BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN TỬ - VIỄN THÔNG

-----□□&□□-----

Báo cáo Đồ Án

Tìm hiểu hệ quản trị MySQL

Giảng viên hướng dẫn :

GV. ThS. Lê Đức Trị

Nhóm 2

- **19200349 – Nguyễn Đình Khôi**
- **19200421 – Nguyễn Hồng Phát**
- **19200495 – Huỳnh Chí Thật**

Mục lục

1. **Định nghĩa về MySQL**
   - MySQL là chương trình dùng để quản lý hệ thống cơ sở dữ liệu (CSDL), trong đó CSDL là một hệ thống lưu trữ thông tin. được sắp xếp rõ ràng, phân lớp ngăn nắp những thông tin mà mình lưu trữ.
   - Vì thế, bạn có thể truy cập dữ liệu một cách thuận lợi, nhanh chóng. MySQL hỗ trợ đa số các ngôn như lập trình. Chính vì thế mà mã nguồn mở phổ biến nhất hiện nay là WordPress đã sử dụng MySQL làm cơ sở dữ liệu mặc định.
2. **Tổng quan về hệ quản trị MySQL**
   - Lưu ý rằng MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở, chính vì vậy mà nó chỉ hỗ trợ những ngôn ngữ theo hướng "mở", các mã nguồn như C++ sẽ không thể sử dụng MySQL cho những dự án của mình, ngoài ra thì theo công ty từ dự án bugnetproject của chính Microsoft thì họ cũng đã xác nhận rằng ngôn ngữ C++ hay .Net Development sẽ không hỗ trợ trên nền tảng mySQL.

2.1. Tại sao lại sử dụng MySQL?

2.1.1. Khả năng mở rộng và tính linh hoạt

Máy chủ cơ sở dữ liệu MySQL đáp ứng nhiều tính năng linh hoạt, nó có sức chứa để xử lý các ứng dụng được nhúng sâu với 1MB dung lượng để chạy kho dữ liệu khổng lồ lên đến hàng terabytes thông tin. Đặc tính đáng chú ý của MySQL là sự linh hoạt về flatform với tất cả các phiên bản của Windows, Unix và Linux đang được hỗ trợ. Và đương nhiên, tính chất mã nguồn mở của MySQL cho phép tùy biến theo ý muốn để thêm các yêu cầu phù hợp cho database server.

2.1.2. Hiệu năng cao

Với kiến trúc storage-engine cho phép các chuyên gia cơ sở dữ liệu cấu hình máy chủ cơ sở dữ liệu MySQL đặc trưng cho các ứng dụng chuyên biệt. Dù ứng dụng là website dung lượng lớn phục vụ hàng triệu người/ngày hay hệ thống xử lý giao dịch tốc độ cao thì MySQL đều đáp ứng được khả năng xử lý khắt khe của mọi hệ thống. Với những tiện ích tải tốc độ cao, cơ chế xử lý nâng cao khác và đặc biệt bộ nhớ caches, MySQL đưa ra tất cả nhưng tính năng cần có cho hệ thống doanh nghiệp khó tính hiện nay.

2.1.3. Tính sẵn sàng cao

MySQL đảm bảo sự tin cậy và có thể sử dụng ngay. MySQL đưa ra nhiều tùy chọn có thể "mì ăn liền" ngay từ cấu hình tái tạo chủ/tớ tốc độ cao, để các nhà phân phối thứ 3 có thể đưa ra những điều hướng có thể dùng ngay duy nhất cho server cơ sở dữ liệu MySQL.

2.1.4. Hỗ trợ giao dịch mạnh mẽ

MySQL đưa ra một trong số những engine giao dịch cơ sở dữ liệu tốt nhất trên thị trường. Các đặc trưng bao gôm, khóa mức dòng không hạn chế,

hỗ trợ giao dịch ACID hoàn thiện, khả năng giao dịch được phân loại và hỗ trợ giao dịch đa dạng (multi-version) mà người đọc không bao giờ cản trở cho người viết và ngược lại. Dữ liệu được đảm bảo trong suốt quá trình server có hiệu lực, các mức giao dịch độc lập được chuyên môn hóa, khi phát hiện có lỗi khóa chết ngay tức thì.

