NHẬP MÔN KHOA HỌC DỮ LIỆU

CƠ SỞ DỮ LIỆU VÀ SQL

ThS. Vũ Hoài Thư

Cơ sở dữ liệu là gì?

- Cơ sở dữ liệu (database) là một hệ thống các thông tin có cấu trúc được lưu trữ trên bộ nhớ ngoài, nhằm thoả mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với nhiều mục đích khác nhau.
- CSDL bao gồm các loại dữ liệu: âm thanh, văn bản, hình ảnh,. . . được mã hóa và lưu trữ dưới dạng file cụ thể.

Phân loại cơ sở dữ liệu

- Cơ sở dữ liệu dạng file: dữ liệu được lưu trữ dưới dạng các file có thể là text, ascii, .dbf. Tiêu biểu cho cơ sở dữ liệu dạng file là.mdb Foxpro.
- Cơ sở dữ liệu hướng đối tượng: dữ liệu cũng được lưu trữ trong các bảng dữ liệu nhưng các bảng có bổ sung thêm các tính năng hướng đối tượng như lưu trữ thêm các hành vi, nhằm thể hiện hành vi của đối tượng.
- Cơ sở dữ liệu bán cấu trúc: dữ liệu được lưu dưới dạng XML.

Vai trò Cơ sở dữ liệu

Trong thời đại công nghệ 4.0 hiện nay, database và quá trình xây dựng cơ sở dữ liệu đóng vai trò quan trọng với mỗi tổ chức/doanh nghiệp.

- Lưu trữ thông tin có hệ thống
- Nâng cao tính bảo mật dữ liệu
- Cho phép truy xuất dữ liệu từ nhiều user
- Quản lý dữ liệu dễ dàng hơn

Ngôn ngữ truy vấn có cấu trúc (Structured Query Language - SQL) là gì?

- SQL là một ngôn ngữ lập trình phục vụ việc lưu trữ và xử lý thông tin trong cơ sở dữ liệu quan hệ.
- Cơ sở dữ liệu quan hệ lưu trữ thông tin dưới dạng bảng có các hàng và cột đại diện cho những thuộc tính dữ liệu và nhiều mối quan hệ khác nhau giữa các giá trị dữ liệu.
- Có thể sử dụng các câu lệnh SQL để lưu trữ, cập nhật, loại bỏ, tìm kiếm và truy xuất thông tin từ cơ sở dữ liệu, duy trì và tối ưu hóa hiệu suất cơ sở dữ liệu.

Một số câu lệnh cơ bản

- CREATE TABLE: tạo một bảng mới trong csdl
- INSERT: Chèn dữ liệu mới thêm vào CSDL
- UPDATE: Sửa đổi, cập nhật dữ liệu trong cơ sở dữ liệu.
- DELETE: Xóa dữ liệu từ cơ sở dữ liệu.
- SELECT: dùng để xác định các cột hiển thị.
- GROUP BY: dùng để tạo các phân nhóm dữ liệu trên 1 cột hoặc nhiều cột.
- ORDER BY: dùng để sắp xếp dữ liệu.

CREATE TABLE

- Lệnh CREATE TABLE dùng để tạo cấu trúc bảng tạm cục bộ (Local Temporary Table) trong CSDL. Chỉ sử dụng trong phạm vi khai báo, ví dụ một kết nối người dùng, một thủ tục.
- Lệnh CREATE TABLE dùng để tạo cấu trúc bảng tạm toàn cục (Global Temporary Table). Sử dụng trong tất cả kết nối người dùng.
- Bảng tạm (Temporary Table) để lưu tạm các dữ liệu quan hệ, được tạo ra trong CSDL tempdb.

