# **BÁO CÁO NHỮNG KIẾN THỨC TÌM HIỂU ĐƯỢC**

1. **SQL**

**SQL là gì?**

SQL là viết tắt của Structured Query Language và nó là một ngôn ngữ máy tính tiêu chuẩn ANSI để truy cập và thao tác với các hệ thống cơ sở dữ liệu. Nó được sử dụng để quản lý dữ liệu trong hệ quản trị cơ sở dữ liệu quan hệ lưu trữ dữ liệu dưới dạng bảng và mối quan hệ giữa các dữ liệu cũng được lưu trữ dưới dạng bảng. Câu lệnh SQL được sử dụng để truy xuất và cập nhật dữ liệu trong cơ sở dữ liệu.

Trong thực tế, SQL là ngôn ngữ chuẩn được sử dụng hầu hết cho hệ cơ sở dữ liệu quan hệ. Tất cả các hệ thống quản lý cơ sở dữ liệu quan hệ (RDMS) như MySQL, MS Access, Oracle, Postgres và SQL Server… đều sử dụng SQL làm ngôn ngữ cơ sở dữ liệu chuẩn.

**Tại sao sử dụng SQL và nó lại quan trọng?**

SQL được sử dụng phổ biến vì nó có các ưu điểm sau:

* Cho phép truy cập dữ liệu trong các hệ thống quản lý cơ sở dữ liệu quan hệ. Cho phép mô tả dữ liệu.
* Cho phép xác định dữ liệu trong cơ sở dữ liệu và thao tác dữ liệu đó. Cho phép nhúng trong các ngôn ngữ khác sử dụng mô-đun SQL, thư viện và trình biên dịch trước.
* Cho phép tạo và thả các cơ sở dữ liệu và bảng.
* Cho phép tạo chế độ view, thủ tục lưu trữ, chức năng trong cơ sở dữ liệu.
* Cho phép thiết lập quyền trên các bảng, thủ tục và view.

**Các lệnh SQL**

Sau đây là một số nhóm lệnh cơ bản (được phân loại theo bản chất của lệnh) của SQL: DDL - Data Definition Language (Ngôn ngữ định nghĩa dữ liệu)

* CREATE: Tạo bảng mới, view của bảng và các đối tượng khác trong cơ sở dữ liệu.
* ALTER: Chỉnh sửa các đối tượng dữ liệu đã có, như bảng.
* DROP: Xóa toàn bộ bảng, view của bảng hoặc các đối tượng khác trong cơ sở dữ liệu.

DML - Data Manipulation Language (Ngôn ngữ để thao tác với dữ liệu)

* SELECT: Trích xuất bản ghi cụ thể từ một hoặc nhiều bảng
* INSERT: Chèn dữ liệu mới vào cơ sở dữ liệu.
* UPDATE: Sửa đổi, cập nhật dữ liệu trong cơ sở dữ liệu.
* DELETE: Xóa dữ liệu từ cơ sở dữ liệu.

DCL - Data Control Language (Ngôn ngữ kiểm soát dữ liệu)

* GRANT: Cấp đặc quyền cho user
* REVOKE: Lấy lại quyền đã cấp cho user

1. **NoSQL**

NoSQL là một DMS không quan hệ, không yêu cầu một lược đồ cố định, tránh các phép nối, và dễ dàng mở rộng. Cơ sở dữ liệu NoSQL được sử dụng cho các kho dữ liệu phân tán với nhu cầu lưu trữ dữ liệu khổng lồ. NoSQL được sử dụng cho dữ liệu lớn và các ứng dụng web thời gian thực. Ví dụ như các công ty như Twitter, Facebook, Google thu thập hàng terabyte dữ liệu người dùng mỗi ngày.

Cơ sở dữ liệu NoSQL là viết tắt của "Không chỉ SQL" hoặc "Không phải SQL". Mặc dù một thuật ngữ tốt hơn sẽ NoREL NoSQL bắt gặp. Carl Strozz giới thiệu khái niệm NoSQL vào năm 1998.

