the ENABLE keyword is used.
- **Verify the Trigger Status:** After executing the enable command, verify that the trigger is now enabled. This can be done by checking the trigger's status in the DBMS's metadata or trigger management system.

Enabling triggers permits them to resume their functionality and respond to the designated event within the database.

Disabling triggers:

Disabling triggers involves suspending or deactivating the execution of a trigger in a database management system (DBMS). When a trigger is disabled, it will not be triggered or executed, even if the associated event occurs.

The process of disabling triggers may vary depending on the specific DBMS being used, but it typically involves executing a command or modifying the trigger's status.

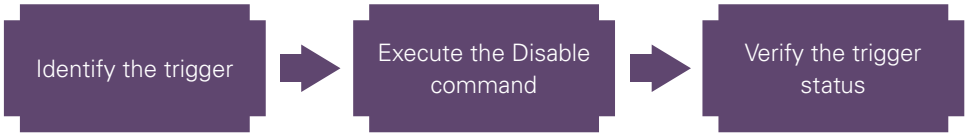

Fig 7: Process of disabling triggers

- **Identify the Trigger:** Determine the name of the trigger that needs to be disabled.
- **Execute the Disable Command:** Utilize the appropriate syntax or command provided by your specific DBMS to disable the trigger. The exact command may differ based on the DBMS in use. For instance, in Oracle, the ALTER TRIGGER statement with the DISABLE keyword is employed.
- **Verify the Trigger Status:** After executing the disable command, verify that the trigger is now disabled. This can be done by checking the trigger's status in the DBMS's metadata or trigger management system.

STUDY NOTE

Disabling triggers are beneficial in temporarily suspending trigger execution during maintenance activities, or troubleshooting processes.

CHECK YOUR PROGRESS

22. Enabling and disabling triggers can be performed on both row-level and _____ triggers.

23. Disabling a trigger temporarily _____ its execution until it is enabled again.

24. A disabled trigger can still be modified or altered. [True/False]

25. Enabling a trigger can only be done during the database creation process. [True/False]

26. A database application is experiencing performance issues during a large data import process. The application uses triggers on the target table to perform certain calculations and validations. To improve performance, the triggers need to be temporarily deactivated during the import process. How can the triggers be disabled in this scenario?

NOTES

NOTES

5.8 Summary

- Cursors are used for processing query results in a procedural manner.
- Cursors allow for fetching and manipulating individual rows of data from a result set.
- Cursor variables, whether constrained or unconstrained, provide flexibility in handling query results.
- Cursor For Loops are a convenient way to iterate over cursor results without the need for explicit cursor operations like opening, fetching, and closing.
- Triggers are event-driven procedures that automatically execute in response to specific events in a database, allowing for the enforcement of business rules, data integrity maintenance, and additional actions or validations.
- Triggers can have both before and after execution options.
- Enabling and disabling triggers provides control over their functionality.
- Understanding and effectively using triggers and cursors can greatly enhance the functionality and performance of database applications.
- Triggers and cursors provide mechanisms for automating tasks, maintaining data integrity, and processing query results. By leveraging these features, developers can build robust and efficient database solutions.

5.9 Case Study

Flipkart's Order Management System and Triggers

Flipkart is one of the leading e-commerce companies in India, known for its vast range of products and efficient order management system. In their order management system, Flipkart utilizes triggers in their database management system to ensure smooth order processing, inventory management, and customer satisfaction.

Flipkart's order management system handles a large volume of orders from customers across the country. To maintain efficient order fulfilment, it is crucial for Flipkart to have real-time inventory updates, track order status, and manage any changes or cancellations promptly. Triggers play a vital role in automating these processes and maintaining data consistency.

Trigger Implementation:

Inventory Management:

- **Trigger Scenario:** When a customer places an order, the inventory needs to be automatically updated to reflect the reduction in available stock.
- **Trigger Implementation:** Flipkart's database system is designed with a trigger that fires on each order placement. The trigger updates the inventory table by deducting the ordered quantity from the corresponding product's available stock. This ensures accurate inventory tracking and prevents overselling.

Order Status Updates:

- **Trigger Scenario:** When an order is marked as shipped, the customer should receive a notification with the updated status.
- **Trigger Implementation:** Flipkart employs a trigger that fires when an order's status is updated to "shipped." The trigger automatically sends an email or notification to the customer, providing them with the shipping details and confirming the shipment. This proactive communication keeps customers informed and enhances their overall experience.