2.1.5. Điểm mạnh của Web và Data Warehouse

Theo công ty thiết kế website Mona Media thì MySQL là nơi cho các website trao đổi thường xuyên bởi nó có engine xử lý tốc độ cao, khả năng chèn dữ liệu cực nhanh và hỗ trợ mạnh các chức năng chuyên dụng của web. Các tính năng này cũng được sử dụng cho môi trường lưu trữ dữ liệu mà MySQL tăng cường đến hàng terabyte cho các server đơn. Những tính năng khác như chỉ số băm, bảng nhớ chính, bảng lưu trữ và cây B đã được gói lại để giảm các yêu cầu lưu trữ tới 80%. Vì thế, MySQL là sự chọn lựa tốt nhất cho cả ứng dụng web và các ứng dụng của doanh nghiệp.

2.1.6. Bảo vệ dữ liệu mạnh mẽ

Việc quan trong của các doanh nghiệp là bảo mật dữ liệu, MySQL tích hợp các tính năng bảo mật an toàn tuyệt đối. Với việc xác nhận truy cập cơ sở dữ liệu, MySQL trang bị các kỹ thuật mạnh, chỉ có người sử dụng đã được xác nhận mới truy cập được vào cơ sở dữ liệu. Chứng chỉ SSH và chứng chỉ SSL cũng được hỗ trợ để đảm bảo kết nối an toàn và bảo mật. Tiện ích backup và recovery cung cấp bởi MySQL và các hãng phần mềm thứ 3 cho phép backup logic và vật lý cũng như recovery toàn bộ hoặc tại một thời điểm.

2.1.7. Phát triển ứng dụng toàn diện

MySQL trở thành cơ sở dữ liệu mã nguồn mở phổ biến nhất hiện nay một phần là do cung cấp hỗ trợ hỗn hợp cho bất cứ sự phát triển ứng dụng nào cần. Trong cơ sở dữ liệu, hỗ trợ có thể được tìm thấy trong các trigger, stored procedure, cursor, view, ANSI-standard SQL,… MySQL cũng cung cấp các bộ kết nối như: JDBC, ODBC,… để tất cả các form của ứng dụng sử dụng MySQL như một erver quản lí dữ liệu được đề xuất hàng đầu.

2.1.8. Quản lý dễ dàng

Cài đặt MySQL khá nhanh và trung binh từ khi download phần mềm tới khi cài đặt thành công chỉ mất chưa đầy 15 phút. Cho dù flatform là Linux, Microsoft Windows, Macintosh hoặc Unix thì quá trình cũng diễn ra nhanh chóng. Khi đã cài đặt, tính năng quản lý như tự khởi động lại, tự động mở rộng không gian và cấu hình động sẵn sàng cho người quản trị cơ sở dữ liệu bắt đầu làm việc. MySQL cung cấp toàn bộ công cụ quản

lý đồ họa cho phép một DBA quản lý, sửa chữa và điều khiển hoạt động của nhiều server MySQL từ một máy trạm đơn.

2.1.9. Mã nguồn mở tự do và hỗ trợ 24/7

Nhiều doanh nghiệp còn băn khoăn trong việc giao toàn bộ cho phần mềm mã nguồn mở bởi khó có thể tìm được hỗ trợ hay bảo mật an toàn phục vụ chuyên nghiệp. Với MySQL mọi sự cam kết đều rõ ràng, MySQL cam kết bồi thường khi gặp sự cố.

2.1.10. Chi phí sở hữu thấp nhất

Sử dụng MySQL cho các dự án, doanh nghiệp đều nhận thấy sự tiết kiệm chi phí đáng kể. Người dùng MySQL cũng không mất nhiều thời gian để sữa chữa hoặc vấn đề thời gian chết.

