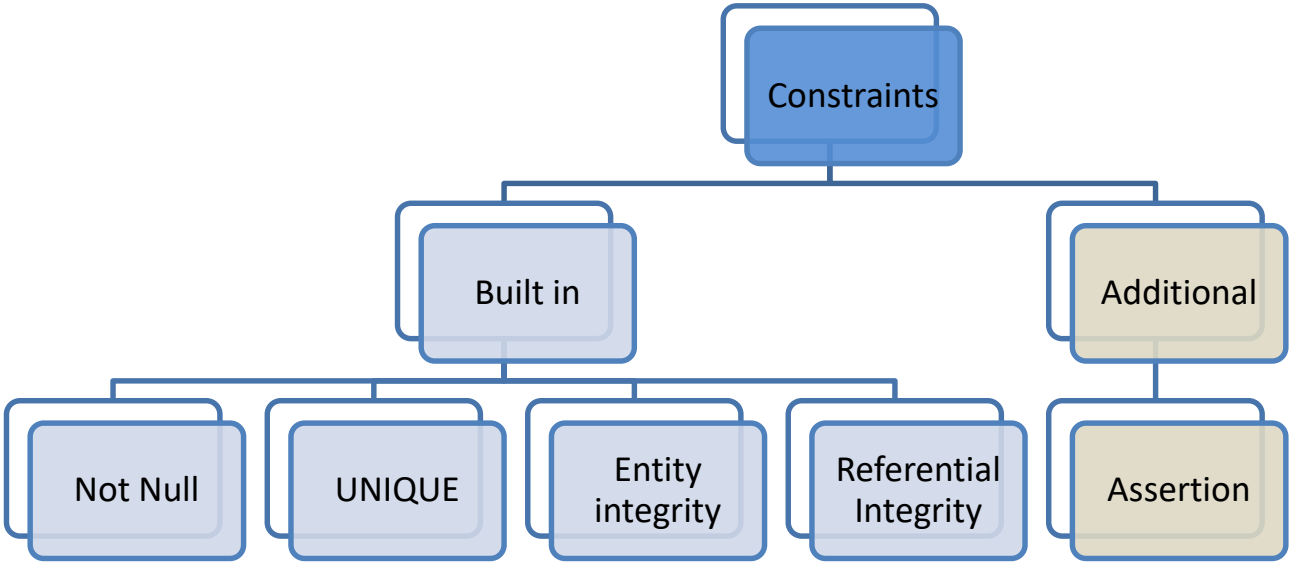
Constraints as Assertions

COURSE : 23CAT- Database Management System
UNIT I : Introduction
CLASS : I Semester / I MCA

Triggers


Constraints
Built in
Additional
Not Null
UNIQUE
Entity integrity
Referential Integrity
Assertion

- **Semantic Constraints**: The following are beyond the scope of the EER and relational model

- **CREATE ASSERTION**
  Specify additional types of constraints outside scope of built-in relational model constraints

- **CREATE TRIGGER**
  Specify automatic actions that database system will perform when certain events and conditions occur

Assertion

- Specify a query that selects any tuples that violate the desired condition
- Use only in cases where it goes beyond a simple CHECK which applies to individual attributes and domains

EXAMPLE

Condition:

Salary of an Employee must not be greater than the salary of the Manager of the corresponding department

```
CREATE ASSERTION SALARY _ CONSTRAINT
    CHECK ( NOT EXISTS ( SELECT * EMPLOYEE E, EMPLOYEE M,
                         DEPARTMENT D E . SAL > M . SAL AND
                         D E . SAL > M . SAL AND
                         D MAR . SSN = M . SSN ) )
```

❑ Schema evolution commands

- DBA may want to change the schema while the database is operational
- Does not require recompilation of the database schema

❑ **DROP** command
- drop named schema elements, such as tables, domains, or constraint
- Options: CASCADE and RESTRICT
- CASCADE removes the schema and all its elements including tables, views, constraints, etc.
- RESTRICT: drops only nothing in it

DROP SCHEMA COMPANY CASCADE

Alter table actions include:

- Adding or dropping a column (attribute)
- Changing a column definition
- Adding or dropping table constraints

Example:

```
ALTER TABLE COMPANY.EMPLOYEE
ADD COLUMN Job VARCHAR(12);
```

Change constraints specified on a table

- Add or drop a named constraint

```
ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE;
```

Trigger Example

```
SQL>CREATE OR REPLACE TRIGGER derive_commission_trg
  2 BEFORE UPDATE OF sal ON emp
  3 FOR EACH ROW
  4 WHEN (new.job = 'SALESMAN')
  5 BEGIN
  6   :new.comm := :old.comm * (:new.sal/:old.sal);
  7 END;
  8 /
```

Trigger name:	derive_commission_trg
Timing:	BEFORE executing the statement
Triggering event:	UPDATE of sal column
Filtering condition:	job = 'SALESMAN'
Target:	emp table
Trigger parameters:	old, new
Trigger action:	calculate the new commission to be updated

Schema Changes

Assertions	Triggers
use Assertions when the given particular condition is always true	We can use Triggers even particular condition may or may not be 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 tables
Do not maintain any track of changes made in table.	It maintains track of all changes occurred in table.
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.
Assertions may slow down performance of queries.	Triggers Can impact performance of data changes.
Examples- CHECK constraints, FOREIGN KEY constraints	Examples – AFTER INSERT triggers, INSTEAD OF triggers

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Triggering Event

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS, SALARY) VALUES (7, 'Kriti', 22, 'HP', 15000.00);

Output

```
Old salary:
New salary: 15000
Salary difference:
```

UPDATE customers SET salary = salary + 3000
WHERE id = 7;

Output

```
Old salary: 15000
New salary: 18000
Salary difference: 3000
```