Order Cancellation and Refunds:

- **Trigger Scenario:** When an order is cancelled, the inventory should be replenished, and the customer should be promptly refunded.
- **Trigger Implementation:** Flipkart's order management system includes triggers that activate when an order cancellation occurs. These triggers update the inventory by adding the cancelled quantity back to the available stock. Additionally, the trigger initiates the refund process by triggering the necessary financial transactions, ensuring that the customer receives their refund promptly.

Benefits and Impact:

- **Real-time inventory updates:** By using triggers, Flipkart ensures that the inventory is accurately reflected in the database, minimizing the chances of overselling or stockouts.
- **Enhanced customer experience:** Triggers facilitate timely order status updates and automatic notifications, keeping customers informed about their orders' progress.
- **Efficient order processing:** Triggers automate various processes such as inventory updates and order cancellations, reducing manual intervention and ensuring smooth order management.

Flipkart's effective utilization of triggers in their order management system showcases the significance of automation and data consistency in the e-commerce industry. By implementing triggers, Flipkart streamlines their processes, provides accurate inventory information, and enhances customer satisfaction. This case study demonstrates the practical application of triggers in a real-life scenario within an Indian company's database management system.

Questions:

1. How do triggers in Flipkart's order management system contribute to maintaining accurate inventory levels? Explain the specific trigger scenario and implementation that ensures inventory updates are reflected in real-time.
2. Analyse the impact of triggers on customer experience in Flipkart's order management system. How does the trigger-based order status update and notification process enhance customer satisfaction? Discuss the benefits of proactive communication and timely updates.
3. Evaluate the efficiency of triggers in Flipkart's order cancellation and refund process. How do triggers automate inventory replenishment and initiate the refund process when an order is cancelled? Discuss the benefits of trigger-based automation in handling order cancellations and ensuring prompt refunds.

5.10 Terminal Questions

SHORT ANSWER QUESTIONS

1. Write a trigger to count number of new tuples inserted using each insert statement.
2. Write the query to update a table by increasing the salary of each employee by 1500. After the update, the SQL%ROWCOUNT attribute is used to find out how many rows were affected by the operation.
3. Declare a cursor named "CustomerCursor" to fetch the customer ID and total purchase amount from the "Orders" table for the year 2022.

LONG ANSWER QUESTIONS

1. A large company wants to implement an employee performance evaluation system to track and assess the performance of its employees. The system should automatically calculate performance metrics based on various factors such as attendance, task completion, and customer feedback. Additionally, the system should generate reports for managers and HR personnel to review and make informed decisions. To address the requirements of the company, a combination of cursors and triggers can be utilized. Create a trigger named "AttendanceTrigger" that fires after an attendance record is inserted into the "Attendance" table.
2. In the above scenario, implement a cursor named "PerformanceEvaluationCursor" that retrieves employee records from the "Employees" table. The cursor can

calculate the overall performance score for each employee by aggregating the scores from the triggers fired on attendance, task completion, and customer feedback. Generate a performance report using the cursor, providing managers and HR personnel with an overview of each employee's performance metrics.

MCQ QUESTIONS

1. You are developing a payroll system for a company that needs to calculate monthly bonuses for employees based on their sales performance. The system should iterate through the sales records and calculate the bonus amount for each employee. Which of the following is the most appropriate method to achieve this?
   a) Using a cursor to iterate through the sales records and calculate bonuses for each employee.
   b) Using a scalar function to calculate bonuses for all employees in a single query.
   c) Using a temporary table to store the sales records and calculate bonuses using set-based operations.
   d) Using a subquery to calculate bonuses for each employee in the main payroll query.
2. You are working on a database migration project where you need to transfer data from an old database table to a new one. The data in the old table needs to be transformed and validated before being inserted into the new table. Which of the following is the most suitable approach for this task?
   a) Using a trigger to automatically transform and validate the data during the insertion process.
   b) Using a bulk insert statement to transfer the data from the old table to the new table.
   c) Using a cursor to fetch and transform the data row by row, and then insert into the new table.
   d) Using a stored procedure to handle the transformation and validation of data before inserting into the new table.
3. You are developing a system that needs to process a large volume of data in batches. Each batch requires complex calculations and updates based on the data. Which of the following options is the most appropriate for this scenario?
   a) Using a cursor to iterate through each batch and perform the necessary calculations and updates.
   b) Using a stored procedure with a loop to process each batch of data.
   c) Using a trigger to automatically process each batch of data as it is inserted into the database.
   d) Using a set-based operation to process the entire data at once without using cursors.