RDBMS truyền thống sử dụng cú pháp SQL để lưu trữ và truy xuất dữ liệu để có thêm thông tin chi tiết. Thay vào đó, một hệ thống cơ sở dữ liệu NoSQL bao gồm một loạt các công nghệ cơ sở dữ liệu có thể lưu trữ dữ liệu có cấu trúc, bán cấu trúc, không có cấu trúc và đa hình.

**Các đặc điểm của NoSQL**

Sau đây mình sẽ liệt kê một vài đặc điểm của NoSQL:

* Phi quan hệ: không có ràng buộc nào cho việc nhất quán dữ liệu.
* Mô hình lưu trữ phân tán các tập tin hoặc dữ liệu ra nhiều máy khác nhau trong mạng LAN hoặc Internet dưới sự kiểm soát của phần mềm.
* NoSQL lưu trữ dữ liệu của mình theo dạng cặp giá trị “key – value”. Sử dụng số lượng lớn các node để lưu trữ thông tin.
* Tính nhất quán không theo thời gian thực: Sau mỗi thay đổi CSDL, không cần tác động ngay đến tất cả các CSDL liên quan mà được lan truyền theo thời gian.
* Mô hình dữ liệu và truy vấn linh hoạt.
* Triển khai đơn giản, dễ nâng cấp và mở rộng.

1. **JOIN**

Trong [SQL Server](https://quantrimang.com/sql-server) (Transact-SQL) JOIN được dùng để lấy dữ liệu từ nhiều bảng, xảy ra khi 2 hoặc nhiều bảng được kết nối với nhau trong một lệnh SQL. Dưới đây là 4 loại JOIN trong SQL Server

* INNER JOIN - hay còn gọi là hình thức kết nối đơn giản
* LEFT OUTER JOIN - hay LEFT JOIN
* RIGHT OUTER JOIN - hay RIGHT JOIN
* FULL OUTER JOIN - hay FULL JOIN

**INNER JOIN**

Nhiều khả năng bạn đã dùng INNER JOIN trong lệnh nào đó rồi. Đây là loại kết nối phổ biến nhất, trả về tất cả các hàng từ các bảng khi điều kiện kết nối được đáp ứng.

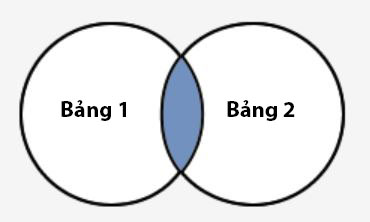
Cú pháp INNER JOIN

SELECT cot

FROM bang1

INNER JOIN bang2

ON bang1.cot = bang2.cot;



**Ví dụ INNER JOIN**

SELECT nhacung.nhacung\_id, nhacung.nhacung\_ten, donhang.donhang\_ngay

FROM nhacung

INNER JOIN donhang

ON nhacung.nhacung\_id = donhang.nhacung\_id;

**LEFT OUTER JOIN**

Một số cơ sở dữ liệu dùng LEFT JOIN thay vì LEFT OUTER JOIN. Kiểu kết nối này trả về các hàng nằm ở bảng bên trái trong điều kiện ON và chỉ lấy ở các bảng còn lại những hàng đáp ứng được điều kiện đặt ra

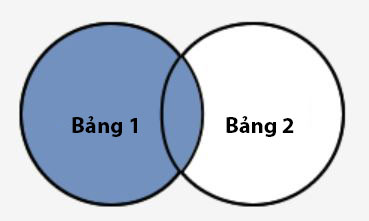
Cú pháp LEFT JOIN

SELECT cot

FROM bang1

LEFT [OUTER] JOIN bang2

ON bang1.cot = bang2.cot;



**Ví dụ LEFT JOIN**

SELECT nhacung.nhacung\_id, nhacung.nhacung\_ten, donhang.donhang\_ngay

FROM nhacung

LEFT OUTER JOIN donhang

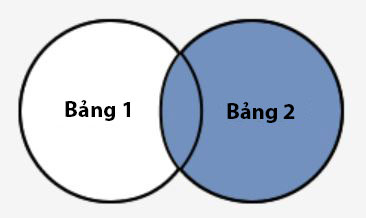
ON nhacung.nhacung\_id = donhang.nhacung\_id;