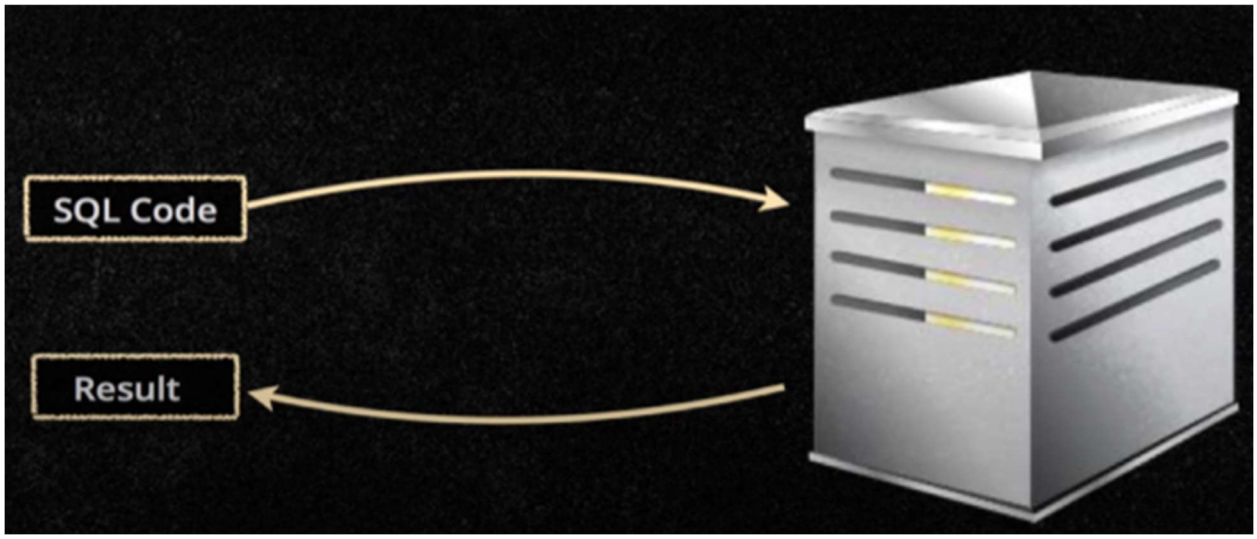
2.2. Chi tiết hoạt động chu kì của một SQL

2.2.1. Các bước thực hiện

- Bước 1: Viết một câu lệnh SQL
- Bước 2: Câu lệnh sẽ gửi lên RDBMS
- Bước 3: Trả về kết quả

2.2.2. Thông tin thêm

- Mỗi RDBMS cung cấp cho ta các công cụ để truy cập, kết nối và viết các câu lệnh SQL
- Các công cụ này thường có giao diện người dung, khác nhau đối với từng RDBMS.



2.3. Cài đặt mySQL

2.3.1. Các phần mềm cần thiết phải có:

- **XAMP**: chứa hệ quản trị cơ sở dữ liệu mySQL server
- **MySQL Workbench**: viết câu lệnh mySQL

2.3.2. Các bước để tạo một hệ quản trị cơ sở dữ liệu mySQ

- Bước 1: Mở phần mềm XAMP và chọn **Start mySQL server**
- Bước 2: Kết nối mySQL server bằng cách mở MySQL Workbench
- Bước 3: Tạo database và bảng cần thiết (thao tác hoặc code)
- Bước 4: Truy suất dữ liệu với các câu lệnh select,…

2.4. Ngôn ngữ mySQL

- Là ngôn ngữ dễ học dễ nhớ, cấu trúc đơn giản
- mySQL không phân biệt chữ hoa và thường
  VD: **khach_hang** và **KHACH_HANG** là như nhau
- Các câu lệnh giống nhau cho các RDBMS, về mySQL thì có đôi chút khác so với các loại RDBMS khác (ngôn ngữ khác).

2.5. Các lệnh cần thiết trong mySQL

2.5.1. Lệnh MySQL quản trị

Đây là tập hợp các lệnh MySQL quan trọng nhất mà bạn có thể gặp phải khi làm việc với cơ sở dữ liệu MySQL.

- **USE <DATANAMESPACE>** : Để chọn một cơ sở dữ liệu trong vùng làm việc MySQL, hãy nhập tên cơ sở dữ liệu.
- **SHOW DATABASE** hiển thị danh sách cơ sở dữ liệu mà MySQL DBMS có thể truy cập.
- **SHOW TABLES** hiển thị các bảng cơ sở dữ liệu sau khi sử dụng lệnh sử dụng để chọn cơ sở dữ liệu.
- **SHOW COLUMNS FROM <TÊN BẢNG >:** Hiển thị các thuộc tính của bảng, loại thuộc tính, thông tin khóa, nếu giá trị NULL được cho phép, giá trị mặc định và dữ liệu khác.
- **SHOW INDEX FROM <TÊN BẢNG>** : hiển thị thông tin về tất cả các chỉ mục của bảng, bao gồm cả Primary Key.
- **SHOW TABLE STATUS LIKE <TÊN BẢNG>:** hiển thị thông tin về hiệu suất và thống kê của hệ thống quản lý cơ sở dữ liệu MySQL.

