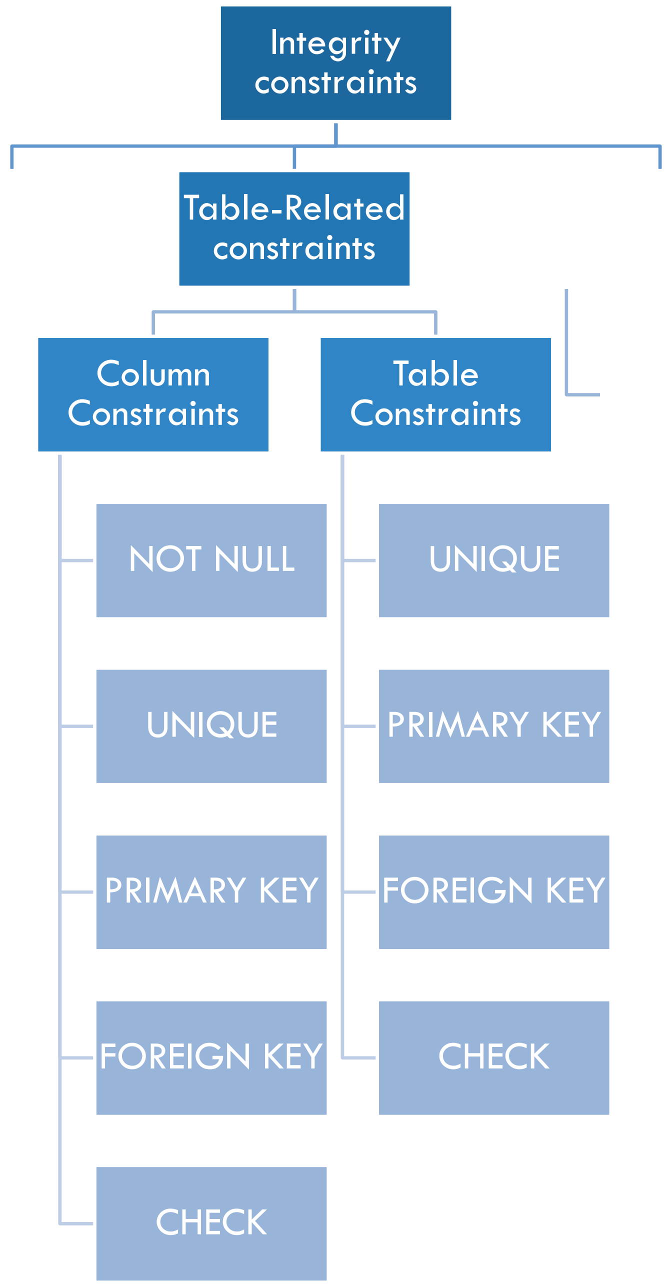
CHAPTER 10: SQL CONSTRAINTS

2023

CHAPTER 10: SQL CONSTRAINTS

Content:

SQL CONSTRAINTS



SQL CONSTRAINTS

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

If there is any violation between the constraint and the data action, the action is aborted.

constraint	Description
NOT NULL	Ensures that a column cannot have a NULL value
UNIQUE	Ensures that all values in a column are different
PRIMARY KEY	A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY	Prevents actions that would destroy links between tables
CHECK	Ensures that the values in a column satisfies a specific condition
DEFAULT	Sets a default value for a column if no value is specified
CREATE INDEX	Used to create and retrieve data from the database very quickly

NAMING CONVENTIONS

Common naming conventions include:

CHECK constraints: ck_*tablename*_*columnname*

PRIMARY KEY constraints: pk_*tablename*

UNIQUE constraints: uq_*tablename*_*columnname*

FOREIGN KEY constraints: fk_*tablename*_*referencedtablename*

NOT NULL CONSTRAINT

If we specify a field in a table to be NOT NULL. Then the field will never accept null value. That is, you will be not allowed to insert a new row in the table without specifying any value to this field.