Cú pháp CREATE TABLE

- Tạo bảng tạm cục bộ (1 dấu trước tên bảng)

```
CREATE TABLE #Tên_bảng
(
    Tên_cột_1 Kiểu_dữ_liệu,
    Tên_cột_2 Kiểu_dữ_liệu,
    ...
    Tên_cột_n Kiểu_dữ_liệu
)
```

- Tạo bảng tạm toàn cục (2 dấu trước tên bảng)

```
CREATE TABLE ##Tên_bảng
(
    Tên_cột_1 Kiểu_dữ_liệu,
    Tên_cột_2 Kiểu_dữ_liệu,
    ...
    Tên_cột_n Kiểu_dữ_liệu
)
```

INSERT

- Lệnh INSERT trong SQL được dùng để thêm một hàng dữ liệu mới vào bảng trong cơ sở dữ liệu.
- Cú pháp của lệnh INSERT:

```
1 INSERT Tên_bảng(Cột_1, Cột_2,... Cột_n)
2 VALUES(Giá_trị_1, Giá_trị_2,... Giá_trị_n)
3 --Hoặc (có thể thêm INTO):
4 INSERT INTO Tên_bảng(Cột_1, Cột_2,... Cột_n)
5 VALUES(Giá_trị_1, Giá_trị_2,... Giá_trị_n)
```

UPDATE

- Lệnh UPDATE dùng để sửa dữ liệu hiện có trong bảng. Có thể sửa nhiều dòng tại một thời điểm.
- Cú pháp của lệnh UPDATE:
  - Điều kiện từ 1 bảng

```
UPDATE Tên_bảng
SET Cột_1 = Giá_trị_1, Cột_2 = Giá_trị_2,... Cột_n = Giá_trị_n
WHERE Điều_kiện
```

  - Điều kiện từ nhiều bảng

```
UPDATE Tên_bảng
    SET Tên_cột = Biểu_thức,...
FROM Tên_bảng_1
    INNER|LEFT|RIGHT JOIN Tên_bảng_2 ON Biểu_thức_liên_kết
WHERE Điều_kiện_sửa_đổi
```

DELETE

- Lệnh DELETE dùng để xóa dữ liệu đang tồn tại trong bảng
- Cú pháp của lệnh DELETE:
  - Điều kiện từ 1 bảng

```
DELETE Tên_bảng
WHERE Điều_kiện
--Hoặc (có thể thêm FROM):
DELETE FROM Tên_bảng
WHERE Điều_kiện
```

  - Điều kiện từ nhiều bảng

```
DELETE Tên_bảng
FROM Tên_bảng_1
    INNER|LEFT|RIGHT JOIN Tên_bảng_2 ON Biểu_thức_liên_kết
WHERE Điều_kiện_xóa_dữ_liệu
--Hoặc (có thể thêm FROM):
DELETE FROM Tên_bảng
FROM Tên_bảng_1
    INNER|LEFT|RIGHT JOIN Tên_bảng_2 ON Biểu_thức_liên_kết
WHERE Điều_kiện_xóa_dữ_liệu
```

SELECT

- Lệnh SELECT dùng để xác định các cột hiển thị.
- Cú pháp của lệnh SELECT:

```
1 SELECT Tên_cột_1, Tên_cột_1,... Tên_cột_n
2 FROM Tên_bảng
```

GROUP BY

- Mệnh đề GROUP BY dùng để tạo các phân nhóm dữ liệu trên 1 cột hoặc nhiều cột.
- Cú pháp của mệnh đề GROUP BY:

```
1 SELECT Tên_cột, Hàm_nhóm_dữ_liệu
2 FROM Tên_bảng
3 WHERE Điều_kiện
4 GROUP BY Biểu_thức_nhóm_dữ_liệu
5 HAVING Điều_kiện_lọc_nhóm_dữ_liệu
6 ORDER BY Tên_cột
```

ORDER BY

- Mệnh đề ORDER BY dùng để sắp xếp dữ liệu. ASC cho thứ tự tăng dần (mặc định). DESC cho thứ tự giảm dần.
- Cú pháp của mệnh đề ORDER BY

```
1  --Sắp xếp tăng dần:
2  SELECT Tên_cột
3  FROM Tên_bảng
4  ORDER BY Tên_cột ASC
5
6  --Sắp xếp giảm dần:
7  SELECT Tên_cột
8  FROM Tên_bảng
9  ORDER BY Tên_cột DESC
```

Truy vấn con

- Trong SQL Server, truy vấn con là một truy vấn nằm trong một truy vấn khác. Có thể tạo các truy vấn trong lệnh SQL. Các truy vấn con này nằm trong mệnh đề WHERE, FROM hoặc SELECT.
- Cú pháp (xem SELECT, GROUP BY, ODER BY, ...)