2.5.2. Các kiểu dữ liệu MySQL

Định nghĩa thích hợp về các trường của bảng rất quan trọng đối với việc tối ưu hóa tổng thể cơ sở dữ liệu của bạn. Bạn chỉ nên sử dụng loại và kích thước của trường mà bạn thực sự yêu cầu. Ví dụ: nếu bạn biết mình sẽ chỉ sử dụng hai ký tự, đừng tạo trường lớn 10 ký tự. Sau khi sắp xếp dữ liệu, bạn sẽ lưu trữ trong các trường đó, các trường (hoặc cột) này được gọi là kiểu dữ liệu.

MySQL sử dụng nhiều kiểu dữ liệu khác nhau được chia thành ba nhóm:

2.5.2.1. Kiểu dữ liệu số

- MySQL hỗ trợ tất cả các kiểu dữ liệu số ANSI SQL tiêu chuẩn, do đó, các định nghĩa này sẽ dễ nhận biết nếu bạn đến từ một hệ thống cơ sở dữ liệu khác.
- Các kiểu dữ liệu số phổ biến và mô tả của chúng được liệt kê bên dưới.
    - INT: là một số nguyên có dấu hoặc không dấu có kích thước chuẩn. Phạm vi được phép, nếu có dấu, là -2147483648 đến 2147483647. Nếu giá trị không có dấu, phạm vi là 0 đến 4294967295. Có thể chỉ định độ rộng của tối đa 11 chữ số.
    - TINYINT: là một số nguyên có dấu hoặc không dấu rất nhỏ. Phạm vi được phép, nếu có dấu, là -128 đến 127. Nếu giá trị không có dấu, phạm vi là 0 đến 255. Có thể chỉ định độ rộng của tối đa bốn chữ số.
    - SMALLINT: Đây là một số nguyên nhỏ có thể có dấu hoặc không dấu. Phạm vi được phép, nếu có dấu, là -32768 đến 32767. Nếu giá trị không được ký, phạm vi là 0 đến 65535. Có thể chỉ định độ rộng của tối đa 5 chữ số.
    - MEDIUMINT: Đây là một số nguyên cỡ vừa có dấu hoặc không dấu. Phạm vi được phép, nếu có dấu, là -8388608 đến 8388607. Nếu giá trị không được ký, phạm vi là 0 đến 16777215. Có thể chỉ định độ rộng của tối đa 9 chữ số.
    - BIGINT: Kích thước của một số nguyên rất lớn, có thể có dấu hoặc không dấu. Phạm vi được phép nếu có dấu là -9223372036854775808 đến 9223372036854775807. Nếu giá trị không được ký, phạm vi là 0 đến 18446744073709551615. Có thể chỉ định độ rộng của tối đa 20 chữ số
    - FLOAT (M, D): Đây là một số dấu phẩy động có dấu không thể không dấu. Cả chiều dài hiển thị (M) và số lượng số thập phân đều có thể được tùy chỉnh (D). Đây là tùy chọn; giá trị mặc định là 10,2, trong đó 2 biểu thị số lượng phần thập phân và 10 biểu thị tổng số chữ số (bao gồm cả số thập phân). Đối với FLOAT, độ chính xác thập phân có thể lên đến 24 chữ số.
    - DOUBLE (M, D): Một số dấu phẩy động có độ chính xác gấp đôi không thể không dấu. Cả chiều dài hiển thị (M) và số lượng số thập phân đều có thể được tùy chỉnh

(D). Đây là tùy chọn; giá trị mặc định là 16,4, trong đó 4 là số thập phân. Đối với DOUBLE, độ chính xác thập phân có thể lên tới 53 chữ số. DOUBLE là một từ đồng nghĩa với REAL.

