**Họ và tên:** Nguyễn Quốc Hùng

**BÁO CÁO EX3 – MYSQL**

1. Sql

* **Sql là gì?:**

SQL là viết tắt của Structured Query Language (Ngôn ngữ truy vấn có cấu trúc) và nó là ngôn ngữ máy tính tiêu chuẩn ANSI để truy cập và thao tác các hệ thống cơ sở dữ liệu. Nó được sử dụng để quản lý dữ liệu trong hệ thống quản lý 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.

* **Các đặc điểm của Sql:**
* **Ngôn ngữ Định nghĩa Dữ liệu (DDL):** SQL cung cấp một tập hợp các lệnh để định nghĩa và sửa đổi cấu trúc của một cơ sở dữ liệu, bao gồm việc tạo bảng, sửa đổi cấu trúc bảng và xóa bảng.
* **Ngôn ngữ Manipulation Dữ liệu (DML):** SQL cung cấp một tập hợp các lệnh để thao tác dữ liệu trong cơ sở dữ liệu, bao gồm việc thêm, sửa đổi và xóa dữ liệu.
* **Ngôn ngữ Truy vấn:** SQL cung cấp một bộ lệnh phong phú để truy vấn cơ sở dữ liệu để lấy dữ liệu, bao gồm khả năng lọc, sắp xếp, nhóm và kết hợp dữ liệu từ nhiều bảng.
* **Kiểm soát Giao dịch:** SQL hỗ trợ xử lý giao dịch, cho phép người dùng nhóm một tập hợp các hoạt động cơ sở dữ liệu thành một giao dịch duy nhất có thể được lùi lại trong trường hợp thất bại.
* **Tính toàn vẹn Dữ liệu:** SQL bao gồm các tính năng để bảo vệ tính toàn vẹn dữ liệu, chẳng hạn như khả năng xác định các ràng buộc về các giá trị có thể được chèn hoặc cập nhật trong một bảng, và để bảo vệ tính toàn vẹn tham chiếu giữa các bảng.
* **Kiểm soát Truy cập Người dùng:** SQL cung cấp các cơ chế để kiểm soát quyền truy cập của người dùng vào cơ sở dữ liệu, bao gồm khả năng cấp và thu hồi quyền để thực hiện các hoạt động cụ thể trên cơ sở dữ liệu.
* **Tính di động:** SQL là một ngôn ngữ tiêu chuẩn hóa, có nghĩa là mã SQL được viết cho một hệ quản trị cơ sở dữ liệu có thể được sử dụng trên một hệ thống khác với sự sửa đổi tối thiểu

1. No sql

* **NoSql là gì?:**

Cơ sở dữ liệu NoSQL là một Hệ thống quản lý dữ liệu không quan hệ (non-relational Data Management System) có lược đồ (schema) linh hoạt. Nó dễ mở rộng. Mục đích chính của việc sử dụng cơ sở dữ liệu NoSQL là dành cho các kho dữ liệu phân tán với nhu cầu lưu trữ dữ liệu lớn. NoSQL được sử dụng cho Dữ liệu lớn và ứng dụng web thời gian thực. Chẳng hạn các công ty như Twitter, Facebook và Google thu thập hàng terabyte dữ liệu người dùng mỗi ngày

* **Các đặc điểm của NoSql:**
* Chúng tuân thủ các schema linh hoạt không đòi hỏi bạn phải xác định hoặc khai báo một schema cố định cho dữ liệu của bạn, làm cho chúng lý tưởng cho dữ liệu bán cấu trúc và không cấu trúc.
* Chúng mở rộng theo chiều ngang, sử dụng phân phối theo phạm vi hoặc băm, thay vì phải phụ thuộc vào việc mở rộng theo chiều dọc để thêm khả năng.
* Chúng được tối ưu hóa cho các mô hình dữ liệu và mẫu công việc cụ thể, như key-value, wide-column, hoặc in-memory.
* Nhiều cơ sở dữ liệu NoSQL thể hiện tính nhất quán tại một thời điểm sau (ví dụ, mô hình nhất quán cuối cùng) thay vì tuân thủ các thuộc tính ACID (atomicity, consistency, isolation, durability) nghiêm ngặt của cơ sở dữ liệu quan hệ và SQL.
* Đa số cơ sở dữ liệu NoSQL không hỗ trợ các giao dịch chéo-shard hoặc các chế độ cách ly linh hoạt.

Những tính năng này làm cho cơ sở dữ liệu phi quan hệ lý tưởng cho các ứng dụng yêu cầu quy mô lớn, độ tin cậy, sẵn có cao và thay đổi dữ liệu thường xuyên. Với một hệ thống NoSQL, khi bạn cần thêm dung lượng, bạn có thể thêm các máy chủ bổ sung để hỗ trợ nhu cầu thay vì phải tắt hệ thống SQL của bạn để nâng cấp vật lý tài nguyên của hệ thống.

1. Join

* **Các loại JOIN:**
* **INNER JOIN:** Trả về các record có giá trị khớp trong cả hai bảng
* **LEFT JOIN:** Trả về tất cả các record từ bảng bên trái (table1) và mọi record phù hợp từ bảng bên phải (table2). Nếu không khớp, giá trị NULL sẽ được hiển thị cho các cột của bảng bên phải (table2).
* **RIGHT JOIN:** Trả về tất cả các record từ bảng bên phải (table2) và mọi record phù hợp từ bảng bên trái (table1). Nếu không khớp, giá trị NULL sẽ được hiển thị cho các cột của bảng bên trái (table1).
* **FULL JOIN:** Trả về các record có giá trị khớp trong cả hai bảng, cũng như bất kỳ record nào chưa khớp từ một trong hai bảng <==> LEFT JOIN union RIGHT JOIN
* **CROSS JOIN:** Trả về tích Descartes của hai bảng (nghĩa là mọi tổ hợp các record từ cả hai bảng)
* **SELF JOIN:** Trả về các record có giá trị khớp với điều kiện khi một bảng được join với chính nó

1. Function

* **Function trong Sql là gì?:**

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
* **Một số function có sẵn của sql:**
* String functions: CONCAT, TRIM, SUBSTRING, LOWER, UPPER,…
* Numeric functions: ABS, ACOS, ASIN, AVG, MIN, MAX, CEIL, FLOOR,…
* Date functions: DATEDIFF, DATE, MONTH, YEAR, …
* Advanced functions: CAST, CONVERT, ISNULL,…
* **Cú pháp tự định nghĩa function:2**

*CREATE FUNCTION sf\_name ([parameter(s)])*

*RETURNS data type*

*DETERMINISTIC*

*STATEMENTS*

Trong đó:

* CREATE FUNCTION sf\_name ([parameter(s)]] là phần bắt buộc. Nó sẽ nói cho máy chủ MySQL tạo một function tên là ‘sf\_name’ với những tham số tự chọn. Những tham số này được định nghĩa trong dấu ngoặc tròn.
* RETURNS data type là bắt buộc. Nó cụ thể hóa loại dữ liệu mà hàm cần trả về.
* DETERMINISTIC có nghĩa là hàm sẽ trả lại đúng giá trị nếu đối số được cung cấp những giá trị đó.
* STATEMENTS là code phương thức để hàm thực thi.

1. Procedure

* **Stored Procedure là gì?:**
* Stored Procedures - Thủ tục lưu trữ: là một tập các khai báo lệnh SQL nhằm thực hiện một hoặc một số các xử lý nào đó.
* Stored Procedure có thể nhận tham số đầu vào nhưng lại không bắt buộc phải trả về kết quả như Function, tuy nhiên có thể trả về một lúc nhiều tham số đầu ra - điều mà Function không làm được.
* **Cú pháp tạo Stored Procedure**

*CREATE PROCEDURE procedure\_name (*

*[IN | OUT | INOUT] parameter\_name datatype[(length)]*

*)*

*BEGIN*

*statements;*

*END $$*

*-- Ký tự phân cách $$ có thể định nghĩa lại bằng khai báo DELIMITER <characters>, ví dụ: DELIMITER $$*

* **Cú pháp gọi Strored Procedure:**

*CALL <procedure name> ([<List value input> , <List output: @out\_name>]);*

* **Cú pháp xóa Strored Procedure:**

*DROP PROCEDURE [IF EXISTS] procedure\_name;*

* **Cú pháp xem danh sách các Strored Procedure:**

*SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search\_condition];*

1. Transaction

* **Transaction trong Sql là gì?:**

Có thể hiểu Transaction là 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.

* **Các kiểu của transaction:**
* Flat transaction – Transaction ngang hàng: Việc chia các operation là ngang hàng nhau. Thực thi các operation là tuần tự từ trái sang phải hoặc từ trên xuống dưới.
* Nested Transaction – Transaction lồng nhau: Việc thực thi các operation dựa theo nguyên tắc từ trong ra ngoài. Như vậy chúng ta thấy các operation ở dạng này có vẻ phụ thuộc vào nhau nhưng khi thực thi thì là độc lập theo nguyên tắc operation trong thực thi xong thì mới đến operation ngoài.
* **Các thuộc tính của Transaction:**
* **Atomicity**: Một transaction xác định ranh giới của nó rất rõ ràng, tức xác định điểm bắt đầu và kết thúc của tiến trình. Như vậy có thể coi nó như một đơn vị thực thi và đơn vị thực thi này thực hiện theo nguyên tắc “all or nothing”. Nghĩa là nếu một thành phần nào đó trong transaction thực thi hỏng (fail) thì đồng nghĩa với việc không có gì xảy ra tức không có gì thay đổi về mặt dữ liệu.
* **Consistency**: Dữ liệu nhất quán với transaction ở thời điểm bắt đầu và kết thúc. Nhất quán ở transaction là strong consistency. Để tìm hiểu kỹ hơn về tính nhất quán, xin đọc lại bài viết NoSQL.
* **Isolation**: Nếu hai transaction thực thi cùng lúc thì nguyên tắc thực thi là thực thi độc lập. Nghĩa là một transaction không thể “nhìn thấy” một transaction khác. “Không nhìn thấy” ở đây là không tác động lẫn nhau, chủ yếu trên dữ liệu.
* **Durability**: Dữ liệu của transaction sau khi thực thi xong được cố định, chính thức và bền vững. Nghĩa là những thay đổi đã được cố định, không có chuyện có thể chuyển lại trạng thái dữ liệu lúc trước khi thực hiện transaction.

1. Index

* **Index trong Sql là gì?:**

Index – Chỉ mục là một cấu trúc dữ liệu giúp tìm các hàng có giá trị cột cụ thể một cách nhanh chóng. Nếu không có chỉ mục, MySQL phải bắt đầu từ hàng đầu tiên và sau đó đọc qua toàn bộ bảng để tìm các hàng có liên quan. Bảng càng lớn thì chi phí thực hiện việc này càng cao. Nếu bảng có chỉ mục cho các cột được đề cập, MySQL có thể nhanh chóng xác định vị trí cần tìm ở giữa tệp dữ liệu mà không cần phải xem tất cả dữ liệu. Điều này nhanh hơn nhiều so với việc đọc từng hàng một cách tuần tự.

* **Ưu điểm/ Nhược điểm:**
* Ưu điểm:
  + Tăng tốc độ tìm kiếm records theo câu lệnh WHERE.
  + Không chỉ giới hạn trong câu lệnh SELECT mà với cả xử lý UPDATE hay DELETE có điều kiện WHERE
* Nhược điểm:
  + Khi sử dụng index thì tốc độ của những xử lý ghi dữ liệu (Insert, Update, Delete) sẽ bị chậm đi vì ngoài việc thêm hay update thông tin data thì MYSQL cũng cần update lại thông tin index của bảng tương ứng.
  + Tốn resource của server như thêm dung lượng cho CSDL.
* **Các kiểu index trong MySQL:**
* **B-Tree Index** : Thông thường khi nói đến index mà không chỉ rõ loại index thì default là sẽ sử dụng B-Tree index.
  + Dữ liệu index được tổ chức và lưu trữ theo dạng tree, tức là có root, branch, leaf.
  + Giá trị của các node được tổ chức tăng dần từ trái qua phải
  + B-Tree index được sử dụng trong các biểu thức so sánh dạng: =, >, >=, <, <= ⇒ Có thể tối ưu tốt cho câu lệnh ORDER BY.
  + Khi truy vấn dữ liệu thì CSDL sẽ không scan dữ liệu trên toàn bộ bảng để tìm dữ liệu, việc tìm kiếm trong B-Tree là 1 quá trình đệ quy, bắt đầu từ root node và tìm kiếm tới branch và leaf, đến khi tìm được tất cả dữ liệu – thỏa mãn với điều kiện truy vấn thì mới dùng lại.
* **Hash Index:** Hash index dựa trên giải thuật Hash Function (hàm băm). Tương ứng với mỗi khối dữ liệu (index) sẽ sinh ra một bucket key(giá trị băm) để phân biệt.
  + Khác với B-Tree, thì Hash index chỉ nên sử dụng trong các biểu thức toán tử là = và <>. Không sử dụng cho toán từ tìm kiếm 1 khoảng giá trị như > hay < .
  + Hash có tốc độ nhanh hơn kiểu B-Tree.
* **R-Tree Index**: Sử dụng cho các loại dữ liệu hình học không gian Spatial data và thường ít khi sử dụng.
* **Cú pháp để tạo Index:**

*CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX index\_name*

*[ USING BTREE | HASH ]*

*ON table\_name*

*(index\_col1 [(length)] [ASC | DESC],*

*index\_col2 [(length)] [ASC | DESC],*

*...*

*index\_col\_n [(length)] [ASC | DESC]);*