**RIGHT OUTER JOIN**

Một số cơ sở dữ liệu dùng RIGHT JOIN thay vì RIGHT OUTER JOIN. Kiểu kết nối này trả về các hàng nằm ở bảng bên phải trong điều kiện ON và chỉ lấy ở các bảng còn lại những hàng đáp ứng được điều kiện đặt ra

Cú pháp RIGHT JOIN

SELECT cot FROM bang1 RIGHT [OUTER] JOIN bang2 ON bang1.cot = bang2.cot;



Ví dụ RIGHT JOIN

SELECT donhang.donhang\_id, donhang.donhang\_ngay, nhacung.nhacung\_ten

FROM nhacung

RIGHT OUTER JOIN donhang

ON nhacung.nhacung\_id = donhang.nhacung\_id;

**FULL OUTER JOIN**

Một số cơ sở dữ liệu dùng FULL JOIN thay vì FULL OUTER JOIN. Kiểu kết nối này trả về tất cả các hàng nằm ở bảng trái và phải và giá trị NULL đặt ở những nơi không đáp ứng được điều kiện.

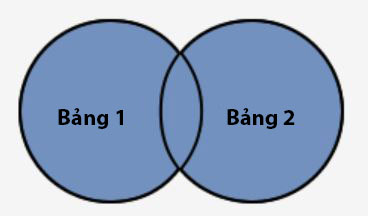
Cú pháp FULL JOIN

SELECT cot

FROM bang1

FULL [OUTER] JOIN bang2

ON bang1.cot = bang2.cot;



**Ví dụ FULL JOIN**

SELECT nhacung.nhacung\_id, nhacung.nhacung\_ten, donhang.donhang\_ngay

FROM nhacung

FULL OUTER JOIN donhang

ON nhacung.nhacung\_id = donhang.nhacung\_id;

1. **Function**

Là một đối tượng trong cơ sở dữ liệu (CSDL) sử dụng trong các câu lệnh SQL, được biên dịch sẵn và lưu trong CSDL nhằm mục đích thực hiện xử lý nào đó như tính toán phức tạp và trả về kết quả là giá trị nào đó.

**Đặc điểm:**

* Luôn trả về giá trị
* Gồm 2 loại: Function hệ thống và Function do người dùng tự định nghĩa
* Function người dùng tự định nghĩa gồm 2 loại:
  + **Scalar-valued**: Trả về giá trị vô hướng của các kiểu dữ liệu T-SQL
  + **Table-valued**: Trả về bảng, là kết quả của một hoặc nhiều lệnh

**Tạo Function trả về giá trị loại Scalar-valued**

CREATE FUNCTION <Tên function>

([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], …, [...]])

RETURNS <kiểu dữ liệu>

[WITH ENCRYPTION]

[AS]

BEGIN

[Thân của hàm]

RETURN <Biểu thức giá trị đơn>

END

**Trong đó:**

* **Tên function**: Tên của hàm chúng ta sẽ tạo
* **Tên tham số**: Là các tham số Input cho hàm. Khai báo báo gồm tên của tham số (trước tên tham số sử dụng tiền tố @), kiểu dữ liệu của tham số, chúng ta có thể chỉ định giá trị mặc định cho tham số. Có thể chỉ định nhiều tham số đầu vào
* **RETURNS**: từ khóa này chỉ định kiểu dữ liệu hàm sẽ trả về. Kiểu dữ liệu phải được chỉ định kiểu độ dài dữ liệu. Ví dụ: varchar (100)
* **WITH ENCRYPTION**: Từ khóa chỉ định code của hàm sẽ được mã hóa trong bảng syscomments.
* **AS**: Từ khóa cho biết code của hàm bắt đầu.
* **BEGIN**: Đi cùng với END để tạo thành bao khối bao các câu lệnh trong thân hàm.
* **RETURN**: Từ khóa này sẽ gửi giá trị tới thủ tục gọi hàm. **Một số lưu ý:**
* Tên function phải là duy nhất trong 1 CSDL. Function được tạo/định nghĩa trong CSDL nào thì chỉ sử dụng trong CSDL đó. Khác với Function có sẵn của SQL được truy cập ở bất cứ đâu.
  + 1. **Tạo Function trả về giá trị loại Table-valued**

Function **Table-valued** có 2 loại:

* **Hàm giá trị bảng đơn giản**: Trả về bảng, là kết quả của một câu lệnh SELECT đơn
* **Hàm giá trị bảng đa câu lệnh**: Trả về bảng, là kết quả của nhiều câu lệnh

1. Hàm giá trị bảng đơn giản

CREATE FUNCTION <Tên function>

([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], …,[...]])

