SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS Specifying Constraints as Assertions and Actions as Triggers

Course Name : 23CAT603 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / I Semester

Unit IV – Specifying Constraints as Assertions and Actions as Triggers Queries

Specifying Constraints as Assertions and Actions as Triggers

Assertions and Triggers in DBMS

In DBMS, there is always the need to make sure that data in the database is both consistent and intact. This can be done with the help of mechanisms such as assertions and triggers or any other similar one. They both encourage compliance with rules and conditions but they do it in different ways and at different times.

What are Assertions?

When a constraint involves 2 (or) more tables, the table constraint mechanism is sometimes hard and results may not come as expected. To cover such a situation SQL supports the creation of assertions that are constraints not associated with only one table. An assertion statement should ensure a certain condition will always exist in the database. DBMS always checks the assertion whenever modifications are done in the corresponding table.

Syntax –

```
CREATE ASSERTION [ assertion_name ] CHECK ( [ condition ] );
```

Specifying Constraints as Assertions and Actions as Triggers

Example

```
CREATE TABLE sailors (sid int,sname varchar(20), rating int,primary key(sid), CHECK(rating >= 1 AND
rating <=10) CHECK((select count(s.sid) from sailors s) + (select count(b.bid)from boats b)<100) );
```

In the above example, we enforcing CHECK constraint that the number of boats and sailors should be less than 100. So here we are able to **CHECK constraints** of two tablets simultaneously.

SQL | CHECK Constraint

SQL Constraints Check Constraint is used to specify a predicate that every tuple must satisfy in a given relation. It limits the values that a column can hold in a relation.

- The predicate in check constraint can hold a sub query.
- Check constraint defined on an attribute restricts the range of values for that attribute.
- If the value being added to an attribute of a tuple violates the check constraint, the check constraint evaluates to false and the corresponding update is aborted.
- Check constraint is generally specified with the CREATE TABLE command in SQL.

Specifying Constraints as Assertions and Actions as Triggers

Syntax:

```
CREATE TABLE pets( ID INT NOT NULL,
Name VARCHAR(30) NOT NULL, Breed
VARCHAR(20) NOT NULL, Age INT,
GENDER VARCHAR(9), PRIMARY KEY(ID),
check(GENDER in ('Male', 'Female',
'Unknown')) );
```

Note: The check constraint in the above SQL command restricts the GENDER to belong to only the categories specified. If a new tuple is added or an existing tuple in the relation is updated with a GENDER that doesn't belong to any of the three categories mentioned, then the corresponding database update is aborted.

Query

Constraint: Only students with age >= 17 are can enroll themselves in a university. **Schema for student database in university:**

Student relation:

StudentID	Name	Age	Gender
1001	Ron	18	Male
1002	Sam	17	Male
1003	Georgia	17	Female
1004	Erik	19	Unknown
1005	Christine	17	Female

```
CREATE TABLE student( StudentID INT NOT
NULL, Name VARCHAR(30) NOT NULL, Age INT
NOT NULL, GENDER VARCHAR(9), PRIMARY
KEY(ID), check(Age >= 17) );
```

Specifying Constraints as Assertions and Actions as Triggers

Explanation: In the above relation, the age of all students is greater than equal to 17 years, according to the constraint mentioned in the check statement in the schema of the relation. If, however following SQL statement is executed:

```
INSERT INTO student(STUDENTID, NAME, AGE, GENDER) VALUES (1006, 'Emma', 16, 'Female');
```

There won't be any database update and as the age < 17 years. **Different options to use Check constraint:**
With alter: Check constraint can also be added to an already created relation using the syntax:

```
alter table TABLE_NAME modify COLUMN_NAME check(Predicate);
```

Giving variable name to check constraint: Check constraints can be given a variable name using the syntax:

```
alter table TABLE_NAME add constraint CHECK_CONST check (Predicate);
```

Remove check constraint: Check constraint can be removed from the relation in the database from SQL server using the syntax:

```
alter table TABLE_NAME drop constraint CHECK_CONSTRAINT_NAME;
```

Drop check constraint: Check constraint can be dropped from the relation in the database in MySQL using the syntax:

```
alter table TABLE_NAME drop check CHECK_CONSTRAINT_NAME;
```

Specifying Constraints as Assertions and Actions as Triggers

View existing constraints on a particular table

If you want to check if a constraint or any constraint exists within the table in mysql then you can use the following command. This command will show a tabular output of all the constraint-related data for the table name you've passed in the statement, in our case we'll use the employee table.

```
SELECT * FROM information_schema.table_constraints WHERE table_schema = schema() AND table_name = 'employee';
```

Specifying Constraints as Assertions and Actions as Triggers

What are Triggers?

A trigger is a database object that is associated with the table, it will be activated when a defined action is executed for the table.
The trigger can be executed when we run the following statements:
INSERT
UPDATE
DELETE
And it can be invoked before or after the event.

