* 1/ View là gì ? mục đích sử dụng view ?

- Một khung nhìn (view) là một câu lệnh SQL được lưu trữ trong cơ sở dữ liệu với một tên xác định.

- Mục đích sử dụng view:

* Cấu trúc dữ liệu theo cách mà người dùng tìm thấy tự nhiên hoặc trực quan.
* Hạn chế quyền truy cập vào dữ liệu theo cách mà người dùng có thể nhìn thấy và (đôi khi) sửa đổi chính xác những gì họ cần.
* Tóm tắt dữ liệu từ các bảng khác nhau có thể được sử dụng để tạo báo cáo.
* 2/ Index là gì ? Ý nghĩa ? cách sử dụng ? Lưu ý khi sử dụng index ?

2.1 Index (chỉ mục) là một cấu trúc dữ liệu giúp tăng tốc độ truy xuất dữ liệu từ bảng. Chỉ mục hoạt động như một bảng tra cứu, cho phép MySQL xác định vị trí của các bản ghi một cách nhanh chóng mà không cần phải quét toàn bộ bảng. Các chỉ mục thường được tạo trên các cột thường xuyên được sử dụng trong các truy vấn tìm kiếm, sắp xếp và lọc dữ liệu.

2.2. Khi nào sử dụng Index?

* Tìm kiếm thường xuyên: Khi bạn thực hiện các truy vấn với điều kiện tìm kiếm (WHERE) trên một cột hoặc nhiều cột.
* Sắp xếp dữ liệu: Khi bạn sử dụng câu lệnh ORDER BY để sắp xếp dữ liệu trong truy vấn.
* Kết nối giữa các bảng: Khi bạn thực hiện các phép toán JOIN giữa các bảng.
* Dữ liệu lớn: Khi bạn có các bảng với số lượng bản ghi lớn, chỉ mục giúp tăng tốc độ truy vấn.

2.3. Lưu ý khi sử dụng Index

* Chi phí ghi: Chỉ mục có thể làm giảm hiệu suất của các thao tác ghi (INSERT, UPDATE, DELETE) vì cơ sở dữ liệu cần phải cập nhật các chỉ mục tương ứng.
* Không nên quá nhiều chỉ mục: Tạo quá nhiều chỉ mục có thể làm tăng chi phí lưu trữ và làm chậm quá trình cập nhật dữ liệu.
* Chọn cột phù hợp: Nên chỉ định các cột thường xuyên được tìm kiếm hoặc có giá trị duy nhất cao để tạo chỉ mục.
* Theo dõi hiệu suất: Nên sử dụng các công cụ và câu lệnh như EXPLAIN để phân tích và tối ưu hóa truy vấn.

2 .4. Cú pháp tạo Index trong MySQL

Để tạo chỉ mục trong MySQL, bạn có thể sử dụng cú pháp sau:

CREATE INDEX index\_name

ON table\_name (column1, column2, ...);

Ví dụ:

Giả sử bạn có một bảng tên là employees và bạn muốn tạo chỉ mục cho cột last\_name:

CREATE INDEX idx\_lastname

ON employees (last\_name);

Các loại Index trong MySQL

* Primary Key: Là một loại chỉ mục đặc biệt mà không cho phép giá trị NULL và đảm bảo tính duy nhất cho mỗi bản ghi.
* Unique Index: Đảm bảo rằng không có hai bản ghi nào có giá trị giống nhau trong cột chỉ mục.
* Full-Text Index: Được sử dụng để tìm kiếm văn bản trong các cột kiểu TEXT.
* Composite Index: Chỉ mục trên nhiều cột.

Sử dụng chỉ mục đúng cách có thể giúp cải thiện hiệu suất của các truy vấn trong MySQL một cách đáng kể.

3/ Store procedure là gì ? Mục đích sử dụng ? Cách tạo Store Procedure ?

