Where clause in SQL

The SQL WHERE Clause

The WHERE clause is used to filter records.

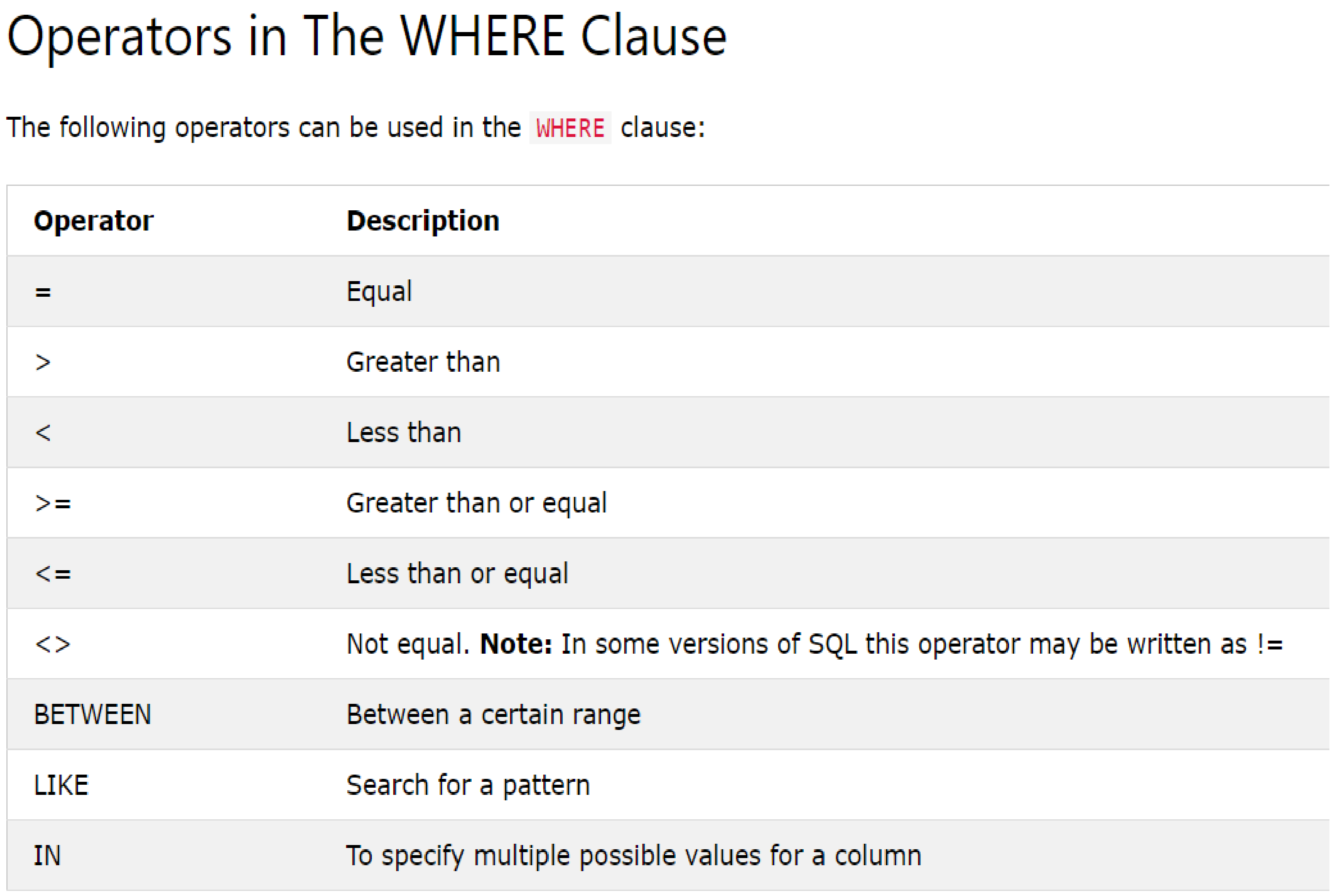
It is used to extract only those records that fulfill a specified condition.

Where clause in SQL

The SQL WHERE Clause

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Where clause in SQL



Operators in The WHERE Clause

The following operators can be used in the `WHERE` clause:

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Where clause in SQL

SQL AND, OR and NOT Operators

The AND and OR operators are used to filter records based on more than one condition:

The AND operator displays a record if all the conditions separated by AND are TRUE.