- DECIMAL (M, D): Một số dấu phẩy động có dấu đã được giải nén. Mỗi số thập phân trong số thập phân được giải nén tương ứng với một byte. Cần xác định độ dài hiển thị (M) và số phần thập phân (D). DECIMAL là từ đồng nghĩa với NUMERIC.

2.5.2.2. Kiểu ngày và giờ

Sau đây là các kiểu dữ liệu ngày và giờ của MySQL:

- DATE: Ngày từ 1000-01-01 đến 9999-12-31 ở định dạng YYYY-MM-DD. Ví dụ, ngày 30 tháng 12 năm 1973 sẽ được giữ là 1973-12-30.
- DATETIME: Ngày và giờ kết hợp giữa 1000-01-01 00:00:00 và 9999-12-31 23:59:59 ở định dạng YYYY-MM-DD HH: MM: SS. Ví dụ: vào ngày 30 tháng 12 năm 1973, 3:30 chiều sẽ được giữ là 1973-12-30 15:30:00.
- TIMESTAMP: Ngày từ nửa đêm ngày 1 tháng 1 năm 1970 đến năm 2037. Định dạng này có vẻ giống với định dạng DATETIME trước đó, nhưng không có dấu gạch ngang giữa các chữ số; ví dụ: 3:30 chiều ngày 30 tháng 12 năm 1973 sẽ được giữ là 19731230153000. (YYYYMMDDHHMMSS).
- TIME: TIME lưu thời gian ở định dạng HH: MM: SS.
- YEAR (M): Lưu trữ một năm dưới dạng số có hai chữ số hoặc bốn chữ số. YEAR có thể là bất kỳ năm nào trong khoảng từ 1970 đến 2069 với điều kiện độ dài được chỉ định là 2 (ví dụ, YEAR (2)) (70 đến 69). YEAR có thể nằm trong khoảng từ 1901 đến 2155 nếu độ dài được cho là 4. Độ dài mặc định là 4 ký tự.

2.5.2.3. Kiểu String

Mặc dù các định dạng số và ngày đều mang tính giải trí nhưng phần lớn dữ liệu của bạn sẽ được lưu trữ ở định dạng chuỗi. Sau đây là danh sách các kiểu dữ liệu chuỗi phổ biến nhất trong MySQL.

- CHAR (M): Một chuỗi có độ dài cố định (ví dụ: CHAR (5)) có độ dài từ 1 đến 255 ký tự, được đệm bên phải với các khoảng trắng có độ dài được chỉ định khi được lưu trữ. Tuy nhiên, không cần thiết phải chỉ định độ dài, mặc định là 1.

- VARCHAR (M): Một chuỗi có độ dài thay đổi với độ dài từ 1 đến 255 ký tự. VARCHAR, chẳng hạn (25). Khi tạo trường VARCHAR, bạn phải chỉ định độ dài.
- BLOB HOẶC TEXT: Một trường có thể chứa tối đa 65535 ký tự. "Đối tượng khổng lồ nhị phân" hay BLOB được sử dụng để lưu trữ khối lượng lớn dữ liệu nhị phân, chẳng hạn như ảnh hoặc các loại tệp khác. Các trường TEXT cũng có thể mang nhiều thông tin. Sự khác biệt giữa hai điều này là trên BLOB, việc sắp xếp và so sánh trên dữ liệu được lưu trữ có phân biệt chữ hoa chữ thường, trong khi trong trường TEXT thì không. Với BLOB và TEXT, bạn không cần phải cung cấp độ dài.
- TINYBLOB hoặc TINY TEXT: Độ dài tối đa 255 ký tự cho cột BLOB hoặc TEXT. Với TINYBLOB và TINYTEXT, bạn không phải cung cấp độ dài.
- MEDIUM BLOB hoặc MEDIUM TEXT: Độ dài tối đa của cột BLOB hoặc TEXT là 16777215 ký tự. Với MEDIUMBLOB hoặc MEDIUM TEXT, bạn không phải chỉ định độ dài.
- LONG BLOB hoặc LONG TEXT: Độ dài tối đa của cột BLOB hoặc TEXT là 4294967295 ký tự. Với LONGBLOB và LONGTEXT, bạn không phải xác định độ dài
- ENUM: Một danh sách được gọi là một kiểu liệt kê. Bạn đang thiết lập một danh sách các đối tượng mà từ đó giá trị phải được chọn khi bạn xác định một ENUM (hoặc nó có thể là NULL). Nếu bạn muốn trường của mình giữ "A," "B" hoặc "C", bạn sẽ xác định ENUM của mình là ENUM ('A,' B, 'C,)

2.5.3. Các loại lệnh cơ sở dữ liệu

Trong MySQL, chủ yếu có 5 loại lệnh cơ sở dữ liệu để lưu trữ và trích xuất dữ liệu.

