Theory: SELECT FROM statement

🕓 26 minutes 0 / 5 problems solved

Skip this topic Start practicing

3257 users solved this topic. Latest completion was about 3 hours ago.

You already know that SQL is designed to handle data structured as tables, which comes in handy in various application areas. You also know that to extract all the data from a table called "table_name", you should use the following query:

```
SELECT * FROM table_name;
```

In this topic, we'll learn more about the `SELECT` statement and how to extract preprocessed data from the table.

§1. Projection

Assume you have a table "weather" that stores information about the weather in London for the past 5 days.

Table "weather"

day	hour	temperature	feels_like	pressure	...	wind_speed	wind_direction	wind_gusts	humidity	phenomena
1	1	14	14	754	...	2	W	8	88	rain shower
1	7	11	11	754	...	1	N	1	92	continuous rain
...	...	...	...	...	...	...	...	...	...	...
5	19	16	13	759	...	5	S	11	93	rain shower

As you can see, there are a lot of attributes for every hour. Do we need all of them? Let's write a query that selects only the basic info to be displayed on the mobile phone screen, for example, day, hour, temperature, and wind speed.

```
SELECT
    day,
    hour,
    phenomena,
    temperature as "temperature in Celsius",
    feels_like as "feels like in Celsius",
    wind_speed as "wind speed in m/s"
FROM
    weather

;
```

After the `SELECT` keyword, we list the columns that we want to select and specify aliases where needed. The query evaluation results in the following table:

day	hour	phenomena	temperature in Celsius	feels like in Celsius	wind speed in m/s
1	1	rain shower	14	14	2
1	7	continuous rain	11	11	1
...	...	...	...	...	...
5	19	rain shower	16	13	5

The type of data extraction when you select a subset of table columns is called **projection.**

Current topic:

SELECT FROM statement ···

Topic depends on:

✕ Basic SELECT statement ···

Topic is required for:

SELECT FROM WHERE statement ···

Results ordering ···

JDBC Statements ···

Set operations ···

Table of contents:

Here's a general schema for such queries: keyword `SELECT`, list of column names with optional aliases, keyword `FROM`, table name, and a semicolon to mark the end of the statement:

```
SELECT
    col1 [AS alias1], ..., colN [AS aliasN]
FROM
    table_name
;
```

§2. Expressions

Now, let's imagine that for some reason we need to have different results based on the same data, for example, add columns that state the place, show the temperature in Fahrenheit, and estimate whether it feels colder than that.

The query below does this easily:

```
SELECT
    'London' as place,
    day,
    hour,
    phenomena,
    temperature*9/5+32 as "temperature in Fahrenheit",
    feels_like < temperature as "feels colder",
    wind_speed as "wind speed in m/s"
FROM
    weather
;
```

We specified the corresponding expressions for attributes "place", "temperature in Fahrenheit", and "feels colder" based on literals and column names. Yes, you can use column names in expressions as well! When the data management system executes your query, it will substitute the column names with the corresponding attribute value for each processed row.

§3. Logical table

A data management system hides the way your data is physically stored behind an abstract concept of a **logical table**. All you need to know to be able to run a query is the **database schema**—table names, column names and types—and appropriate access permissions. Internally, the query processor maps table and column references from queries to physical data such as files, network connections, or even the results of executing other queries.

§4. Conclusion

Let's take a final look at the overall template for statements that extract data from a table and evaluate expressions in it: keyword `SELECT`, list of expressions with optional aliases, keyword `FROM`, table name, and a semicolon to mark the end of the statement.

```
SELECT
    exp1 [AS alias1], ..., expN [AS aliasN]
FROM
    table_name
;
```

Report a typo

293 users liked this theory. **4** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion