Table of Contents

Triggers
- Example: SQL Query for Trigger

Mutating Errors
- Example: SQL Query for Mutating trigger

Instead of Triggers
- Benefits of using instead of triggers over regular triggers
- Limitations of instead of triggers
- Example: SQL Query for Instead trigger

Choosing between triggers and instead of triggers

Differences between triggers and instead of triggers
- Related posts:

Triggers

A trigger is a stored procedure that automatically executes in response to a specific event in the database, such as an INSERT, UPDATE, or DELETE statement on a particular table.

Triggers can be used to enforce data integrity, perform complex calculations, audit changes, or perform other actions automatically.

Example: SQL Query for Trigger

```
CREATE TRIGGER log_emp_inserts
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
  INSERT INTO logs (message, created_at)
  VALUES ('Employee inserted: ' || :new.name, sysdate);
END;
```

Explanation: This trigger creates a new row in the “logs” table whenever a new row is

inserted into the “employees” table. The message stored in the “logs” table includes the name of the employee and the date and time of the insertion.

Mutating Errors

Mutating errors occur when a trigger attempts to modify the data in the table that triggered the trigger itself.

This can lead to unexpected behavior and data inconsistencies.

Common causes of mutating errors include:

- Directly modifying the triggering table: This can lead to infinite loops or inconsistencies in the data.
- Reading from the triggering table: Even reading from the triggering table can lead to unexpected results if the trigger is also modifying the table.
- Using pseudo-columns like :old and :new: If you access these pseudo-columns before the UPDATE or DELETE statement has been fully executed, you may encounter a mutating error.

Example: SQL Query for Mutating trigger

```
CREATE TRIGGER update_employee_salary
AFTER UPDATE ON employees
FOR EACH ROW
```

```
BEGIN
  UPDATE employees SET salary = salary * 1.1
  WHERE employee_id = :old.employee_id;
END;
```

Explanation: This trigger attempts to update the salary of the employee who just had their record updated. However, this will result in a mutating error because the trigger is modifying the “employees” table, which is the same table that triggered the trigger.

Instead of Triggers

Instead of triggers are a special type of trigger that can be used to replace the default behavior of an INSERT, UPDATE, or DELETE statement.

Instead of executing the original statement, the trigger code is executed instead.

This can be useful for situations where you want to completely customize the behavior of an operation.

Benefits of using instead of triggers over regular triggers

- Prevents mutating errors: Since the original statement is not executed, there is no risk of modifying the triggering table and causing errors.
- Improves performance: Instead of triggers can be more efficient than regular triggers because they avoid the overhead of executing the original statement.

- Offers greater flexibility: Instead of triggers allow for complete customization of the behavior of an operation, which is not possible with regular triggers.

Limitations of instead of triggers

- Less intuitive: The code for an instead of trigger can be more complex and difficult to understand than the code for a regular trigger.
- Limited scope: Instead of triggers can only be used for INSERT, UPDATE, and DELETE statements.
- Debugging can be challenging: Debugging an instead of trigger can be more difficult than debugging a regular trigger because the original statement is not executed.

Example: SQL Query for Instead trigger

```
CREATE TRIGGER calculate_employee_bonus
INSTEAD OF INSERT ON employees
FOR EACH ROW
BEGIN
  IF :new.salary > 50000 THEN
    INSERT INTO employee_bonuses (employee_id, bonus)
    VALUES (:new.employee_id, :new.salary * 0.1);
  END IF;
  INSERT INTO employees (employee_id, name, salary)
  VALUES (:new.employee_id, :new.name, :new.salary);
END;
```

Explanation: This instead of trigger replaces the default behavior of the INSERT statement on the “employees” table. It checks the salary of the newly inserted employee and if it is above a certain threshold, it calculates and inserts a bonus record for the employee. It then finally

inserts the new employee record into the “employees” table.

Choosing between triggers and instead of triggers

The choice between a trigger and an instead of trigger depends on specific needs. If there is a requirement to simply perform an action after an event occurs, a regular trigger may suffice.

However, in cases where data modification or the complete replacement of operation behavior is necessary, an instead of trigger might be a more suitable option.

Differences between triggers and instead of triggers

Feature	Trigger	Instead of Trigger
Execution timing	After an event	Instead of the original statement
Data modification	Can modify	Cannot modify
Performance	May be less efficient	May be more efficient
Flexibility	Less flexible	More flexible
Debugging	Easier	More challenging

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. Dedicated Server vs Multi-Threaded Server
46. Distributed database, database links, and snapshot
47. RDBMS Security
48. SQL queries for various join types
49. Cursor management: nested and parameterized cursors
50. Oracle exception handling mechanism

51. Stored Procedures and Parameters