CHAPTER 5: DATABASES AND SQL

Subject: Introduction to data science

Instructor: Đinh Xuân Trường
Posts and Telecommunications Institute of Technology
Faculty of Information Technology 1

Hanoi, 2024
http://www.ptit.edu.vn

What is a database?

* A database is a system of structured information stored on external memory, to satisfy the simultaneous information exploitation requirements of many users or users. Many application programs with many different purposes.
* Database includes various types of data: audio, text, images,... encoded and stored as specific files.

Database classification

- *File format database*: data is stored in the form of files that can be text, ascii, .dbf. A typical file database is **.mdb Foxpro**.
- Object-oriented database: data is also stored in data tables, but the tables have additional object-oriented features such as storing additional behaviors, to express Shows the object's behavior.
- Semi-structured database: data is saved as XML.

Database Roles

In the current age of 4.0 technology, databases and the database building process play an important role for every organization/business.

- Stores information systematically
- Improve data security
- Allows data retrieval from multiple users
- Easier data management

What is Structured Query Language (SQL)?

* SQL is a programming language that serves to store and process information in relational databases.
* Relational databases store information as tables with rows and columns that represent data attributes and various relationships between data values.
* Can use SQL statements to store, update, drop, search and retrieve information from the database, maintain and optimize database performance Whether.

Nội dung chính

Some basic statements

∗ CREATE TABLE: create a new table in the database

∗ INSERT INTO: used to insert new records in a table

∗ UPDATE: Modify and update data in the database.

∗ DELETE: Delete data from the database.

∗ SELECT: used to determine display columns.

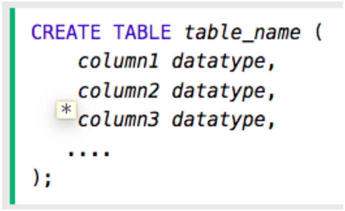
∗ GROUP BY: used to create data groupings on 1 column or multiple columns.

∗ ORDER BY: used to sort data.

Nội dung chính

SQL CREATE TABLE Statement

* The CREATE TABLE statement is used to create a new table in a database.
* Syntax



```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Nội dung chính

SQL INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- INSERT INTO Syntax:
  - Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

  - If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Nội dung chính

SQL UPDATE Statement

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

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

Nội dung chính

SQL DELETE Statement

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

```
DELETE FROM table_name WHERE condition;
```

Nội dung chính

SQL SELECT Statement

* The SELECT statement is used to select data from a database.
* Example: Return data from the Customers table

```
SELECT CustomerName, City FROM Customers;
```

Nội dung chính

SQL GROUP BY Statement

* TThe GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
* The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.
* GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Nội dung chính

SQL ORDER BY Keyword

* The ORDER BY keyword is used to sort the result-set in ascending or descending order.
* Example: Sort the products by price:

```
SELECT * FROM Products
ORDER BY Price;
```

Nội dung chính

Nội dung chính

SQL Subqueries

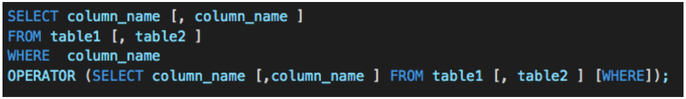
* An SQL Subquery, is a SELECT query within another query. It is also known as Inner query or Nested query and the query containing it is the outer query.
* The outer query can contain the SELECT, INSERT, UPDATE, and DELETE statements. We can use the subquery as a column expression, as a condition in SQL clauses, and with operators like =, >, <, >=, <=, IN, BETWEEN, etc.
* Rules to be followed
  - Subqueries must be enclosed within parentheses.
  - Subqueries can be nested within another subquery.
  - A subquery must contain the SELECT query and the FROM clause always.

SQL Subqueries

* Rules to be followed (continous)
  - A subquery consists of all the clauses an ordinary SELECT clause can contain: GROUP BY, WHERE, HAVING, DISTINCT, TOP/LIMIT, etc. However, an ORDER BY clause is only used when a TOP clause is specified. It can't include COMPUTE or FOR BROWSE clause.
  - A subquery can return a single value, a single row, a single column, or a whole table. They are called scalar subqueries.

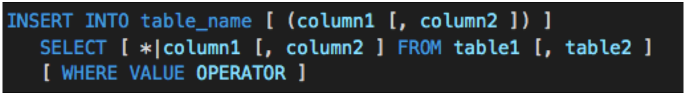
Syntax of SQL Subqueries

∗ Subqueries with the SELECT Statement



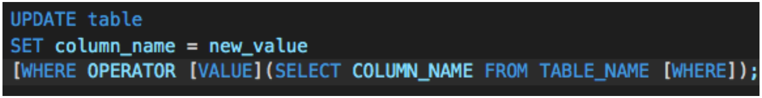
```
SELECT column_name [, column_name ]
FROM table1 [, table2 ]
WHERE  column_name
OPERATOR (SELECT column_name [,column_name ] FROM table1 [, table2 ] [WHERE]);
```

∗ Subqueries with the INSERT Statement



```
INSERT INTO table_name [ (column1 [, column2 ]) ]
   SELECT [ *|column1 [, column2 ] FROM table1 [, table2 ]
   [ WHERE VALUE OPERATOR ]
```

∗ Subqueries with the UPDATE Statement



