upGrad

#LifeKoKaroLift

TechM Full Stack Software Development

upGrad

Course: Foundation of Database
Lecture On: Advanced SQL Queries and Normalisation
Instructor: Vishwa Mohan

upGrad

Today's Agenda

1. **Sorting in SQL**

   ORDER BY: ASC/DESC

2. **Operators in SQL**
3. **Searching in SQL**
4. **Aggregation in SQL**
5. **UNION in SQL**
6. **Normalization**

upGrad

Poll 1

On which of the following levels can the ALTER command be applied?

1. Column

2. Table

3. Both

4. None

Poll 1 (Answer)

On which of the following levels can the ALTER command be applied?

1. Column

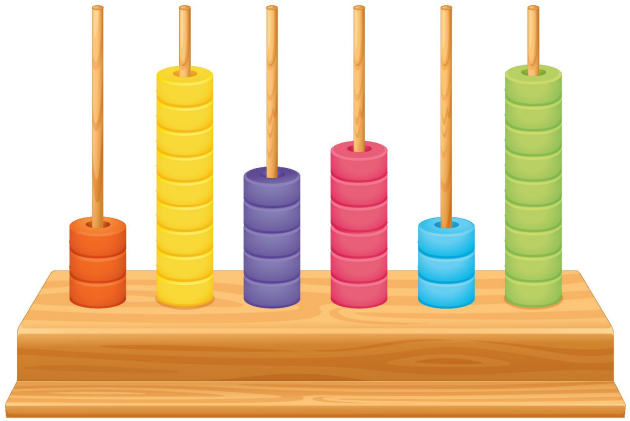
2. Table

3. **Both**

4. None

Sorting

upGrad



Sorting: This involves setting the result set in a specific order, either ascending or descending.

The **ORDER BY** keyword is used for sorting records in either ascending (**ASC**) or descending order (**DESC**).

The following syntax is used for sorting:

```
SELECT column1, column2 FROM table_name
ORDER BY column1, column2;

SELECT column1, column2 FROM table_name
ORDER BY column1, column2  DESC;
```

Sorting is an expensive operation and should be performed only when needed.

Let’s take a look at Example 1.

upGrad

Hands-On Exercise on Sorting (10 min)

Let's perform the following operations on the **movie** table created previously:

1. Add three movies.
2. Sort movies by their release date in ascending order.
3. Sort movies by their release date in descending order.

Poll 2

What is the purpose of the ORDER BY clause?

1. Aggregating records from a table

2. Sorting the results in ascending or descending order using row data

3. Sorting the result set in ascending or descending order using column data

4. All of the above

Poll 2 Answer

What is the purpose of the ORDER BY clause?

1. Aggregating records from a table

2. Sorting the results in ascending or descending order using row data

3. **Sorting the result set in ascending or descending order using column data**

4. All of the above

Operators in SQL

upGrad

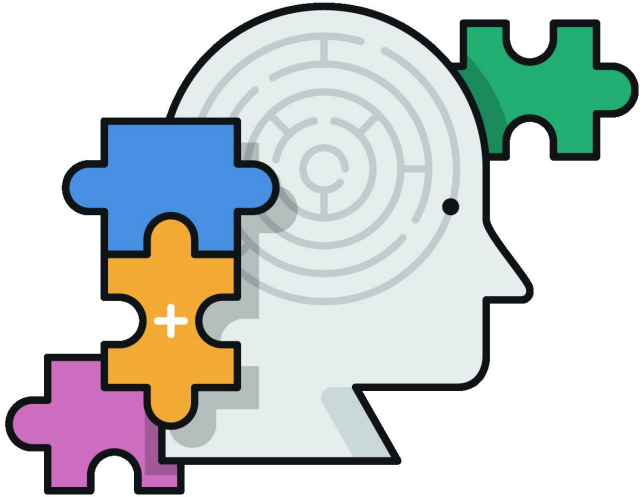
Operators are reserved keywords used to perform various operations such as addition, subtraction, searching etc in SQL.

The following are SQL operators:

1. Arithmetic operators
2. Bitwise operators
3. Comparison operators
4. Logical operators

Logical Operators

upGrad



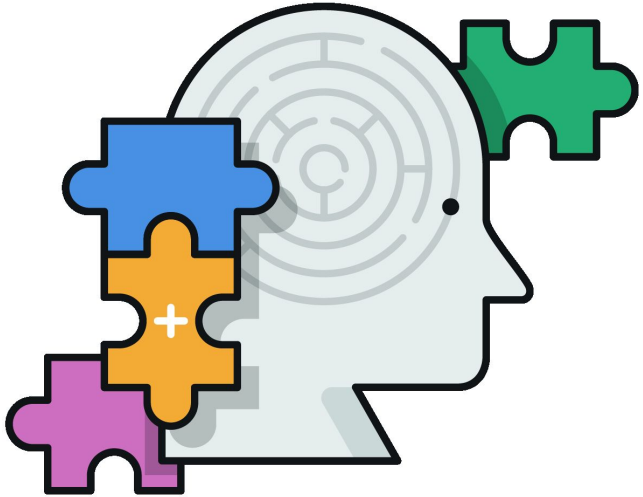
Logical operators are used to compare two conditions simultaneously and give either a **TRUE** result or a **FALSE** one.

The various types of logical operators are as follows:

AND
OR
NOT

Logical Operators

upGrad



Logical operator: AND

The operator AND is TRUE when all the conditions mentioned in the WHERE clause are satisfied.

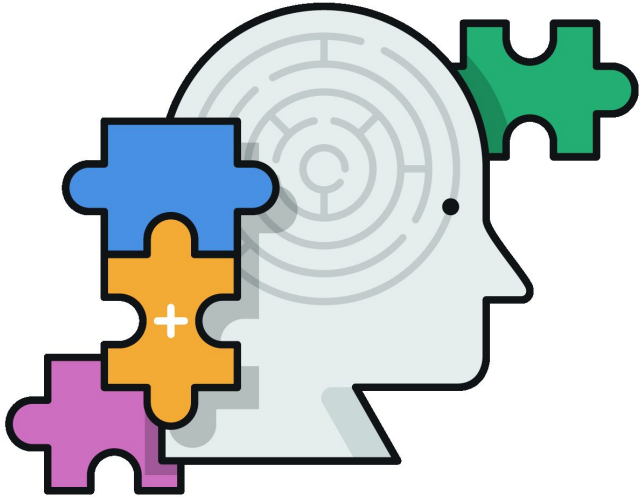
The following syntax is used:

SELECT *column_name*
FROM *table_name*
WHERE condition AND condition;

Let’s take a look at Example 2.

Logical Operators

upGrad



Logical operator: OR

The operator OR is True when any one of the specified conditions of the WHERE Clause is satisfied, i.e., not all the conditions need to be true.

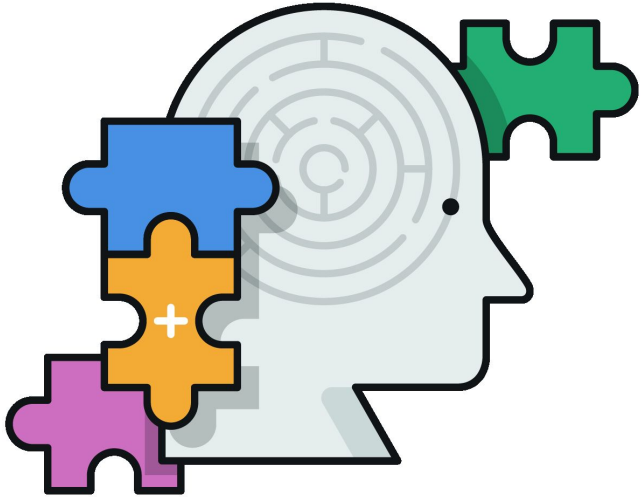
The following syntax is used:

SELECT *column_name*
FROM *table_name*
WHERE condition OR condition;

Continuing Example 2...

Logical Operators

upGrad



Logical operator: NOT

The operator NOT is TRUE when the condition in the WHERE clause is not true.

The following syntax is used:

SELECT *column_name*
FROM *table_name*
WHERE NOT condition;

Continuing Example 2

Hands-On Exercise on Operators (10 min)

upGrad

Perform the following operations on the movie table:

1. Search for a movie with the ID ‘M0003’ that has a release date after 15 February 2020.
2. Search for a movie with either the ID ‘M0003’ or a release date after 15 February 2020.
3. Search for all the movies that are not in Hindi.

Poll 3

What will be the output/result of the following query?

```
SELECT *
FROM employee
WHERE employee_id > 1,234 AND salary > ₹50,000
```

1. Get only employe_id and salary information for all the employees.

2. Get all the information from all the employees whose employee_id is greater than 1,234.

3. Get all the information from all the employees whose employee_id is greater than 1,234 and salary is more than ₹50,000.

4. All of the above

Poll 3 (Answer)

What does the following query mean?

```
SELECT *
FROM employee
WHERE employee_id > 1,234 AND salary > 50,000
```

1. Get only employe_id and salary information for all the employees.

2. Get all the information from all the employees whose employee_id is greater than 1,234.

3. Get all the information from all the employees whose employee_id is greater than 1,234 and salary is more than 50,000.

4. All of the above

Searching

upGrad

Searching: This his refers to searching for a specific record in a database.

Searching is done using **WHERE** clause using operators:

- **IN**
- **NOT IN**
- **LIKE**: *pattern search*
- **BETWEEN**: *range search*

Searching is most commonly performed in a database after insertion of records.

Let’s take a look at Example 3.

Searching

upGrad

Searching: WHERE

WHERE is used to filter records in a database.

The syntax used is as follows:

```
SELECT *
FROM table_name
WHERE condition;
```

The following operators are supported in the WHERE clause:

= , > , < , >= , <= and <>

The WHERE clause can also be used in update and delete statements.

Continuing Example 3...

Searching

upGrad

Searching: WHERE with **IN**

This is used to check whether any record is present in a specified list.

The syntax used is as follows:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Continuing Example 3...

Searching: WHERE with **NOT IN**

This is used to check whether any record is absent from a specified list.

The syntax used is as follows:

```
SELECT column_name(s)
FROM table_name
WHERE column_name NOT IN (value1, value2, ...);
```

Continuing Example 3...

Searching

upGrad

Searching: WHERE with **LIKE**

This is used to check for a specified pattern in the data of a column.

The syntax used is as follows:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

The LIKE operator can be used with the following wildcards:

1. %: Represents zero or more characters
2. _: Represents a single character

Continuing Example 3...

Searching: WHERE with **BETWEEN**

This is used to filter records by specifying a range of values inclusive of the range.

The syntax used is as follows:

```
SELECT column_name1, column_name2
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Continuing Example 3...

upGrad

Hands-On Exercise on Searching (10 min)

Perform the following operations on the movie table:

1. Determine whether or not a movie is present in the table by searching for its name as Oceans 11
2. Search for movies having movie names as Oceans 11 and Gladiator
3. Determine not having movie names Oceans 11 and Gladiator.
4. Find a movie whose name starts with the letter ‘A’.
5. Find a movie whose name has ‘V’ as the second letter.
6. Find all the movies with durations between 90 and 150 minutes.

Poll 4

State whether the following statement is true of false. The DELETE without WHERE clause is similar to the TRUNCATE command.

1. True

2. False

Poll 4 (Answer)

State whether the following statement is true of false. The DELETE without WHERE clause is similar to the TRUNCATE command.

1. True

2. False

Poll 5

What is the purpose of the WHERE clause?

1. It provides the columns that satisfy the condition specified in the clause.
2. It provides the rows that satisfy the condition specified in the clause.
3. It provides the rows and the columns that satisfy the condition specified in the clause
4. None of above

Poll 5 (Answer)

What is the purpose of the WHERE clause?

1. It provides the columns that satisfy the condition specified in the clause.
2. It provide the rows that satisfy the condition specified in the clause.
3. It provides the rows and the columns that satisfy the condition specified in the clause.
4. None of above

Aggregation

upGrad

Aggregation is used to perform a calculation on a set of records to return a single scalar value.

The commonly performed aggregations are as follows:
COUNT: Used to count the number of rows in a specified column or table
MAX: Used to calculate the largest value in a set of values
SUM: Used to calculate the sum of a set of values
MIN: Used to calculate the smallest in a set of values
AVG: Used to calculate the average of a set of values

Aggregation

upGrad

COUNT:

The syntax used is as follows:

```
SELECT COUNT(coulmn_name)
FROM table_name
WHERE condition;
```

MAX:

The syntax used is as follows:

```
SELECT MAX(coulmn_name)
FROM table_name
WHERE condition;
```

SUM:

The syntax used is as follows:

```
SELECT SUM (coulmn_name)
FROM table_name
WHERE condition;
```

Let’s take a look at Example 4.

Aggregation Hands-On Exercise (5 min)

Perform the following operations on the movie table:

1. Find the count of all the upcoming movies.
2. Find the maximum duration amongst all movies present in table.

Poll 6

State whether the following statement is true or false.
TOTAL is a built-in aggregation function.

1. True

2. False

Poll 6 (Answer)

State whether the following statement is true or false.
TOTAL is a built-in aggregation function.

1. True

2. False

Grouping

upGrad

Grouping of records in SQL can be done using the following commands:

1. **GROUP BY:** This command is used to group rows that have same values. It is applied on a SELECT statement to get a set of rows or groups based on values of columns or expressions. Aggregate functions can be applied on these groups.
2. **HAVING:** This command is used to provide conditional checks in the case of aggregate functions.

The following syntax is used:

```
SELECT column_name(s)
FROM table_name
GROUP BY column_name(s) HAVING condition
```

**** Here, the condition can be applied on aggregated functions as well.*

Let’s take a look at Example 5.

Hands-On Exercise on Grouping (5 min)

In the movie table, find the following:

1. The count of movies based on their status
2. The count of movies with the same status having duration of 120

upGrad

Poll 7

Will the following aggregation query work?

```
SELECT first_name, COUNT (DISTINCT last_name)  AS 'Last Name'
FROM user
WHERE user_id >= 5000
GROUP BY last_name;
```

1. Yes

2. No

Poll 7 (Answer)

Will the following aggregation query work?