Tối ưu truy vấn

Việc xác định khi nào một truy vấn là "tối ưu" phụ thuộc vào nhiều yếu tố, bao gồm:

- Thời Gian Thực Hiện: Thời gian mà truy vấn mất để hoàn thành.
- Sử Dụng Tài Nguyên: Bao gồm lượng CPU, bộ nhớ và I/O đĩa mà truy vấn tiêu thụ.
- Khả Năng Mở Rộng: Truy vấn nên thực hiện tốt khi kích thước của bộ dữ liệu tăng lên.
- Dễ Bảo Trì: Truy vấn nên được viết một cách dễ hiểu, dễ bảo trì và sửa đổi nếu cần.
- Tính Đồng Nhất: Kết quả của truy vấn nên chính xác và đồng nhất với dữ liệu trong cơ sở dữ liệu.

Tối ưu truy vấn

Một số ví dụ trong việc tối ưu hóa truy vấn SQL.

- Chỉ lấy những dòng cần thiết

```
-- Not optimized
SELECT * FROM users;
-- Optimized
SELECT user_id, name FROM users;
```

- Sử dụng INDEX cho những cột quan trọng

```
-- Not optimized
SELECT * FROM users WHERE name = 'John';
-- Optimized
CREATE INDEX idx_name ON users(name);
SELECT * FROM users WHERE name = 'John';
```

Tối ưu truy vấn

- Tránh sử dụng SELECT* mà chỉ lấy những cột cần thiết

```
-- Not optimized
SELECT * FROM orders WHERE order_date > '2023-01-01';
-- Optimized
SELECT order_id, customer_id, order_date FROM orders WHERE
    order_date > '2023-01-01';
```

- Sử dụng JOIN thay vì Subquery

```
-- Not optimized
SELECT * FROM customers WHERE customer_id IN (SELECT customer_id
    FROM orders);
-- Optimized
SELECT customers.* FROM customers JOIN orders ON
    customers.customer_id = orders.customer_id;
```

- Sử dụng GROUP BY và HAVING đúng cách

NoSQL (Not only SQL)

- NoSQL là hệ cơ sở dữ liệu không ràng buộc, phân tán, mã nguồn mở, khả năng mở rộng theo chiều ngang, có thể chứa hàng petabytes, độ chịu tải và chịu lỗi cao, yêu cầu về tài nguyên phần cứng thấp
- Không sử dụng mô hình dữ liệu quan hệ
- Hệ thống lưu trữ phân tán

NoSQL (Not only SQL)

Tính năng	SQL	NoSQL
Hiệu suất	Kém hơn NoSQL vi khi truy vấn nó phải tính toán, kiểm tra và xử lý các mối quan hệ trong bảng	Tốt hơn SQL vì nó bỏ qua các ràng buộc
Mở rộng theo chiều ngang	Có thể thực hiện được nhưng quá trình mở rộng sẽ rất phức tạp nếu đã tồn tại dữ liệu trong database	Mở rộng dễ dàng

NoSQL (Not only SQL)

Tính năng	SQL	NoSQL
Tốc độ read/write	Kém hơn NoSQL vì phải đảm bảo tính ràng buộc dữ liệu giữa các bảng. Nếu sử dụng nhiều server phải bảo toàn tính nhất quán dữ liệu ờ các server.	Tốc độ nhanh hơn SQL vì bỏ qua cơ chế ràng buộc của các bảng.
Phần cứng	Đòi hỏi phần cứng cao	Không đòi hỏi quá cao phần cứng

NoSQL (Not only SQL)