3.1. Stored Procedure là gì?

Stored Procedure (thủ tục lưu trữ) trong MySQL là một tập hợp các câu lệnh SQL được lưu trữ trong cơ sở dữ liệu và có thể được gọi và thực thi như một đơn vị duy nhất. Stored Procedure giúp tăng hiệu suất và khả năng quản lý của các tác vụ lặp lại, giảm độ phức tạp trong mã lệnh SQL và có thể giúp bảo mật dữ liệu.

3.2. Mục đích sử dụng Stored Procedure

* Tăng hiệu suất: Giảm bớt chi phí mạng vì các câu lệnh được thực hiện trực tiếp trên máy chủ cơ sở dữ liệu.
* Quản lý mã dễ dàng: Có thể dễ dàng bảo trì và cập nhật các câu lệnh SQL trong một thủ tục lưu trữ mà không cần thay đổi mã trong ứng dụng.
* Tính bảo mật: Cho phép ẩn logic truy vấn và hạn chế quyền truy cập vào bảng cơ sở dữ liệu trực tiếp.
* Thực thi nhiều câu lệnh: Có thể thực hiện nhiều thao tác SQL trong một lần gọi.

3.3. Cú pháp tạo Stored Procedure

Để tạo một Stored Procedure trong MySQL, bạn có thể sử dụng cú pháp sau:

CREATE PROCEDURE procedure\_name (parameter1 datatype, parameter2 datatype, ...)

BEGIN

-- Các câu lệnh SQL

END;

Ví dụ:

Giả sử bạn muốn tạo một Stored Procedure để thêm một nhân viên mới vào bảng employees:

CREATE PROCEDURE AddEmployee (

IN emp\_first\_name VARCHAR(50),

IN emp\_last\_name VARCHAR(50),

IN emp\_salary DECIMAL(10, 2)

)

BEGIN

INSERT INTO employees (first\_name, last\_name, salary)

VALUES (emp\_first\_name, emp\_last\_name, emp\_salary);

END;

3.4. Cú pháp gọi Stored Procedure

Để gọi một Stored Procedure, bạn có thể sử dụng cú pháp sau:

CALL procedure\_name(parameter1, parameter2, ...);

Ví dụ:

Để gọi thủ tục AddEmployee với các tham số cụ thể:

CALL AddEmployee('John', 'Doe', 60000.00);

3.5. Các loại tham số trong Stored Procedure

Trong MySQL, có ba loại tham số mà bạn có thể sử dụng trong Stored Procedure:

1. IN: Tham số đầu vào, cho phép bạn truyền dữ liệu vào Stored Procedure. Đây là loại tham số mặc định nếu bạn không chỉ định rõ.
   * Ví dụ:

IN emp\_name VARCHAR(50)

1. OUT: Tham số đầu ra, cho phép Stored Procedure trả giá trị cho caller.
   * Ví dụ:

OUT total\_count INT

1. INOUT: Tham số đầu vào và đầu ra, cho phép bạn truyền dữ liệu vào Stored Procedure và cũng nhận giá trị trả về.
   * Ví dụ:

INOUT emp\_id INT

Ví dụ với các loại tham số

CREATE PROCEDURE GetEmployeeCount(

IN dept\_id INT,

OUT emp\_count INT

)

BEGIN

SELECT COUNT(\*) INTO emp\_count

FROM employees

WHERE department\_id = dept\_id;

END;

Để gọi thủ tục này, bạn có thể làm như sau:

SET @emp\_count = 0;

CALL GetEmployeeCount(1, @emp\_count);

SELECT @emp\_count; -- Hiển thị số lượng nhân viên trong bộ phận với ID 1

4/ Function ? mục đích sử dụng ? Cú pháp ?

4.1. Function là gì?

Trong MySQL, Function (hàm) là một tập hợp các câu lệnh SQL thực hiện một tác vụ cụ thể và trả về một giá trị duy nhất. Khác với Stored Procedure, function luôn trả về một giá trị và thường được sử dụng trong các câu lệnh SQL như SELECT, WHERE, hoặc để tính toán.