```
UPDATE table
SET column_name = new_value
[WHERE OPERATOR [VALUE](SELECT COLUMN_NAME FROM TABLE_NAME [WHERE]);
```

Nội dung chính

SQL query optimization basics

* Query optimization is a process of defining the most efficient and optimal way and techniques that can be used to improve query performance based on the rational use of system resources and performance metrics.
* The purpose of query tuning is to find a way to decrease the response time of the query, prevent the excessive consumption of resources, and identify poor query performance.
* In the context of query optimization, query processing identifies how to faster retrieve data from SQL Server by analyzing the execution steps of the query, optimization techniques, and other information about the query.

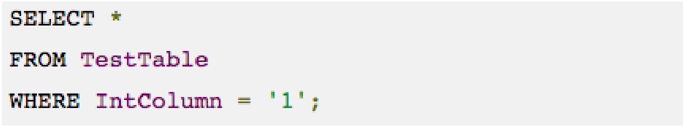
Some Query optimization tips for better performance

Monitoring metrics can evaluate query runtime, detect performance pitfalls, and show how they can be improved.

- Execution plan: A SQL Server query optimizer executes the query step by step, scans indexes to retrieve data, and provides a detailed overview of metrics during query execution.
- Input/Output statistics: Used to identify the number of logical and physical reading operations during the query execution that helps users detect cache/memory capacity issues.
- Buffer cache: Used to reduce memory usage on the server.
- Latency: Used to analyze the duration of queries or operations.
- Indexes: Used to accelerate reading operations on the SQL Server.
- Memory-optimized tables: Used to store table data in memory to make reading and writing operations run faster.

Tip 1: Check for unused indexes

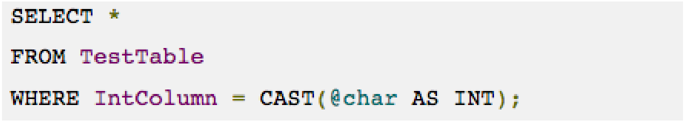
* You may encounter a situation where indexes exist but are not being used. One of the reasons for that might be implicit data type conversion. Let's consider the following query:



```
SELECT *
FROM TestTable
WHERE IntColumn = '1';
```

Tip 1: Check for unused indexes

- When executing this query, SQL Server will perform implicit data type conversion, i.e. convert int data to varchar and run the comparison only after that. In this case, indexes won't be used. How can you avoid this? We recommend using the CAST() function that converts a value of any type into a specified datatype. Look at the query below.



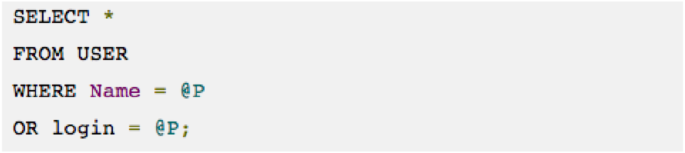
```
SELECT *

FROM TestTable

WHERE IntColumn = CAST(@char AS INT);
```

Tip 2: Avoid using multiple OR in the FILTER predicate

* When you need to combine two or more conditions, it is recommended to eliminate the usage of the OR operator or split the query into parts separating search expressions. SQL Server can not process OR within one operation. Instead, it evaluates each component of the OR which, in turn, may lead to poor performance.



```
SELECT *
FROM USER
WHERE Name = @P
OR login = @P;
```

Tip 2: Avoid using multiple OR in the FILTER predicate

* If we split this query into two SELECT queries and combine them by using the UNION operator, SQL Server will be able to make use of the indexes, and the query will be optimized.

```
SELECT * FROM USER
WHERE Name = @P
UNION
SELECT * FROM USER
WHERE login = @P;
```

Tip 3: Avoid using SELECT DISTINCT

* The SQL DISTINCT operator is used to select only unique values of the column and thus eliminate duplicated values. It has the following syntax:

```
SELECT DISTINCT column_name FROM table_name;
```

Tip 4: Use SELECT fields instead of SELECT*

- The SELECT statement is used to retrieve data from the database. In the case of large databases, it is not recommended to retrieve all data because this will take more resources on querying a huge volume of data.
- If we execute the following query, we will retrieve all data from the Users table, including, for example, users' avatar pictures. The result table will contain lots of data and will take too much memory and CPU usage.

```
SELECT
   *
FROM Users;
```

Nội dung chính

What is a NoSQL database?

- NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables.
- NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph.
- They provide flexible schemas and scale easily with large amounts of data and high user loads.

What is a NoSQL database?

Over time, four major types of NoSQL databases emerged:

- *Document databases* store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects.
- *Key-value databases* are a simpler type of database where each item contains keys and values.
- *Wide-column stores* store data in tables, rows, and dynamic columns.
- *Graph databases* store data in nodes and edges. Nodes typically store information about people, places, and things, while edges store information about the relationships between the nodes.

When should NoSQL be used?

When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:

* Fast-paced Agile development
* Storage of structured and semi-structured data
* Huge volumes of data
* Requirements for scale-out architecture
* Modern application paradigms like microservices and real-time streaming