```
SELECT first_name, COUNT (DISTINCT last_name)
FROM user
WHERE user_id >= 5000
GROUP BY last_name;
```

1. Yes

2. No

WHERE vs HAVING

upGrad

The following table presents the differences between the **WHERE** and **HAVING** clauses**:**

WHERE	HAVING
Applies to rows	Applies to groups
Used before the ***GROUP BY*** clause	Used after the ***GROUP BY*** clause
Cannot be used with aggregation functions	Can be used with aggregation functions

Poll 8

Can you use the HAVING clause without GROUP BY clause to perform aggregation?

1. Yes
2. No

Poll 8 (Answer)

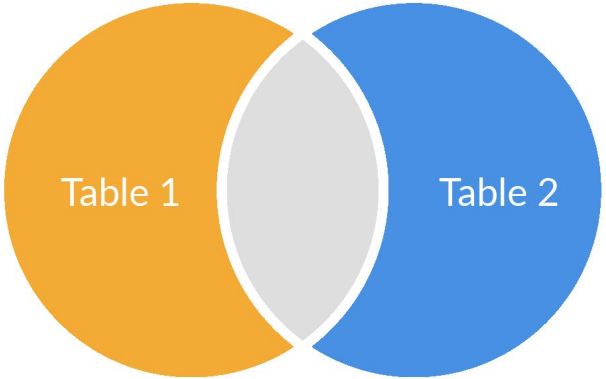
Can you use the HAVING clause without GROUP BY clause to perform aggregation?

1. **Yes**

2. No

UNION

upGrad



UNION

The UNION operator is used to combine the result set of two or more ***SELECT*** statements and does not contain duplicates.

Following are some requirements regarding this operator:

1. Each SELECT statement should have the same number of columns.
2. Columns should have the same data type.
3. Columns should also be in the same order.

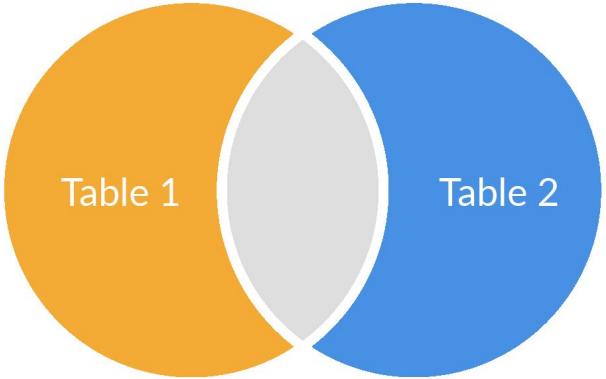
The following syntax is used:

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Let’s take a look at Example 6.

UNION

upGrad



UNION ALL

This operator is used to combine the result set of two or more ***SELECT*** statements. It is the same as the UNION operator but also contains the duplicate values.

The following syntax is used:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

Continuing Example 6...

Hands-On Exercise on UNION (5 min)

upGrad

Find and make a list of all the unique registered mobile numbers in the store using both the *store_customer* and *employee* tables that you just created in example 6.

Poll 9

Which of the following is the UNION operator used to achieve?

1. Combine the records from two or more tables having different columns
2. Combine the records from two or more tables having similar columns
3. None of the above
4. Both 1 and 2

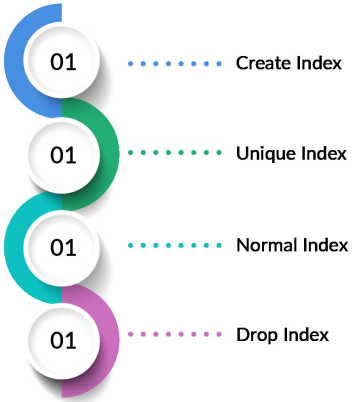
Poll 9 (Answer)

Which of the following is the UNION operator used to achieve?

1. Combine the records from two or more tables having different columns
2. **Combine the records from two or more tables having similar columns**
3. None of the above
4. Both 1 and 2

INDEX

upGrad



An **INDEX** is used to retrieve data from a table.
Indexes are created to make searches faster when the database has millions of records.

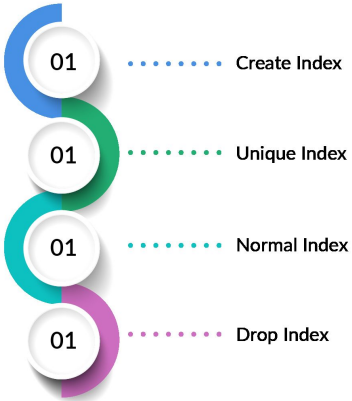
This is similar to using the index in a book to jump to a particular chapter.

Following are the two types of indexes:

a. Normal INDEX: An index in which duplicates are allowed
b. Unique INDEX: An index in which duplicates are not allowed

INDEX

upGrad



CREATE INDEX

Normal index:

The following syntax is used:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

Unique index:

The following syntax is used:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

*** Can be created on either single or multiple columns

DROP an INDEX

The following syntax is used:

```
DROP INDEX index_name;
```

Let’s take a look at Example 7.

upGrad

Hands-On Exercise on Index (5 min)

Create the following indexes on the movie table:

1. Create a normal index for the release date of a movie.
2. Create a unique index for the movie names.
3. Drop both the indexes.
4. Create a composite index for the movie names and the release dates.

upGrad

Poll 10

Which of the following can you achieve using indexing?

1. Improvement in the write speed

2. Improvement in the read speed

3. Both 1 and 2

4. None of the above

Poll 10 (Answer)

Which of the following can you achieve using indexing?

1. Improvement in the write speed

2. Improvement in the read speed

3. Both 1 and 2

4. None of the above

Data Anomalies

upGrad

Data anomalies occur in database tables if they are constructed poorly.

Let’s take a look at some anomalies that may occur in the following table.

car_customers

	CUSTOM...	CUSTOMER_NAME	CAR_NUMBER_PLATE	CAR_NAME	DATE_TRANSACTION	OWNER_ID	OWNER_NAME
1	C1001	Sachin	car1234,car7658	swift,Thar	12/01/2020,18/01/2020	CU001,CU009	Dev,Irfan
2	C1002	Rahul	car3421,car6534,car3789	Baleno,Honda,Swift	12/01/2020,14/01/2020,15/01/2020	CU002,CU004	Rohit,Shikhar
3	C1003	Vishwa	car1200,car7865	Creta,Nano	11/04/2020,18/05/2019	CU003,CU005	Aporv,Heena
4	C1002	Ram	car3001,car6004,car3709	i20,Figo,Fortuner	10/04/2020,17/01/2020,13/03/2020	CU006,CU007	Rohit,Shikhar

Types of Anomalies

upGrad

The three types of anomalies that can occur in database tables are as follows:

1. **Insertion anomaly**: This occurs when data insertion is not possible owing to the prerequisite data not being present.
2. **Updation anomaly**: This occurs when an update in one data leads to updates in multiple data, thereby leading to data inconsistency.
3. **Deletion anomaly**: This occurs when deletion of one data leads to multiple deletions and loss of information.

Anomalies are reduced by the process of normalisation.

Let’s take a look at Example 8.

Normalisation

upGrad

Data Normalisation

Data normalisation is the process of organising data in a database such that a piece of data is not repeated at various tables.

The data becomes consistent because it is not repeated in different rows of the same table. Information about one business entity is stored in one table.

Different business entities are stored in different tables.

- The 1NF step removes multiple values from one field of the table.
- The 2NF step removes partial functional dependencies from the table.
- The 3NF step removes transitive dependencies from the table.

upGrad

Following are the criteria for a table to be in 1NF:

It should not have any multi-valued field. If you store multiple values in one field and separate them by commas, then the relational database (RDBMS) will consider the entire line to be one string, not a list. Hence, the application written to retrieve the data from the table will have to convert this string to a list in order to access each data element separately. This is very unreliable.

Continuing Example 8...

Hands-on Exercises: Stub for Normalization

upGrad

Run this query!!

```
CREATE TABLE car_customer
(
   customer_id CHAR(5) NOT NULL,
   customer_name VARCHAR(30),
   car_number_plate VARCHAR(50),
   car_name VARCHAR(50),
   transaction_date VARCHAR(50),
   owner_id VARCHAR(20),
   owner_name VARCHAR(50)
);


INSERT INTO car_customer
VALUES ('C1001','Sachin','car1234,car7658','swift,Thar','12/01/2020,18/01/2020','CU001,CU009','Dev,Irfan');

INSERT INTO car_customer
VALUES
('C1002','Rahul','car3421,car6534,car3789','Baleno,Honda,Swift','12/01/2020,14/01/2020,15/01/2020','CU002,CU004
','Rohit,Shikhar');

INSERT INTO car_customer
VALUES ('C1003','Vishwa','car1200,car7865','Creta,Nano','11/04/2020,18/05/2019','CU003,CU005','Aporv,Heena');

INSERT INTO car_customer
VALUES
('C1002','Ram','car3001,car6004,car3709','i20,Figo,Fortuner','10/04/2020,17/01/2020,13/03/2020','CU006,CU007','
Rohit,Shikhar');



SELECT * FROM car_customer;
```

Hands-On Exercise on 1NF (10 min)

Convert the car_customer table into 1NF.

Following are the criteria for a table to be in 2NF:

1. It must be in 1NF first.
2. All the non-prime attributes must be fully functionally dependent on the composite key or the primary key of the table so that there is no partial dependency on the composite key of the table.

Continuing Example 8...

Hands-On Exercise on 2NF (10 min)

upGrad

Now that you have learnt about 2NF, convert the car_customer table to 2NF.

upGrad

Following are the criteria for a table to be in 3NF:

1. It must be in 2NF first.
2. There must be no indirect relationship between the primary key(s) and non-prime attributes in a table that results in a transitive dependency.

Continuing Example 8...

Hands-On Exercise on 3NF (10 min)

Now that you have learnt about 3NF, convert the car_customer table to 3NF.

Poll 11

Why is normalization used?

1. To avoid redundancy
2. To avoid deletion anomaly
3. To avoid updation anomaly
4. All of the above

Poll 11 (Answer)

Why is NORMALISATION used?

1. To avoid redundancy
2. To avoid deletion anomaly
3. To avoid updation anomaly
4. **All of the above**

upGrad
DISCUSSION TIME

upGrad

#LifeKoKaroLift

Thank you!