4.2. Cú pháp tạo Function

Cú pháp để tạo một hàm trong MySQL là:

CREATE FUNCTION function\_name (parameter1 datatype, parameter2 datatype, ...)

RETURNS return\_datatype

BEGIN

-- Các câu lệnh SQL

RETURN return\_value;

END;

Giải thích:

* function\_name: Tên của hàm.
* parameter1, parameter2: Các tham số đầu vào của hàm (có thể có hoặc không).
* return\_datatype: Kiểu dữ liệu của giá trị mà hàm trả về.
* RETURN: Từ khóa để trả về giá trị từ hàm.

Ví dụ:

Giả sử bạn muốn tạo một hàm để tính lương sau khi cộng thêm 10% tiền thưởng:

CREATE FUNCTION CalculateSalary (base\_salary DECIMAL(10, 2))

RETURNS DECIMAL(10, 2)

BEGIN

RETURN base\_salary \* 1.10;

END;

4.3. Cú pháp gọi Function

Bạn có thể gọi hàm MySQL trong các câu lệnh SQL, giống như cách bạn gọi các hàm tích hợp sẵn. Cú pháp gọi hàm như sau:

SELECT function\_name(parameter1, parameter2, ...);

Ví dụ:

Để gọi hàm CalculateSalary vừa tạo ở trên:

SELECT CalculateSalary(50000.00) AS new\_salary;

Kết quả sẽ là lương sau khi thêm 10% tiền thưởng.

5/ Trigger là gì ? Ý nghĩa và cách sử dụng ?

1. Trigger là gì?

Trigger trong MySQL là một thủ tục tự động được kích hoạt (triggered) khi có một sự kiện nào đó xảy ra trên một bảng (table) nhất định, như khi dữ liệu trong bảng được thêm mới, cập nhật hoặc xóa. Triggers thường được sử dụng để thực thi các quy tắc nghiệp vụ hoặc thực hiện các thao tác dữ liệu tự động khi một sự kiện cụ thể xảy ra.

2. Trigger dùng để làm gì?

Trigger thường được sử dụng trong các tình huống sau:

* Duy trì tính toàn vẹn dữ liệu: Đảm bảo rằng các quy tắc hoặc ràng buộc dữ liệu được tuân thủ khi các thao tác thêm, sửa hoặc xóa dữ liệu xảy ra.
* Tự động ghi log: Ghi lại lịch sử thay đổi dữ liệu trong bảng, ví dụ, ghi log mỗi khi dữ liệu được cập nhật hoặc xóa.
* Đồng bộ hóa dữ liệu: Tự động đồng bộ hóa dữ liệu giữa các bảng hoặc các hệ thống khác khi có thay đổi.
* Thực thi ràng buộc nghiệp vụ: Đảm bảo rằng các quy tắc nghiệp vụ được tuân thủ, ví dụ: kiểm tra điều kiện trước khi cho phép cập nhật dữ liệu.

3. Cú pháp tạo Trigger

Cú pháp để tạo trigger trong MySQL:

CREATE TRIGGER trigger\_name

trigger\_time trigger\_event

ON table\_name

FOR EACH ROW

BEGIN

-- Các câu lệnh SQL thực thi khi trigger được kích hoạt

END;

Giải thích:

* trigger\_name: Tên của trigger.
* trigger\_time: Thời gian kích hoạt trigger (BEFORE hoặc AFTER).
* trigger\_event: Sự kiện kích hoạt trigger (INSERT, UPDATE, hoặc DELETE).
* table\_name: Tên của bảng mà trigger áp dụng.
* FOR EACH ROW: Đảm bảo trigger được thực thi cho mỗi hàng bị ảnh hưởng bởi sự kiện.