RETURNS TABLE

[WITH ENCRYPTION]

[AS]

RETURN <Câu lệnh SQL>

END

* **Lưu ý** Hàm giá trị bảng đơn còn được gọi là hàm giá trị bảng nội tuyến. Có thể được dùng trong câu lệnh truy vấn thay thế cho tên bảng hoặc tên View.

**b) Hàm giá trị bảng đa câu lệnh**

CREATE FUNCTION <Tên function>

([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], …,[...]])

RETURNS @<tên biến trả về> TABLE (<tên cột 1> <kiểu dữ liệu> [tùy chọn thuộc tính], ..., <tên cột n> <kiểu dữ liệu> [tùy chọn thuộc tính])

[AS]

BEGIN

<Câu lệnh SQL>

RETURN

END

**Thay đổi, xóa, xem nội dung Function**

1. **Thay đổi Function**

Để thay đổi các hàm đã khai báo ta sử dụng câu lệnh ALTER FUNCTION. Cú pháp tương tự như tạo mới Function, chỉ thay từ khóa CREATE bằng từ khóa ALTER

ALTER FUNCTION <Tên function>

([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], …,[...]])

RETURNS <kiểu dữ liệu> | TABLE

[WITH ENCRYPTION]

[AS]

BEGIN

[Thân của hàm]

RETURN <Biểu thức giá trị đơn> | Câu lệnh SQL

END

1. Xóa Function

Để xóa hàm ta dùng câu lệnh DROP FUNCTION.

DROP FUNCTION [schema\_name.] <Tên function>

1. Xem nội dung Function

Để xem nội dung function ta sử dụng Store Procedure (Thủ tục) có sẵn của SQL là sp\_helptext (Transact-SQL)

EXEC sp\_helptext 'FunctionName'

1. **STORE PROCEDURE**

Stored Procedure là 1 phần không thể thiếu của SQL Server. Chúng có thể hỗ trợ rất nhiều cho lập trình và cấu hình cơ sở dữ liệu.

Một Stored Procedure là bao gồm các câu lệnh Transact-SQL và được lưu lại trong cơ sở dữ liệu. Các lập trình viên chỉ cần gọi ra và thực thi thông qua SQL Server Management Studio hoặc ngay trong ứng dụng đang phát triển.

Transact-SQL dựa trên SQL, nó là một ngôn ngữ lập trình được sử dụng làm trung gian giữa cơ sở dữ liệu và các ứng dụng. Nó tương đối dễ học vì thực chất nó được tạo bởi hầu hết là các lệnh SQL.

**Tạo Stored Procedure**

Bạn tạo Stored Procedure được lưu trữ trong SQL Server Management Studio bằng cách sử dụng câu lệnh CREATE PROCEDURE:

CREATE PROCEDURE StoredProcedureName AS

...

Đoạn mã sau tạo một Stored Procedure gọi là "MyStoredProcedure":

CREATE PROCEDURE MyStoredProcedure AS

SET ROWCOUNT 10

SELECT Products.ProductName AS TenMostExpensiveProducts, Products.UnitPrice

FROM Products

ORDER BY Products.UnitPrice DESC

# **Chỉnh sửa Stored Procedure**

Nếu bạn cần sửa đổi một Stored Procedure, bạn chỉ cần thay CREATE bằng ALTER.

ALTER PROCEDURE MyStoredProcedure AS

...