2.5.3.1. DDL - Ngôn ngữ định nghĩa dữ liệu

DDL hoặc Ngôn ngữ Định nghĩa Dữ liệu bao gồm các lệnh bao gồm việc xác định cấu trúc của lược đồ cơ sở dữ liệu và bảng cơ sở dữ liệu. Nó chủ yếu liên quan đến cấu trúc của lược đồ cơ sở dữ liệu và bảng.

Các lệnh bao gồm trong DDL là:

Create - Được sử dụng để tạo lược đồ bảng

Drop - Bỏ cơ sở dữ liệu khỏi bộ nhớ
Alter - Thay đổi cấu trúc của lược đồ bảng
Truncate - Xóa tất cả dữ liệu khỏi lược đồ cơ sở dữ liệu
Comment - Các câu lệnh này chỉ nhằm mục đích hiểu cấu trúc lược đồ. Chúng không đóng góp vào cấu trúc cơ sở dữ liệu thực tế
Rename - Được sử dụng để đổi tên bảng cơ sở dữ liệu

2.5.3.2. DML - Ngôn ngữ thao tác dữ liệu

DML hoặc Ngôn ngữ thao tác dữ liệu bao gồm các lệnh bao gồm thao tác dữ liệu của bảng cơ sở dữ liệu. Nó liên quan đến việc tạo và thao tác dữ liệu của bảng cơ sở dữ liệu.
Các lệnh có trong DML là:
Select: Được sử dụng để trích xuất thông tin từ bảng cơ sở dữ liệu
Insert: Dùng để chèn dữ liệu vào bảng cơ sở dữ liệu
Update: Cập nhật dữ liệu hiện có trong bảng cơ sở dữ liệu nơi một điều kiện cụ thể được xác định.
Delete: Xóa dữ liệu khỏi bảng cơ sở dữ liệu

2.5.3.3. DCL - Ngôn ngữ điều khiển dữ liệu

DCL hoặc Ngôn ngữ điều khiển dữ liệu bao gồm các lệnh đó để xử lý các quyền, quyền hạn hoặc các điều khiển khác của hệ thống cơ sở dữ liệu. Nó liên quan đến các quyền cụ thể đối với những người dùng cụ thể để truy cập vào cơ sở dữ liệu.
Các lệnh có trong DCL là:
Grant: Được sử dụng để cấp quyền cho một người dùng cụ thể truy cập vào cơ sở dữ liệu với quyền đọc hoặc ghi.
Revoke: Được sử dụng để xóa quyền truy cập cơ sở dữ liệu của một người dùng cụ thể.

2.5.3.4. TCL - Ngôn ngữ kiểm soát giao dịch

TCL hoặc Ngôn ngữ điều khiển giao dịch bao gồm các lệnh xử lý các giao dịch của cơ sở dữ liệu, bao gồm tập hợp các câu lệnh giữa máy khách và máy chủ cụ thể.
Các lệnh có trong TCL là:
Commit: Dùng để lưu trữ dữ liệu trước đó.
Rollback: Được sử dụng để khôi phục dữ liệu hoặc bảng đã xóa trước đó.
Savepoint: Dùng để tạo một điểm trong cơ sở dữ liệu hay còn gọi là checkpoint. Nó được sử dụng để khôi phục giao dịch.

2.5.3.5. KEYS – Khoá

Bất cứ khi nào chúng tôi tạo dữ liệu trong cơ sở dữ liệu, sẽ có khả năng xảy ra các mục trùng lặp trong bảng. Chúng tôi có thể xác định duy nhất các giá trị / bản ghi này từ cơ sở dữ liệu bằng cách sử dụng các cột cụ thể hoặc kết hợp các cột có bản chất duy nhất còn được gọi là 'khóa'.

