CSE 4308 - DBMS LAB

DBMS LAB 04 MATERIAL

Prepared by:
Mohammad Anas Jawad
Lecturer, IUT CSE

Department of Computer Science and Engineering
Islamic University of Technology
May 30, 2021

Contents

Note: Majority of this material has been sourced from w3schools. Take a look at `https://www.w3schools.com/sql/` for more detailed examples on these topics.

1 SQL STATEMENTS

1.1 The WHERE Clause

The WHERE clause is used to specify conditions on the tuples we want to be displayed. The where clause allows us to select only those rows in the result relation of the from clause that satisfy a specified predicate.

The basic syntax involving the WHERE clause is as follows:

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

SQL allows us to specify multiple conditions as well by using logical operators. For example, the query "Find the names of all instructors in the Computer Science department who have salary greater than $70,000." can be written as:

```
SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.' AND salary > 70000;
```

A list of operators supported in SQL can be found in the following picture:

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

Figure 1: Operators in The WHERE Clause [Source: w3schools]

1.1.1 AND, OR and NOT

The WHERE clause can be combined with 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.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;

SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;

SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

1.1.2 The IN Operator

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

The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
--Example--
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

It is also possible to imply conditions that rely on other relations by using sub-queries inside the WHERE clause.

For example, the following SQL statement selects all customers that are from the same countries as the suppliers:

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

1.1.3 The BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;

--Example--

SELECT * FROM Products
```

```
WHERE Price NOT BETWEEN 10 AND 20;

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND NOT CategoryID IN (1,2,3);

SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

1.1.4 NATURAL JOIN

Unlike the Cartesian product of two relations, which concatenates each tuple of the first relation with every tuple of the second, natural join considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relations.

inst_ID	*name*	*dept_name*	*salary*	*teaches_ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Physics	95000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Physics	95000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Physics	95000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Physics	95000	10101	FIN-201	1	Spring	2010
10101	Srinivasan	Physics	95000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Physics	95000	22222	PHY-101	1	Fall	2009

Figure 2: Cartesian Product between instructor and teaches relation

ID	name	dept_name	salary	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Figure 3: Natural join between instructor and teaches relation

```
select name, course id
from instructor, teaches
where instructor.ID = teaches.ID;

--Equivalent to--

select name, course id
from instructor natural join teaches;
```

1.2 The ORDER BY Clause

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

If you do not specify the sorting order, the ORDER BY keyword sorts the records in ascending order by default. Otherwise, to specify ascending or descending order of sorting, we use the keywords 'ASC' and 'DESC' respectively.

```
--Ascending--
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC;
--Descending--
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... DESC;

--Example--

SELECT * FROM Customers
ORDER BY Country, CustomerName;

SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

1.3 The UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

--Example--

UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Warning: Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

1.4 The DELETE Statement

The DELETE statement is used to delete existing records in a table.

```
DELETE FROM table_name WHERE condition;

--Example--

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

Warning: Similar to the UPDATE statement, not specifying a WHERE clause will result in deleting all the rows.

1.5 SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;

--Example--

SELECT DISTINCT Country FROM Customers;
```

1.6 Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of the query.

```
SELECT column_name AS alias_name
FROM table_name;


SELECT column_name(s)
FROM table_name AS alias_name;
```