4. You are working on a database application for an e-commerce company. Whenever a new order is placed, the system needs to automatically update the inventory quantity for the corresponding products. Which of the following is the most appropriate approach to achieve this?
   a) Using a trigger to update the inventory quantity after each order insertion.
   b) Using a stored procedure to update the inventory quantity in a batch process.
   c) Using a scheduled job to update the inventory quantity at regular intervals.
   d) Using an application code to manually update the inventory quantity after each order.
5. You are developing a system to maintain an audit trail for a critical database table. Whenever a record is inserted, updated, or deleted in the table, a log entry needs to be added to the audit trail table. Which of the following is the most appropriate approach for achieving this requirement?
   a) Using triggers to automatically insert log entries into the audit trail table.
   b) Using stored procedures to handle the insertion of log entries.
   c) Using application code to manually insert log entries after each data modification.
   d) Using a scheduled job to periodically scan the table and insert log entries.
6. You are developing a database application for a hospital that needs to track and manage patient appointments. Whenever a new appointment is scheduled, the system should automatically update the doctor's schedule by deducting the allocated time slot. Which of the following is the most appropriate approach to achieve this?
   a) Using a trigger to update the doctor's schedule after each appointment insertion.
   b) Using a cursor to iterate through the appointments and update the doctor's schedule.
   c) Using a combination of a trigger and a cursor to ensure accurate updates in real-time.
   d) Using a stored procedure to handle the updating of the doctor's schedule.
7. In SQL, which command is used to enable/disable a database trigger?
   a) ALTER TABLE
   b) MODIFY TRIGGER
   c) ALTER DATABASE
   d) ALTER TRIGGER
8. Select the incorrect statement:
   a) We should use cursor in all cases
   b) A static cursor can move forward and backward direction
   c) A forward only cursor is the fastest cursor
   d) All of the mentioned

9. Programmers cannot control the _____ cursors and the information in it.
   a) Implicit
   b) Explicit
   c) Both a and b
   d) None

10. Which attribute returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows?
   a) %NOTFOUND
   b) %ISOPEN
   c) %ROWCOUNT
   d) %FOUND

11. You are developing a reporting system for a company that requires generating monthly sales reports for different regions. The reports should include sales data for a specific region based on user input. Which of the following is the most appropriate approach to achieve this?
   a) Using a parametric cursor to retrieve sales data for the specified region.
   b) Using a scalar function to calculate and retrieve sales data for the specified region.
   c) Using a temporary table to store sales data and filter it based on the specified region.
   d) Using a subquery to retrieve sales data for the specified region in the main report query.

12. You are developing a payroll system for a company. You need to process employee payroll data in batches of 100 employees at a time due to system constraints. Which type of cursor would be most suitable for this task?
   a) Unconstrained cursor
   b) Constrained cursor
   c) Forward-only cursor
   d) Dynamic cursor

13. You are developing a data analysis tool that requires iterating through a result set multiple times to perform complex calculations and aggregations. The tool needs the flexibility to move backward and forward within the result set. Which type of cursor would be most suitable for this requirement?
   a) Constrained cursor
   b) Unconstrained cursor
   c) Forward-only cursor
   d) Dynamic cursor

14. You are working on a customer management system that requires sending a welcome email to new customers upon their registration. Which part of the trigger contains the logic for sending the email?
   a) Trigger event
   b) Trigger body
   c) Trigger action
   d) Trigger condition

15. You are developing a database application that requires updating a column value whenever a specific condition is met. However, you want to ensure that the trigger cannot modify other columns in the same table. Which trigger restriction can you use to achieve this?
   a) INSERT restriction
   b) UPDATE restriction
   c) DELETE restriction
   d) REFERENTIAL constraint

NOTES

5.11 Answers

CHECK YOUR PROGRESS

1. Explicit
2. Procedural
3. Parameters
4. Fetch and manipulate
5. Result set
6. False
7. False
8. True
9. Constrained Cursor Variable
10. Executing
11. False
12. True
13. To be solved by student
14. False
15. True
16. False
17. SQL
18. Multiple
19. True
20. Validation
21. Access, modify
22. Statement-level
23. Suspends
24. True
25. False
26. To be solved by student

