SQL Constraints: Comprehensive Review

DSC 208R - Data Management for Analytics

Introduction to SQL Constraints

Constraints are rules enforced on data columns in a table. They are used to limit the type of data that can go into a table, ensuring the accuracy and reliability of the data. Constraints can be column level or table level.

Types of Constraints

- **Key Constraints**:
  - 'PRIMARY KEY'
  - 'FOREIGN KEY'
- **Value Constraints**:
  - 'NOT NULL'
  - 'UNIQUE'
  - 'CHECK'
  - 'DEFAULT'

Key Constraints

'PRIMARY KEY'

A 'PRIMARY KEY' uniquely identifies each record in a table.

- It must contain unique values.
- It cannot contain 'NULL' values.
- A table can have only one primary key.
- It can consist of single or multiple columns.

Example (Single Column Primary Key):

```
CREATE TABLE Movie (
    name VARCHAR(50) PRIMARY KEY,
    year INT,
    genre VARCHAR(50)
);
```

Example (Multi-column Primary Key):

```
CREATE TABLE ActedIn (
    actorname VARCHAR(50),
    moviename VARCHAR(50),
    PRIMARY KEY (actorname, moviename)
);
```

‘FOREIGN KEY‘

A ‘FOREIGN KEY‘ is a field (or collection of fields) in one table that refers to the ‘PRIMARY KEY‘ in another table.

- It establishes a link between two tables.
- It ensures referential integrity, meaning that relationships between tables are consistent.

Example:

```
CREATE TABLE ActedIn (
    actorname VARCHAR(50),
    moviename VARCHAR(50),
    PRIMARY KEY (actorname, moviename),
    FOREIGN KEY (moviename) REFERENCES Movie(name)
);
```

Explanation: This constraint ensures that any ‘moviename‘ inserted into ‘ActedIn‘ must already exist as a ‘name‘ in the ‘Movie‘ table.

Referential Integrity Options for ‘FOREIGN KEY‘

These options define what happens to foreign key rows when the referenced primary key row is deleted or updated.

- **‘ON DELETE‘ / ‘ON UPDATE‘**: Specifies actions for delete or update operations on the referenced primary key.
  - **‘CASCADE‘**: If a row in the parent table (with the primary key) is deleted or updated, the corresponding rows in the child table (with the foreign key) are also deleted or updated.
  - **‘RESTRICT‘**: Prevents the deletion or update of a parent row if there are dependent rows in the child table. This is often the default behavior.
  - **‘SET NULL‘**: Sets the foreign key column(s) in the child table to ‘NULL‘ when the parent row is deleted or updated. Requires the foreign key column to be nullable.
  - **‘SET DEFAULT‘**: Sets the foreign key column(s) in the child table to their default value when the parent row is deleted or updated.

Example with ‘ON DELETE CASCADE‘:

```
CREATE TABLE ActedIn (
    actorname VARCHAR(50),
    moviename VARCHAR(50),
    PRIMARY KEY (actorname, moviename),
    FOREIGN KEY (moviename) REFERENCES Movie(name) ON DELETE CASCADE
);
```

Explanation: If a movie is deleted from the ‘Movie‘ table, all ‘ActedIn‘ records referencing that movie will automatically be deleted.

Value Constraints

‘NOT NULL‘

The ‘NOT NULL‘ constraint ensures that a column cannot have a ‘NULL‘ value.

Example:

```
CREATE TABLE Movie (
    name VARCHAR(50) PRIMARY KEY,
    year INT NOT NULL,
    genre VARCHAR(50)
);
```

Explanation: This ensures that every movie record must have a ‘year‘ value.

‘UNIQUE‘

The ‘UNIQUE‘ constraint ensures that all values in a column (or a set of columns) are different.

- It allows ‘NULL‘ values, but only once (behavior can vary slightly by DBMS).
- A table can have multiple ‘UNIQUE‘ constraints.

Example:

```
CREATE TABLE Person (
    id INT PRIMARY KEY,
    email VARCHAR(100) UNIQUE,
    name VARCHAR(50)
);
```

Explanation: This ensures that no two persons can have the same ‘email‘.

‘CHECK‘

The ‘CHECK‘ constraint is used to limit the value range that can be placed in a column.
Example:

```
CREATE TABLE Actor (
    name VARCHAR(50) PRIMARY KEY,
    age INT CHECK (age > 0 AND age < 120)
);
```

Explanation: This ensures that the ‘age‘ of an actor must be between 1 and 119, inclusive.

‘DEFAULT‘

The ‘DEFAULT‘ constraint is used to set a default value for a column when no value is specified during an ‘INSERT‘ operation.
Example:

```
CREATE TABLE Movie (
    name VARCHAR(50) PRIMARY KEY,
    year INT NOT NULL,
    genre VARCHAR(50) DEFAULT 'Unspecified'
);
```

Explanation: If a ‘genre‘ is not provided when inserting a new movie, it will automatically be set to ’Unspecified’.

Adding and Dropping Constraints with ‘ALTER TABLE‘

Adding a ‘PRIMARY KEY‘

```
ALTER TABLE Movie
ADD PRIMARY KEY (name);
```

Adding a ‘FOREIGN KEY‘

```
ALTER TABLE ActedIn
ADD FOREIGN KEY (moviename) REFERENCES Movie(name);
```

Adding a ‘CHECK‘ Constraint

```
ALTER TABLE Actor
ADD CHECK (age > 0 AND age < 120);
```

Dropping a ‘PRIMARY KEY‘

```
ALTER TABLE Movie
DROP PRIMARY KEY;
```

Dropping a ‘FOREIGN KEY‘

```
ALTER TABLE ActedIn
DROP FOREIGN KEY moviename; -- Actual constraint name might be needed for some DBMS
```

Note: For some DBMS, you might need to specify the system-generated or user-defined name of the foreign key constraint instead of the column name.

Dropping a ‘CHECK‘ Constraint

```
ALTER TABLE Actor
DROP CHECK age_check; -- Assuming 'age_check' is the constraint name
```

Note: You typically need the specific name of the ‘CHECK‘ constraint to drop it.