For example, the below query creates a table Student with the fields ID and NAME as NOT NULL. That is, we are bound to specify values for these two fields every time we wish to insert a new row.

```
CREATE TABLE Student
 (ID int(6) NOT NULL,
 NAME varchar(10) NOT NULL,
 ADDRESS varchar(20)
 );
```

UNIQUE CONSTRAINT

- ☐ It allows only unique values and won't allow duplicates
- ☐ For unique key an implicit unique index get defined
- ☐ A unique constraint allows you to require that a column or *set of columns* contain unique values only
- ☐ A unique key can hold maximum of 32 columns

```
CREATE TABLE Music
(
Artist varchar(30) ,
CDName varchar(100) UNIQUE,
Year int
);
```

```
CREATE TABLE Music
(
Artist varchar(30) ,
CDName varchar(100),
Year int,
Constraint un_ArtistCD UNIQUE (Artist, CDName)
);
```

PRIMARY KEY CONSTRAINT

- It acts as both (unique +not null) which means it won't allow duplicate and null values
- IA unique index is defined on primary key columns
- There should be only one primary key for an entire table
- A Primary Key field can hold maximum of 32 columns
- Materialized view get defined only on tables, which contains primary keys
- Generally, we call primary key table, a *parent* table
- If a value should be generated by default, add "Auto_increment" in the syntax

PRIMARY KEY CONSTRAINT

```
Create table cats

Create table cats

(

CatID int NOT NULL auto_increment Primary key,

Name varchar(20) NOT NULL ,

age int NOT NULL  ,

dob date default getdate()

)
```

OR

```
(

CatID int NOT NULL auto_increment ,

Name varchar(20) NOT NULL,

age int NOT NULL ,

dob date default getdate() ,

primary key(catID)

)
```

PRIMARY KEY CONSTRAINT (MULTIPLE COLUMNS)

```
Create table people
(
Name varchar(20) NOT NULL,
Surname varchar(20),
age int NOT NULL ,
dob date default getdate() ,
Constraint pk_NameSur primary key(Name, Surname)
)
```

```
CONSTRAINT constraintName Type FOREIGN KEY (curcolName) REFERENCES otherTbl (PrimaryKeyColName)
```

FOREIGN KEY CONSTRAINT

- It's a reference integrity constraint - if ever you provide any value into the foreign key columns before begin inserted, that value will be referred through foreign key with the primary key column
- The foreign key allows null values for flexibility and allows duplicates
- Primary Key and Foreign Key column names could be different, but data types should be same

```
CREATE TABLE employees (

employee_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) UNIQUE,

department_id INT,

CONSTRAINT fk_employees_departmentid FOREIGN KEY (department_id)

REFERENCES departments (department_id)
```

```
CREATE TABLE employees (

employee_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) UNIQUE,

department_id INT REFERENCES departments (department_id)

);
```

DEFAULT CONSTRAINT

- It takes default values (if user won’t provide any value to a columns, then default provides default values to columns).
- It won’t allow sub queries and user defined functions.

```
CREATE TABLE Colleges (
  college_id INT PRIMARY KEY,
  college_code VARCHAR(20),
  college_country VARCHAR(20) DEFAULT 'US'
)
```

CHECK CONSTRAINT

☐ To restrict/enforce other than standard integrity rules (Primary key (unique + not null) we use check constraints.

☐ Check constraint throws an error only when condition becomes false, won't throw for 'true and null'

☐ It won't allow sub queries as a condition

☐ It won't allow user defined functions

CHECK CONSTRAINT

The CHECK constraint is used to limit the value range (*Data validation*) that can be placed in a column.

If you define a CHECK constraint <u>on a column</u> it will allow only certain values for this column.

If you define a CHECK constraint <u>on a table</u>, it can limit the values in certain columns based on values in other columns in the row.

Note that you need to first create the field then create the check.

