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. Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table. The following constraints are commonly used in SQL:

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 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 value 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

SQL NOT NULL Constraint

By default, a column can hold NULL values. The NOT NULL constraint enforces a column to NOT accept

NULL values. This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

```
CREATE TABLE Persons
( ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255) NOT NULL,
 Age int );
```

SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different. Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns. PRIMARY KEY constraint automatically has a UNIQUE constraint. However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

```
CREATE TABLE Persons (
```

```
ID int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int    );

CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CONSTRAINT UC_Person UNIQUE (ID,LastName)  );
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values.A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

```
CREATE TABLE Persons (
ID int NOT NULL PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int );
```

Multiple Primary key

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
phoneno varchar(20),
CONSTRAINT PK_Person PRIMARY KEY (ID,phoneno)    );
```

SQL FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables. A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

```
CREATE TABLE Customers(
c_id INT PRIMARY KEY,
c_name VARCHAR(255),
c_address varchar )

CREATE TABLE Orders (
order_id INT PRIMARY KEY,
c_id int REFERENCES Customers(c_id) );
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column. If you define a CHECK constraint on a column it will allow only certain values for this column. If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

The following SQL 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:

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

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

SQL DEFAULT on CREATE TABLE

The DEFAULT constraint is used to set a default value for a column. The default value will be added to all new records, if no other value is specified.

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

```
CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int ,

City varchar(255) DEFAULT 'Sundarharaincha'  );
```

```
CREATE TABLE Persons (

Personid int NOT NULL AUTO_INCREMENT,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (Personid) );
```

SQL CREATE INDEX Statement

The CREATE INDEX statement is used to create indexes in tables. Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

```
CREATE UNIQUE INDEX index_name

ON table_name (column1, column2, ...);
```

```
Create Unique Index std_index on

Student(id,name,rollno)
```

Sql Query

Create :

```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
column3 datatype,
....       );

CREATE TABLE Persons (
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255),
primary key (PersonID) );
```

Sql Query

Create :

```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
column3 datatype,
....   );

CREATE TABLE Persons (
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255),
primary key (PersonID)    );

CREATE TABLE Persons (
PersonID int,
```

```
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255),
Salary numeric(8,2)
primary key (PersonID) );

CREATE TABLE Employees (
EmpID varchar (255) PRIMARY KEY,
EmpName varchar(255) NOT NULL,
Gender varchar(10) check(Gender in ('M', 'F', 'O')),
Age int check (Age>16) );

Create table Students (
SID varchar(200) Primary Key,
Name varchar(255) Not Null,
Login varchar(255) UNIQUE,
Age int DEFAULT '18,
GPA neumeric(3,2) check(GPA BETWEEN 0 AND 4 ) );
```

Foreign Key

```
CREATE TABLE student (
id INT PRIMARY KEY,
first_name VARCHAR(100) NOT NULL,
last_name VARCHAR(100) NOT NULL,
city_id INT FOREIGN KEY REFERENCES city(id)    );
```

Basic Retrieval Queries

1. The SELECT-FROM-WHERE Structure of Basic SQL Queries

SELECT <attribute list>

FROM <table list>

WHERE <condition>;

WHERE Clause

The WHERE clause is used to fetch the data which specify the given condition. It is used to filter records and select only necessary records. It is used with SELECT, UPDATE, DELETE, etc. query. SQL also implements and, or, and not in the WHERE clause which is known as the Booleancondition.

HAVING Clause

The HAVING clause is generally used along with the GROUP BY clause. This clause is used in the column operation and is applied to aggregate rows or groups according to given conditions.

Select Clause

From Clause

Where Clause

Basic Retrieval Queries

Unspecified WHERE Clause and Use of the Asterisk

 Select *

Substring Pattern Matching and Arithmetic Operators

Like (wild card)

+,-,*,/

Group By

SELECT COUNT(CustomerID), Country FROM Customers GROUP BY Country;

TOP clause

The SELECT TOP clause is used to specify the number of records to return. The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

SELECT TOP 3 * FROM Customers;

Ordering of Query Results

SELECT * FROM members ORDER BY date_of_birth DESC;

SELECT * FROM members ORDER BY date_of_birth ASC

Drop table

DROP TABLE table_name;

INSERT INTO table_name (column1, column2, column3, ...)VALUES (value1, value2, value3, ...);

SELECT * FROM table_name;

SELECT * FROM Customers WHERE Country='Mexico';

UPDATE CustomersSET ContactName = 'Alfred Schmidt',

City= 'Frankfurt' WHERE CustomerID = 1;

DELETE FROM table_name WHERE condition;

Some questions

Create a table Student_info as follows and perform the query according to condition as below.

i. Insert a record on the table.
ii. Display all the record of the table .
iii. Display record whose Address is Itahari.
iv. Display Name of student whose Age greater than 25.
v. Update and set age to 20 of student whose Id is 5.
vi. Delete record of a student whose Id is 25.
vii. Delete a table.

AND OR NOT Opearators

SELECT * FROM Customers WHERE Country='Germany' AND City='Berlin';

SELECT * FROM Customers WHERE City='Berlin' OR City='München';

SELECT * FROM Customers WHERE Country='Germany' OR Country='Spain';

SELECT * FROM Customers WHERE NOT Country='Germany';

Order BY

SELECT * FROM Customers ORDER BY Country;

SELECT * FROM Customers ORDER BY Country DESC;

SELECT * FROM Customers ORDER BY Country ASC;

Group BY

The GROUP BY statement groups rows that have the same values into

summary rows, like "find the number of customers in each country".

Null Value

SELECT CustomerName, ContactName, Address FROM Customers WHERE Address IS NULL;

i. SELECT MAX(Price) AS LargestPrice
ii. FROM Products;
iii. SELECT DISTINCT Country FROM Customers;
iv. SELECT COUNT(DISTINCT Country) FROM Customers;
v. SELECT COUNT(ProductID)FROM Products;
vi. SELECT AVG(Price) FROM Products;
vii. SELECT SUM(Quantity) FROM OrderDetails;
viii. SELECT * FROM CustomersWHERE CustomerName LIKE 'a%';
ix. SELECT * FROM Customers WHERE CustomerName LIKE '%a';
x. SELECT * FROM Customers WHERE CustomerName LIKE '%or%';
xi. SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;
xii. SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'UK');

Changing and adding table attribute

ALTER TABLE Customers ADD Email varchar(255);

ALTER TABLE Customers DROP COLUMN Email;

ALTER TABLE table_name

RENAME COLUMN old_name to new_name;