Syntax –

```
create trigger [trigger_name]
[before | after] {insert |
update | delete}
on [table_name]
[for each row]
[trigger_body]
```

Example

In the above example, we are creating triggers before updates. so, if the new age is greater than 60 we should not update else we should update. We can call this trigger by using “$” symbol.

```
create trigger t1 before UPDATE on sailors
for each row
begin
     if new.age>60 then
          set new.age=old.age;
     else set new.age=new.age;
     end if;
end;
$
```

Specifying Constraints as Assertions and Actions as Triggers

Difference Between Assertions and Triggers

Assertions	Triggers
We can use Assertions when we know that the given particular condition is always true.	We can use Triggers even particular condition may or may not be true.
When the SQL condition is not met then there are chances to an entire table or even Database to get locked up.	Triggers can catch errors if the condition of the query is not true.
Assertions are not linked to specific table or event. It performs task specified or defined by the user.	It helps in maintaining the integrity constraints in the database tables, especially when the primary key and foreign key constraint are not defined.
Assertions do not maintain any track of changes made in table.	Triggers maintain track of all changes occurred in table.
Assertions have small syntax compared to Triggers.	They have large Syntax to indicate each and every specific of the created trigger.

Specifying Constraints as Assertions and Actions as Triggers

Difference Between Assertions and Triggers

Assertions	Triggers
Modern databases do not use Assertions.	Triggers are very well used in modern databases.
Purpose of assertions is to Enforces business rules and constraints.	Purpose of triggers is to Executes actions in response to data changes.
Activation is checked after a transaction completes	Activation is activated by data changes during a transaction
Granularity applies to the entire database	Granularity applies to a specific table or view
Syntax Uses SQL statements	Syntax Uses procedural code (e.g. PL/SQL, T-SQL)

Specifying Constraints as Assertions and Actions as Triggers

Difference Between Assertions and Triggers

Assertions	Triggers
Error handling Causes transaction to be rolled back.	Error handling can ignore errors or handle them explicitly
Assertions may slow down performance of queries.	Triggers Can impact performance of data changes.
Assertions are Easy to debug with SQL statements.	Triggers are more difficult to debug procedural code
Examples- CHECK constraints, FOREIGN KEY constraints	Examples – AFTER INSERT triggers, INSTEAD OF triggers

Assertions and triggers are both important in managing the databases because of the following reasons. Assertions are especially useful for expressing and enforcing compound existing and future conditions across multiple tables, on the other hand, trigger are all-inclusive and are broadly used for other purposes such as automating responses to alterations to data. Comparing the described above two mechanisms makes understanding of what tool to choose for enforcing business rules and maintaining data integrity in the hands of a database administrator.

3. Grouping and Aggregation:

Aggregation functions such as COUNT(), SUM(), AVG(), MAX(), MIN() are used with GROUP BY to summarize data.

- **GROUP BY**:

sql

```
SELECT CustomerID, COUNT(*) AS TotalOrders FROM Orders GROUP BY CustomerID;
```

- **HAVING**: This clause is used to filter groups after aggregation.

sql

```
SELECT CustomerID, COUNT(*) AS TotalOrders FROM Orders GROUP BY CustomerID HAVING COUNT(*) > 5;
```

4. Complex Filters:

Using multiple conditions in the WHERE clause can make queries more complex.

- Combining AND, OR, and NOT:

sql

```
SELECT * FROM Orders WHERE (OrderDate > '2024-01-01' AND CustomerID = 101) OR (TotalAmount > 500);
```

5. Union and Union All:

These are used to combine the results of two or more SELECT statements into a single result set.

- **UNION** removes duplicates:

sql

```
SELECT CustomerName FROM Customers UNION SELECT CustomerName FROM Suppliers;
```

- **UNION ALL** includes duplicates:

sql

```
SELECT CustomerName FROM Customers UNION ALL SELECT CustomerName FROM Suppliers;
```

6. Case Statements:

A CASE statement is used to perform conditional logic within a query.

sql

```
SELECT CustomerName, CASE WHEN TotalAmount > 500 THEN 'VIP' ELSE 'Regular' END AS CustomerStatus FROM Customers;
```

7. Window Functions:

Window functions are used to perform calculations across a set of table rows that are related to the current row.

- **ROW_NUMBER()**:

sql

```
SELECT CustomerID, OrderID, ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS RowNum FROM Orders;
```

Schema Modification in DBMS involves changing the structure of a database, such as adding, altering, or deleting tables, columns, and constraints.Complex SQL Retrieval Queries involve sophisticated querying techniques such as joins, subqueries, aggregations, and window functions to extract meaningful information from multiple tables in a database. These queries allow for powerful data analysis and manipulation.Together, schema modifications and complex queries enable the creation of flexible, high-performance database systems capable of handling intricate data relationships and operations.

References

1. https://www.geeksforgeeks.org/difference-between-assertions-and-triggers-in-dbms/
2. https://www.geeksforgeeks.org/sql-check-constraint/
3. https://www.rose-hulman.edu/class/cs/csse333/Slides/Triggers.pdf