Tính năng	SQL	NoSQL
Thay đổi số node trong hệ thống	Vì tính nhất quán về dữ liệu nên khi thêm hay xóa 1 node cần phải shutdown hệ thống trong 1 khoảng thời gian	Vì tính nhất quán cuối nên sẽ không cần shut-down hệ thống
Truy vấn và báo cáo	Dễ dàng sử dụng ngôn ngữ SQL query để truy vấn trực tiếp hoặc dữ liệu từ database dùng công cụ hổ trợ để lấy báo cáo	Việc lấy báo cáo dữ liệu trực tiếp từ NoSQL chưa được hỗ trợ tốt, thực hiện chủ yếu qua giao diện ứng dụng

NoSQL (Not only SQL)

Tính năng	SQL	NoSQL
Mở rộng dữ liệu	Khi muốn bổ sung thêm cột cho một bảng, chúng ta phải khai báo trước	Không cần khai báo trước
Ứng dụng	Sử dụng để xây dựng những hệ thống có quan hệ chặt chẽ và cần tính đồng nhất về dữ liệu như: tài chính ngân hàng, chứng khoán.	Sử dụng để xây dựng những hệ thống thông tin lớn, không quá quan trọng về vấn đề đồng nhất dữ liệu trong một thời gian nhất định.

So sánh một số thuật ngữ cơ bản trong SQL và NoSQL

SQL	**NoSQL**	**Giải thích**
Database	Database	Cơ sở dữ liệu
Table	Collection	Chứa nhiều bản ghi (records)
Row/ Record	Document	Mỗi dòng là một tài liệu dạng (key-value)
Column	Field	Tên trường trong document (có thể khác nhau giữa các document)
Primary key	ObjectId	Mỗi document có id duy nhất
Schema cố định	Schema linh hoạt	Không cần định nghĩa sẵn cột; các document có thể khác cấu trúc

Một số lệnh cơ bản

- Tạo dữ liệu: insertOne() hoặc insertMany()
- Truy vấn dữ liệu: find()
- Cập nhật dữ liệu: updateOne() hoặc updateMany()
- Xoá dữ liệu: deleteOne() hoặc deleteMany()
- Đếm số lượng document: countDocuments()

Cú pháp tạo dữ liệu

- Thêm một document:

```
db.<collection_name>.insertOne({
  <field1>: <value1>,
  <field2>: <value2>,
  ...
});
```

- Thêm nhiều document:

```
db.<collection_name>.insertMany([
  { <field1>: <value1>, <field2>: <value2>, ... },
  { <field1>: <value1>, <field2>: <value2>, ... },
  ...
]);
```

Cú pháp truy vấn dữ liệu

- Hiểu thị tất cả dữ liệu

```
db.<collection_name>.find();
```

- Lấy dữ liệu với nhiều điều kiện

```
db.<collection_name>.find({
  $and: [
    { <field1>: <condition1> },
    { <field2>: <condition2> }
  ]
});
```

- Giới hạn và sắp xếp kết quả

```
db.<collection_name>.find().sort({ <field>: 1 }).limit(<number>);
```

Cú pháp cập nhật dữ liệu

- Cập nhật một document:

```
db.<collection_name>.updateOne(
  { <filter_condition> },
  { $set: { <field_to_update>: <new_value> } }
);
```

- Cập nhật nhiều document:

```
db.<collection_name>.updateMany(
  { <filter_condition> },
  { $set: { <field_to_update>: <new_value> } }
);
```

Cú pháp xoá dữ liệu

- Xoá một document:

```
db.<collection_name>.deleteOne({ <filter_condition> });
```

- Xoá nhiều document:

```
db.<collection_name>.deleteMany({ <filter_condition> });
```

Cú pháp liên kết các collection

Cú pháp liên kết 2 collection:

```
db.<collection1>.aggregate([
  { $lookup: {
      from: "<collection2>",
      localField: "<field_in_collection1>",
      foreignField: "<field_in_collection2>",
      as: "<alias_field>"
  }}
]);
```