Ví dụ:

Tạo một trigger để tự động ghi log khi có một hàng mới được thêm vào bảng employees:

CREATE TRIGGER log\_insert\_employee

AFTER INSERT

ON employees

FOR EACH ROW

BEGIN

INSERT INTO employee\_logs (employee\_id, action, action\_time)

VALUES (NEW.id, 'INSERT', NOW());

END;

4. Trigger Time (Thời gian kích hoạt Trigger)

Trigger có thể được kích hoạt tại hai thời điểm:

* BEFORE: Trigger được kích hoạt trước khi hành động (INSERT, UPDATE, DELETE) thực sự xảy ra. Điều này hữu ích khi bạn muốn kiểm tra hoặc thay đổi dữ liệu trước khi hành động được thực hiện.

Ví dụ: BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE.

* AFTER: Trigger được kích hoạt sau khi hành động đã xảy ra. Thường dùng để thực hiện các tác vụ sau khi dữ liệu đã được thay đổi, như ghi log hoặc đồng bộ dữ liệu.

Ví dụ: AFTER INSERT, AFTER UPDATE, AFTER DELETE.

5. Trigger Event (Sự kiện kích hoạt Trigger)

Trigger có thể được kích hoạt bởi ba loại sự kiện:

* INSERT: Kích hoạt khi một hàng mới được thêm vào bảng.
* UPDATE: Kích hoạt khi một hàng trong bảng bị sửa đổi.
* DELETE: Kích hoạt khi một hàng trong bảng bị xóa.

6/ Transaction là gì ? Các tính chất của Transaction ? A.C.I.D ?

**Transaction (giao dịch)** trong hệ quản trị cơ sở dữ liệu là một đơn vị công việc logic, bao gồm một hoặc nhiều hoạt động (như thêm, xóa, sửa dữ liệu) thực hiện trên cơ sở dữ liệu. Một giao dịch đảm bảo rằng các thao tác này đều được thực hiện thành công hoặc nếu có lỗi, tất cả các thao tác sẽ bị hủy bỏ để cơ sở dữ liệu quay về trạng thái ban đầu trước khi giao dịch bắt đầu.

### **Các tính chất của Transaction - ACID**

**1/ Atomicity (Tính nguyên tử)**: Giao dịch phải được thực hiện toàn bộ hoặc không thực hiện gì cả. Nếu bất kỳ phần nào của giao dịch thất bại, toàn bộ giao dịch sẽ bị hủy bỏ và cơ sở dữ liệu phải trở lại trạng thái ban đầu, giống như chưa có giao dịch nào xảy ra.

**2/ Consistency (Tính nhất quán)**: Sau khi giao dịch kết thúc, cơ sở dữ liệu phải ở trạng thái hợp lệ. Các ràng buộc (constraints) và quy tắc (rules) của cơ sở dữ liệu không bị vi phạm. Điều này có nghĩa là giao dịch không được làm hỏng tính toàn vẹn của cơ sở dữ liệu.

**3/ Isolation (Tính độc lập)**: Các giao dịch khác nhau được thực hiện độc lập với nhau. Một giao dịch không được phép nhìn thấy kết quả tạm thời của một giao dịch khác cho đến khi giao dịch đó hoàn thành. Tính chất này đảm bảo rằng các giao dịch chạy song song sẽ không gây ảnh hưởng tiêu cực đến nhau.

**Durability (Tính bền vững)**: Khi một giao dịch đã hoàn tất (được commit), mọi thay đổi của giao dịch đó phải được ghi vào cơ sở dữ liệu một cách vĩnh viễn, ngay cả khi có sự cố xảy ra như hệ thống bị lỗi hoặc mất điện. Điều này đảm bảo rằng dữ liệu sẽ không bị mất sau khi giao dịch đã hoàn thành.

### **$$ Mục tiêu và ý nghĩa của ACID**

Các tính chất ACID đảm bảo rằng cơ sở dữ liệu duy trì tính toàn vẹn và an toàn khi thực hiện các giao dịch, đặc biệt là trong môi trường xử lý nhiều giao dịch đồng thời. Khi các giao dịch tuân theo ACID, người dùng có thể tin tưởng vào sự nhất quán và đáng tin cậy của dữ liệu, bất kể có sự cố xảy ra trong quá trình giao dịch.