CSE 4308 - DBMS LAB

DBMS LAB 06 MATERIAL

Prepared by:
Mohammad Anas Jawad
Lecturer, IUT CSE

Department of Computer Science and Engineering
Islamic University of Technology
July 4, 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 LIKE Operator

In simple terms, the LIKE operator is used to match substrings in a query. It 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 represents a single character.

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Examples showing the use of LIKE operator with % and _:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

Figure 1: Examples of using the LIKE operator [Source: w3schools]

1.2 Wildcards

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the `LIKE` operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Some of the wildcards recognized by SQL are as follows:

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt

Figure 2: Wildcards in SQL [Source: w3schools]

1.3 Aggregate Functions

Aggregate functions are functions that take a collection (a set or multiset) of values as input and return a single value. SQL offers five built-in aggregate functions:

- Average: avg
- Minimum: min
- Maximum: max
- Total: sum
- Count: count

1.3.1 AVG function

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
--Example
SELECT AVG(Price)
FROM Products;
```

1.3.2 MIN function

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
--Example
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

1.3.3 MAX function

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
--Example
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

1.3.4 SUM function

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
--Example
SELECT SUM(Quantity)
FROM OrderDetails;
```

1.3.5 COUNT function

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
--Example
SELECT COUNT(ProductID)
FROM Products;
```

1.4 GROUP BY

There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples; we specify this wish in SQL using the **GROUP BY** clause.

The attribute or attributes given in the group by clause are used to form groups.

```
SELECT DEPT_NAME, AVG(SALARY) AS AVG_SALARY
FROM INSTRUCTOR
GROUP BY DEPT_NAME;
```

Note: Attributes that appear in the `SELECT` statement but not in the aggregate function must appear in the `GROUP BY` clause, otherwise the query will be considered erroneous. For example, the following query is incorrect as ID appears in the `SELECT` statement, but not in the `GROUP BY` clause:

```
SELECT DEPT_NAME, ID, AVG(SALARY) AS AVG_SALARY
FROM INSTRUCTOR
GROUP BY DEPT_NAME;
```

1.5 HAVING

The HAVING clause is used to specify conditions on groups rather than on tuples.

For example, we might be interested in only those departments where the average salary of the instructors is more than $42,000. We cannot use the WHERE clause in this case because WHERE is used to specify conditions on a single set of tuples, whereas in this case, we are interested to find a group of departments where the average salary is more than $42,000.

```
SELECT DEPT_NAME, AVG(SALARY) AS AVG_SALARY
FROM INSTRUCTOR
GROUP BY DEPT_NAME
HAVING AVG(SALARY)>42000;
```

Note: As was the case for the SELECT statement, any attribute that is present in the HAVING clause without being aggregated must appear in the GROUP BY clause, otherwise the query is treated as erroneous.

2 POINTS TO REMEMBER

The sequence of operations whenever multiple of these clauses are involved is:

FROM -> WHERE -> GROUP BY -> AGGREGATE FUNCTION -> HAVING -> ORDER BY

1. As was the case for queries without aggregation, the *from* clause is first evaluated to get a relation.

2. If a *where* clause is present, the predicate in the *where* clause is applied on the result relation of the from clause.

3. Tuples satisfying the *where* predicate are then placed into groups by the *group by* clause if it is present. If the *group by* clause is absent, the entire set of tuples satisfying the *where* predicate is treated as being in one group.

4. The *having* clause, if it is present, is applied to each group; the groups that do not satisfy the *having* clause predicate are removed.

5. The *select* clause uses the remaining groups to generate tuples of the result of the query, applying the aggregate functions to get a single result tuple for each group.