The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

Where clause in SQL

SQL AND, OR and NOT Operators

```
SELECT * FROM Customers
WHERE Country='Germany' AND City='Berlin’;
SELECT * FROM Customers
WHERE Country='Germany' OR Country='Spain’;
SELECT * FROM Customers
WHERE NOT Country='Germany';
```

Where clause in SQL

SQL AND, OR and NOT Operators

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR
City='München’);
```

Where clause in SQL

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK’);
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

Where clause in SQL

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK’);
Country NOT IN ('Germany', 'France', 'UK');
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

Where clause in SQL

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

The percent sign (%) represents zero, one, or multiple characters

The underscore sign (_) represents one, single character

Where clause in SQL

The SQL LIKE Operator

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%’;
SELECT * FROM Customers
WHERE CustomerName LIKE '%a’;
SELECT * FROM Customers
WHERE CustomerName LIKE '%or%’;
SELECT * FROM Customers
WHERE CustomerName LIKE '_r%’;
SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%';
```

Constraints in SQL

Constraints are the rules that we can apply on the type of data in a table.

That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

Constraints in SQL

The available constraints in SQL are:

- NOT NULL: This constraint tells that we cannot store a null value in a column.

- UNIQUE: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.

- PRIMARY KEY: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.(unique + not null)

Constraints in SQL

- FOREIGN KEY: A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as Foreign key.

- CHECK: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.

- DEFAULT: This constraint specifies a default value for the column when no value is specified by the user.

Constraints in SQL

How to specify constraints?

- We can specify constraints at the time of creating the table using CREATE TABLE statement.
- We can also specify the constraints after creating a table using ALTER TABLE statement.

Constraints in SQL

NOT Null

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

Constraints in SQL

UNIQUE

```
CREATE TABLE Student
(
ID int(6) UNIQUE,
NAME varchar(10),
ADDRESS varchar(20)
);
```

Constraints in SQL

PRIMARY KEY

```
CREATE TABLE Student
(
ID int(6),
NAME varchar(10),
ADDRESS varchar(20),
PRIMARY KEY(ID)
);
```

Constraints in SQL

Check Constraint

```
CREATE TABLE Student
(
ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int CHECK (AGE >= 18)
);
```

Constraints in SQL

Check Constraint

```
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'))
    );
```

Constraints in SQL

Check Constraint

```
CREATE TABLE employees (
   emp_id INT NOT NULL PRIMARY KEY,
   emp_name VARCHAR(55) NOT NULL,
   hire_date DATE NOT NULL,
   salary INT NOT NULL CHECK (salary >= 3000 AND salary <=
10000),
   dept_id INT
);
```

Constraints in SQL

Default Constraint

```
Create Table My_Table1
(
  Id int default(1500),
  Age int,
  Salary int default(100)
);
```

Constraints in SQL

Alter Command

ALTER TABLE Student **MODIFY** s_id int NOT NULL;

ALTER TABLE Student **MODIFY** age INT NOT NULL UNIQUE;

ALTER table Student ADD PRIMARY KEY (s_id);

Constraints in SQL

Alter Command

With alter: Check constraint can also be added to an already created relation using the syntax:

alter table TABLE_NAME modify COLUMN_NAME check(Condition);

Example : alter table dept add check(deptid > 5);

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 (Condition);

alter table TABLE_NAME drop check CHECK_CONST_NAME;

Constraints in SQL

Foreign Key Constraint

Foreign Key is used to relate two tables. The relationship between the two tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

This is also called a referencing key.

We use ALTER statement and ADD statement to specify this constraint.

Constraints in SQL

Customer_Detail Table

c_id	**Customer_Name**	**address**
101	**Adam**	**Noida**
102	**Alex**	**Delhi**
103	**Stuart**	**Rohtak**

Order_Detail Table

Order_id	**Order_Name**	**c_id**
10	Order1	101
11	Order2	103
12	Order3	102

Constraints in SQL

```
CREATE table Order_Detail(
    order_id int PRIMARY KEY,
    order_name varchar(60) NOT NULL,
    c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id)
);
```

Constraints in SQL

```
CREATE table Order_Detail(
    order_id int PRIMARY KEY,
    order_name varchar(60) NOT NULL,
    c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id)
);

ALTER table Order_Detail ADD FOREIGN KEY (c_id)
REFERENCES Customer_Detail(c_id);
```