# **Thực thi Stored Procedure**

Bạn có thể chạy một Stored Procedure bằng cách sử dụng EXECUTE hoặc EXEC. Ví dụ: để chạy Stored Procedure ở trên, hãy nhập như sau:

EXEC MyStoredProcedure

Nếu Stored Procedure có khoảng trắng bên trong tên của nó, hãy đặt nó giữa các dấu ngoặc kép:

EXEC "My Stored Procedure"

Nếu Stored Procedure của bạn cần truyên thêm các param:

EXEC MyStoredProcedure @ParameterName="MyParameter"

1. **TRANSACTION**

Một tiến trình xử lý có xác định điểm đầu và điểm cuối, được chia nhỏ thành các operation (phép thực thi), tiến trình được thực thi một cách tuần tự và độc lập các operation đó theo nguyên tắc hoặc tất cả đều thành công hoặc một operation thất bại thì toàn bộ tiến trình thất bại. Nếu việc thực thi một operation nào đó bị fail đồng nghĩa với việc dữ liệu phải rollback về trạng thái ban đầu. Transaction được xem là thành công khi và chỉ khi tất cả thao tác sửa đổi dữ liệu trong một giao dịch được thực hiện và được lưu trong cơ sở dữ liệu vĩnh viễn (dữ liệu đã được commit). Nếu Transaction được quay lui (rollback) hoặc hủy bỏ, khi đó nó có nghĩa là Transaction đã gặp các lỗi và không thực hiện bất kỳ thay đổi nào trong nội dung của cơ sở dữ liệu. Do đó, Transaction có thể được triển khai hoặc rollback.

**Thuộc tính của Transaction**

**1. Atomicity - Tính bảo toàn:**nguyên tắc "all or nothing", đảm bảo rằng tất cả các câu lệnh trong nhóm lệnh được thực thi thành công. Nếu không, Transaction bị hủy bỏ tại thời điểm thất bại và tất cả các thao tác trước đó được khôi phục về trạng thái cũ đồng nghĩa với việc không có gì thay đổi về mặt dữ liệu.

**2. Consistency - Tính nhất quán:** đảm bảo rằng cơ sở dữ liệu thay đổi chính xác các trạng thái khi một transaction được thực thi thành công.

**3. Isolation - Tính độc lập:** cho phép các Transaction hoạt động độc lập và minh bạch với nhau.

**4. Durability - Tính bền vững:** đảm bảo rằng kết quả của một transaction được xác định, không có chuyện dữ liệu của Transaction sau khi thực thi có thể chuyển lại trạng thái dữ liệu lúc trước khi thực hiện.

Xử lý Transaction

COMMIT: để lưu các thay đổi.

ROLLBACK: để quay trở lại trạng thái trước khi có thay đổi. SAVEPOINT: tạo các điểm (point) bên trong các nhóm Transaction để ROLLBACK, tức là để quay trở lại điểm trạng thái đó.

SET TRANSACTION: đặt một tên cho một Transaction.

Các lệnh điều khiển Transaction chỉ được sử dụng với các lệnh thao tác dữ liệu như INSERT, UPDATE và DELETE. Tuy nhiên chúng không thể được sử dụng trong lệnh CREATE TABLE hoặc DROP TABLE vì các hoạt động này được tự động xác định trong cơ sở dữ liệu.

1. **INDEX**

Khái niệm về chỉ mục (index) trong mysql

Chỉ mục (Index) là bảng tra cứu đặc biệt mà Database Search Engine có thể sử dụng để tăng nhanh thời gian và hiệu suất thu thập dữ liệu. Hiểu đơn giản, một chỉ mục là một con trỏ tới dữ liệu trong một bảng. Một chỉ mục trong một Database là tương tự như một chỉ mục trong Mục lục của cuốn sách.

Ví dụ, nếu bạn muốn tham chiếu tất cả các trang trong một cuốn sách về một chủ đề nào đó, đầu tiên bạn nghĩ ngay đến mục lục của nó, mà liệt kê tất cả các chương, chủ đề theo thứ tự và sau đó được tham chiếu tới một hoặc nhiều trang cụ thể. Khi đã có mục lục của cuốn sách, việc tìm kiếm trang sách đó sẽ nhanh chóng và đỡ tốn công hơn rất nhiều so với việc bạn phải lật từng trang.

* **Cú pháp**