Chủ yếu có 5 loại khóa trong cơ sở dữ liệu:

- Primary Key: Một thuộc tính có thể được sử dụng để xác định từng bộ giá trị duy nhất trong bảng kết quả được gọi là khóa chính. Có thể chỉ có một khóa chính duy nhất trong bảng.
- Candidate Key: Một tập hợp tối thiểu các thuộc tính có thể xác định duy nhất các bộ giá trị trong bản ghi được gọi là khóa ứng viên. Có thể có nhiều hơn một khóa ứng viên trong bảng còn được gọi là khóa tổng hợp.
- Super Key: Một tập hợp các thuộc tính có thể xác định duy nhất một bộ dữ liệu trong bản ghi được gọi là siêu khóa. Vì vậy, một khóa ứng viên là một siêu khóa nhưng ngược lại không phải lúc nào cũng đúng.
- Alternate Key: Khóa ứng viên không phải là khóa chính được gọi là khóa thay thế.
- Foreign Key: Một thuộc tính chỉ có thể nhận các giá trị hiện tại dưới dạng giá trị hiện tại làm giá trị của một số thuộc tính khác, là khóa ngoại của thuộc tính mà nó tham chiếu đến.

2.5.3.6. Hạn chế

Bất cứ khi nào chúng ta xác định cấu trúc của cơ sở dữ liệu, chúng ta có thể tạo các ràng buộc trên cột để đảm bảo các điều kiện nhất định được đáp ứng trước khi giá trị được chấp nhận trong cột đó. Dưới đây được đề cập là một số ràng buộc:

- NOT NULL: Được sử dụng để đảm bảo rằng giá trị null không thể được lưu trữ trong một cột
- UNIQUE: Được sử dụng để đảm bảo rằng tất cả các giá trị được lưu trữ trong một cột là duy nhất.
- CHECK: Được sử dụng để đảm bảo rằng tất cả các giá trị thỏa mãn một điều kiện cụ thể trước khi chúng có thể được chấp nhận trong một cột
- DEFAULT: Được sử dụng để xác định một tập hợp các giá trị mặc định khi không có giá trị nào cho một cột được chỉ định.

- INDEX: Được sử dụng để lưu trữ và truy xuất dữ liệu đến và đi từ cơ sở dữ liệu rất nhanh chóng.
  Sắp xếp kết quả:
  - ✓ Có nhiều lý do tại sao chúng ta cần sắp xếp kết quả như sau:
  - ✓ Dữ liệu, mà bạn muốn sắp xếp, sẽ không phải lúc nào cũng có trong cơ sở dữ liệu (SQL trong trường hợp này). Nếu bạn lấy một số dữ liệu từ người dùng làm đầu vào nhưng muốn thực hiện sắp xếp trước khi thực hiện bất kỳ thao tác nào thì bạn phải sắp xếp tại chỗ trong mã của chúng tôi.
  - ✓ Đôi khi logic sắp xếp của bạn có thể rất khó (kết hợp nhiều trường và hoạt động). Dữ liệu của chúng tôi có thể trải dài trên nhiều bảng. Trong các vấn đề của thế giới thực, loại dữ liệu mà chúng ta sẽ nhận được sẽ không đơn giản. Nó sẽ không được sắp xếp theo thứ tự tăng dần hoặc giảm dần. Nó sẽ rất nhiều không được sắp xếp. Đôi khi chúng tôi có thể phải thực hiện một số hành động trước khi sắp xếp. Logic sắp xếp của chúng tôi có thể cần nhiều trường và nhiều thao tác trước khi sắp xếp. Hãy lấy một ví dụ. Chúng ta có 2 cột trong bảng A (col1, col2) và 1 cột trong bảng B (col3). Vì một số lý do chúng ta cần sắp xếp thứ tự dựa trên (col1 * Col2 + col3). (Như bài toán về phương trình đường thẳng mx + c). Bây giờ chúng ta không thể thực hiện bằng cách sử dụng các lệnh của SQL một cách dễ dàng. Đó là nơi chúng tôi sẽ cần phân loại trong mã.
  - ✓ Trong một số trường hợp, việc sắp xếp sẽ nhanh hơn nếu được thực hiện bằng mã dựa trên tập dữ liệu của chúng tôi. Chúng tôi có thể cần các thuật toán sắp xếp khác nhau cho hiệu suất dựa trên tập dữ liệu của chúng tôi. Mọi thứ phụ thuộc vào dữ liệu của chúng tôi.
  - ✓ Chúng tôi cần nghiên cứu các thuật toán sắp xếp để cải thiện việc thiết kế thuật toán của bạn và tìm hiểu các cách tiếp cận và mẫu khác nhau mà bạn sẽ thực sự cần trong thế giới CNTT. Chúng tôi sẽ không bao giờ viết sắp xếp bong bóng hoặc sắp xếp đống trong bất kỳ công ty nào như nó vốn có. Bạn chỉ cần biết