Constraints in SQL

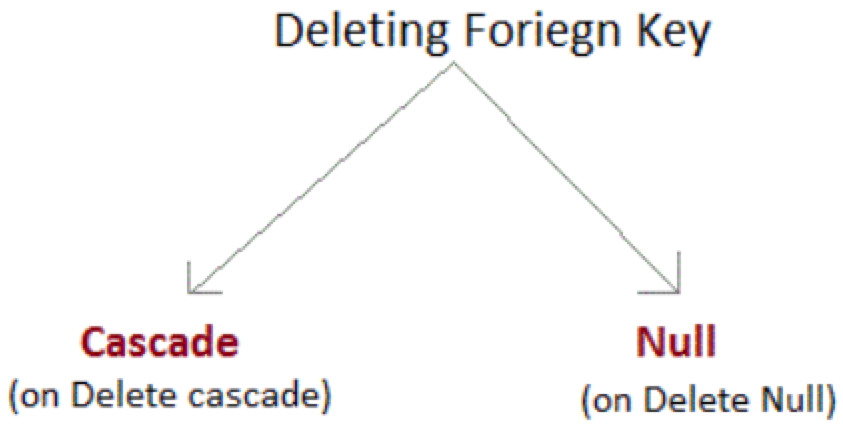
Behaviour of Foriegn Key Column on Delete

There are two ways to maintain the integrity of data in Child table, when a particular record is deleted in the main table.

When two tables are connected with Foriegn key, and certain data in the main table is deleted, for which a record exits in the child table, then we must have some mechanism to save the integrity of data in the child table.

Constraints in SQL

Behaviour of Foriegn Key Column on Delete



Constraints in SQL

Behaviour of Foriegn Key Column on Delete

1. On Delete Cascade : This will remove the record from child table, if that value of foreign key is deleted from the main table.
2. On Delete Null : This will set all the values in that record of child table as NULL, for which the value of foreign key is deleted from the main table.
3. If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

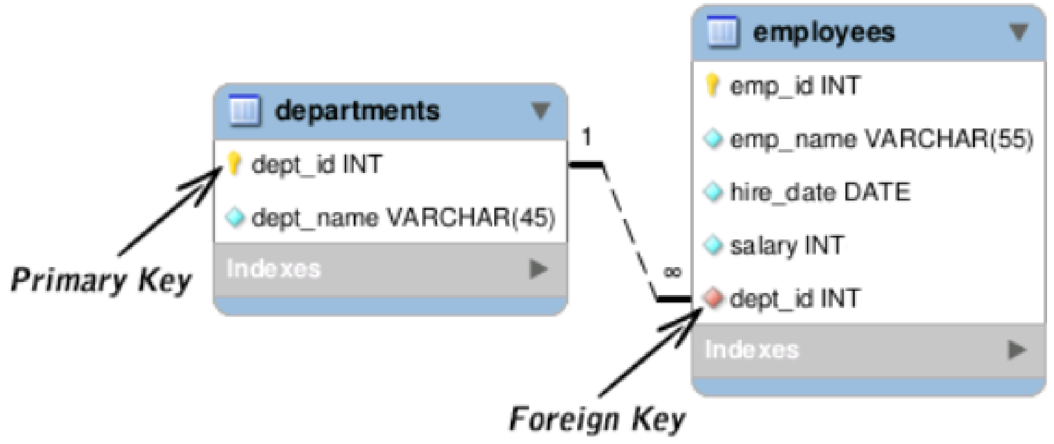
ERROR : Record in child table exist

Constraints in SQL

FOREIGN KEY Constraint

A foreign key (FK) is a column or combination of columns that is used to establish and enforce a relationship between the data in two tables.

Here's a sample diagram showing the relationship between the **employees** and **departments** table. If you look at it carefully, you will notice that the **dept_id** column of the **employees** table matches the primary key column of the **departments** table. Therefore, the **dept_id** column of the **employees** table is the foreign key to the **departments** table.



Constraints in SQL

```
CREATE TABLE employees (
   emp_id INT NOT NULL PRIMARY KEY,
   emp_name VARCHAR(55) NOT NULL,
   hire_date DATE NOT NULL,
   salary INT,
   dept_id INT) ,
   FOREIGN KEY (dept_id) REFERENCES departments(dept_id)
);
```