Example:

The following SQL example is a table creation and creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18 or older:

CHECK CONSTRAINT

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL ,
    FirstName varchar(255) ,
    Age int CHECK (Age>=18)
)
```

CHECK CONSTRAINT

```
CREATE TABLE employees (
  employee_id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  email VARCHAR(50) UNIQUE,
  salary MONEY CHECK (salary >= 0),
  department_id INT REFERENCES departments
(department_id)
);
```

CHECK EXISTING CONSTRAINTS

Check constraints

```
SELECT CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS
```

other constraints

```
SELECT CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
```

Practice CONSTRAINTS

In the HR database Create the PL_employees table so that it has the following fields (Employee ID, First name, last name, Wage amount, Age, citizenship , Email) with these constraints:

1. Make sure that the Employee ID is a primary key
2. First/last names should not be greater than 20 characters
3. Ensure that the employees are earning more than the minimum wage of 10,000
4. The employees are older than 18 years.
5. Add a column for citizenship and validates that the citizenship is South African.
6. Ensure the values are not null when the record is added.
7. Ensure the email address of the employees are unique.

Once your table has been created check all the constraints applied.

ALTER CONSTRAINTS

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name constraint_type (column_name);

ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

```
ALTER TABLE employees
ADD CONSTRAINT ck_employees_name CHECK (name IS NOT NULL);
```

In this example, the ALTER TABLE statement modifies the employees table and adds a constraint named ck_employees_name that ensures the name column cannot contain a NULL value.

Note: In SQL Server, the naming conventions for constraints follow the pattern ck_<table_name>_<column_name> for CHECK constraints, pk_<table_name> for PRIMARY KEY constraints, fk_<table_name>_<referenced_table_name> for FOREIGN KEY constraints, etc.

```
ALTER TABLE employees
ADD CONSTRAINT ck_employees_name CHECK (name IS NOT NULL);

ALTER TABLE employees
ADD CONSTRAINT pk_employees PRIMARY KEY (employee_id);

ALTER TABLE employees
ADD CONSTRAINT uq_employees_email UNIQUE (email);

ALTER TABLE employees
ADD CONSTRAINT fk_employees_departments FOREIGN KEY (department_id)
REFERENCES departments (department_id);
```

AUTO-INCREMENT

SQL Server:

```
ALTER TABLE employees
ADD id INT IDENTITY(1,1) NOT NULL;
```

```
ALTER TABLE employees
ALTER COLUMN id INT IDENTITY(1,1)
NOT NULL;
```

MySQL

```
ALTER TABLE employees
MODIFY COLUMN id INT
AUTO_INCREMENT PRIMARY KEY;
```

```
ALTER TABLE employees
ADD COLUMN id INT
AUTO_INCREMENT PRIMARY KEY;
```

DROPPING CONSTRAINTS

Recall that when you create a table/field you sometimes pass constraints on your fields to ensure that data entry is up to standard and data is usable.

These are the UNIQUE PRIMARY KEY FOREIGN KEY or CHECK constraints. These can be dropped (deleted).

Syntax to drop a constraint:

```
ALTER TABLE table_name
        DROP CONSTRAINT field_name;
```

Example for **Unique** constraint:

```
ALTER TABLE Persons

DROP CONSTRAINT UC_Person;
```

UC- unique constraint…

CONSTRAINT

Example for **Primary Key** constraint:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

Example for **Foreign Key** constraint:

```
ALTER TABLE Persons
DROP CONSTRAINT FK_Person;
```

Example for **Check** constraint:

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_Person;
```

DROP CONSTRAINTS

The DROP CONSTRAINT command is used to delete a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint.

Syntax example MSQL:

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

Syntax example MySQL:

```
ALTER TABLE Persons
DROP INDEX UC_Person;
```

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

Practice CONSTRAINTS

In the HR database Create a table employment types table that has the following fields:

job_id as a PRIMARY KEY

job_title,

min_salary

max_salary

Practice CONSTRAINTS (ALTER)

On the Employment type table that was created in a previous exercise, add the following constraints:

1. job_id as a PRIMARY KEY (remove this constraint)
2. job_title – must be unique
3. min_salary – Above 10 000
4. max_salary – Below 100 000