SHORT ANSWER QUESTIONS

1.
```
Declare count int
    Set count=0;
    delimiter $$
    CREATE TRIGGER Count_tupples
                AFTER INSERT ON employee
    FOR EACH ROW
    BEGIN
    SET count = count + 1;
    END; $$
    delimiter;
```

2.
```
DECLARE
       total_rows number;
    BEGIN
       UPDATE Emp
       SET Salary = Salary + 1500;

       total_rows := SQL%ROWCOUNT;

       dbms_output.put_line(total_rows || ' rows updated.');
    END;
```

3. DECLARE CustomerCursor CURSOR FOR

```
SELECT CustomerID, SUM(TotalAmount) AS TotalPurchaseA-
mount
FROM Orders
WHERE YEAR(OrderDate) = 2022
GROUP BY CustomerID;
OPEN CustomerCursor;
DECLARE @CustomerID INT;
DECLARE @TotalPurchaseAmount DECIMAL(10, 2);
FETCH NEXT FROM CustomerCursor INTO @CustomerID, @Total-
PurchaseAmount;
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Process the fetched data here
    PRINT 'CustomerID: ' + CAST(@CustomerID AS VARCHAR) +
', Total Purchase Amount: ' + CAST(@TotalPurchaseAmount
AS VARCHAR);
    FETCH NEXT FROM CustomerCursor INTO @CustomerID, @
TotalPurchaseAmount;
END;
CLOSE CustomerCursor;
DEALLOCATE CustomerCursor;
```

LONG ANSWER QUESTIONS

1. The "AttendanceTrigger" can be implemented as follows:

```
CREATE TRIGGER AttendanceTrigger
AFTER INSERT ON Attendance
FOR EACH ROW
BEGIN
    -- Perform necessary calculations and updates based
on the attendance record
    -- Update performance metrics or any other relevant
data
    -- Generate reports or trigger further actions if
needed
END;
```

The trigger will be executed after an attendance record is inserted into the "Attendance" table. Within the trigger body, you can perform calculations and updates based on the attendance record, such as updating performance metrics or any other relevant data. Additionally, you can generate reports or trigger further actions as required by the system. This trigger allows for the automatic calculation and tracking of employee performance based on their attendance records.

NOTES

2.
```
-- Create the PerformanceEvaluationCursor
DECLARE @EmployeeID INT, @OverallPerformance DECIMAL(10,
2);
DECLARE PerformanceEvaluationCursor CURSOR FOR
SELECT EmployeeID
FROM Employees;
OPEN PerformanceEvaluationCursor;
FETCH NEXT FROM PerformanceEvaluationCursor INTO @Employ-
eeID;
-- Declare variables to store aggregated scores
DECLARE @AttendanceScore DECIMAL(10, 2), @TaskCompletion-
Score DECIMAL(10, 2), @FeedbackScore DECIMAL(10, 2);
-- Declare variables for performance report
DECLARE @EmployeeName VARCHAR(50), @PerformanceReport
NVARCHAR(MAX) = '';
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Reset aggregated scores for each employee
    SET @AttendanceScore = 0;
    SET @TaskCompletionScore = 0;
    SET @FeedbackScore = 0;
    -- Calculate attendance score
    -- Implement the AttendanceTrigger logic here to cal-
culate and update the @AttendanceScore variable
    -- Calculate task completion score
    -- Implement the TaskCompletionTrigger logic here to
calculate and update the @TaskCompletionScore variable
    -- Calculate feedback score
    -- Implement the CustomerFeedbackTrigger logic here
to calculate and update the @FeedbackScore variable
    -- Calculate overall performance score
    SET @OverallPerformance = (@AttendanceScore + @Task-
CompletionScore + @FeedbackScore) / 3;
    -- Retrieve employee name
    SELECT @EmployeeName = EmployeeName
    FROM Employees
    WHERE EmployeeID = @EmployeeID;
    -- Build the performance report
    SET @PerformanceReport = @PerformanceReport + 'Em-
ployee Name: ' + @EmployeeName + CHAR(13) +
        'Overall Performance Score: ' + CAST(@OverallPer-
formance AS VARCHAR(10)) + CHAR(13) + CHAR(13);
    FETCH NEXT FROM PerformanceEvaluationCursor INTO @
EmployeeID;
END;
```

```
CLOSE PerformanceEvaluationCursor;
DEALLOCATE PerformanceEvaluationCursor;
-- Print or return the performance report
PRINT @PerformanceReport;
```

MCQS ANSWERS

1. a) Using a cursor to iterate through the sales records and calculate bonuses for each employee.

2. c) Using a cursor to fetch and transform the data row by row, and then insert into the new table.

3. b) Using a stored procedure with a loop to process each batch of data.

4. a) Using a trigger to update the inventory quantity after each order insertion.

5. a) Using triggers to automatically insert log entries into the audit trail table.

6. c) Using a combination of a trigger and a cursor to ensure accurate updates in real-time.

7. d) ALTER TRIGGER

8. a) We should use cursor in all cases

9. a) Implicit

10. d) %FOUND

11. a) Using a parametric cursor to retrieve sales data for the specified region.

12. b) Constrained cursor

13. b) Unconstrained cursor

14. c) Trigger action

15. b) UPDATE restriction

5.12 Assignment

MULTIPLE CHOICE QUESTIONS

1. What is the purpose of the trigger body in the trigger syntax?
   a) To define the trigger event
   b) To specify the table on which the trigger operates
   c) To define the actions to be performed when the trigger is fired
   d) To specify the condition for the trigger execution

NOTES

NOTES

2. Which type of cursor is more suitable when you need fine-grained control over the cursor operations, such as explicit opening, fetching, and closing?
   a) Implicit cursor
   b) Explicit cursor
   c) Constrained cursor
   d) Unconstrained cursor

3. Which type of cursor is automatically created and managed by the database engine without the need for explicit declaration and control?
   a) Implicit cursor
   b) Explicit cursor
   c) Constrained cursor
   d) Unconstrained cursor

4. You have developed a trigger that validates the data entered into a table and prevents invalid records from being inserted. However, after thorough testing, you realize that the trigger is causing performance issues. What can you do to improve performance without removing the trigger altogether?
   a) Enable the trigger
   b) Disable the trigger
   c) Modify the trigger
   d) Delete the trigger

5. You have implemented a trigger that updates a denormalized table whenever changes occur in a related table. Due to certain business requirements, you need to re-enable the trigger after it has been disabled. Which action should you take to accomplish this?
   a) Enable the trigger
   b) Disable the trigger
   c) Modify the trigger
   d) Delete the trigger

QUESTIONS

1. Create a trigger named "UpdateStockTrigger" that fires after an insert into the "Orders" table and updates the stock quantity in the "Products" table based on the quantity of items ordered.

2. You are writing a stored procedure that retrieves data from multiple tables with complex join conditions and calculations. The result set structure is fixed and known in advance. Which type of cursor variable would you choose? Explain the reason for your choice.

3. How would you create a cursor to retrieve the customer name and total orders for customers who have placed more than 5 orders?

4. How would you write a trigger to update the total order amount in the "Orders" table after an insert or update operation on the "OrderItems" table?

5. Define a trigger named "AuditLogTrigger" that fires after a delete on the "Employees" table and inserts a record into the "AuditLog" table to track the deleted employee details.

5.13 References

Books:

- https://www.google.co.in/books/edition/SQL_in_a_Nutshell/J1hcTQB3JbcC?hl=en&gbpv=1&dq=triggers+in+DBMS&printsec=frontcover
- https://www.google.co.in/books/edition/Oracle_PL_SQL_Programming/47WzweLc0uAC?hl=en&gbpv=1&dq=triggers+in+DBMS&printsec=frontcover

Web References:

- https://www.javatpoint.com/cursor-in-sql
- https://www.geeksforgeeks.org/cursors-in-pl-sql/
- https://www.geeksforgeeks.org/sql-triggers/

NOTES

BHARATI VIDYAPEETH (Deemed to be University), Pune
Centre for Distance and Online Education

Bharati Vidyapeeth Bhavan 5th Floor, L.B.S. Marg, Pune - 411030.
Tel. : 020-24407264, 9158990101, 9168300101
E-mail : cdoe.support@bharatividyapeeth.edu
Website : https://bharatividyapeethonline.com