logic. Tất cả các ngôn ngữ lập trình đều có các hàm sắp xếp đã được triển khai mà bạn chỉ cần gọi cho mục đích sắp xếp. Chúng sẽ nhanh hơn và được kiểm tra tốt.

- ✓ Vì vậy, chúng ta cần biết tất cả các cách tiếp cận khác nhau và các giải pháp khả thi để giải quyết một vấn đề. Một giải pháp không thể tốt nhất trong mọi trường hợp. Tùy thuộc vào dữ liệu và bối cảnh, bạn sẽ cần các cách tiếp cận khác nhau. Vì vậy, chúng ta cần khám phá tất cả các giải pháp khả thi và sử dụng chúng dựa trên nhu cầu của mình.

- *Select:*

- ✓ Bất cứ khi nào chúng tôi tìm nạp kết quả từ cơ sở dữ liệu bằng cách sử dụng truy vấn SQL Select, chúng tôi sẽ nhận được kết quả theo kiểu không có thứ tự.
- ✓ VD: Select * from employees;
- ✓ Câu lệnh này tìm nạp tất cả các kết quả từ bảng nhân viên nhưng theo kiểu đặt hàng trước. Nếu chúng ta muốn kết quả được hiển thị theo thứ tự, trong trường hợp đó, chúng ta phải thực hiện các bước tiếp theo.

- *Order by:*

- ✓ Chúng ta có thể sắp xếp kết quả trong SQL Select bằng mệnh đề ORDER BY và xác định tên cột trên cơ sở chúng ta muốn sắp xếp kết quả.
- ✓ VD: Select * from employees order by emp_id;
- ✓ Câu lệnh này tìm nạp kết quả từ bảng nhân viên và sắp xếp kết quả trên cơ sở cột emp_id theo thứ tự tăng dần.
- ✓ Theo mặc định, kết quả thu được theo thứ tự tăng dần.
- ✓ Desc:

  Chúng tôi cũng có thể sắp xếp các kết quả theo thứ tự giảm dần bằng cách sử dụng từ khóa “desc”. Sử dụng từ khóa desc, tất cả các kết quả được hiển thị theo thứ tự giảm dần bắt đầu từ giá trị cao nhất được hiển thị ở trên cùng và giá trị thấp nhất được hiển thị ở dưới cùng.

  Ví dụ: Select * from employees order by emp_id desc;
- ✓ Sorting by multiple columns:

Nếu có hai hoặc nhiều kết quả giống nhau thu được trong một nhóm, trong trường hợp đó, chúng ta có thể sử dụng lập chỉ mục ưa thích trong đó kết quả thu được bằng cách sắp xếp kết quả trên cơ sở cột đầu tiên và nếu kết quả trùng nhau, thì chúng ta có thể sắp xếp kết quả trên cơ sở của cột thứ hai.

- ✓ VD: Select * from employees order by emp_id,salary;
  Khi chúng tôi cố gắng sắp xếp kết quả dựa trên 2 cột, trong trường hợp đó, kết quả được sắp xếp ban đầu trên cơ sở emp_id cột đầu tiên và sau đó dựa trên mức lương của cột thứ hai.
- ✓ Limit:
  Chúng tôi cũng có thể hiển thị kết quả bằng cách giới hạn số hàng được hiển thị bằng cách sử dụng hàm LIMIT.
  Ví dụ: select * from employees order by emp_id limit 3;
  Câu lệnh này hiển thị 3 hàng trên cùng từ bảng nhân viên sắp xếp kết quả trên cơ sở bảng emp_id theo thứ tự tăng dần.
- ✓ Offset:
  Bất cứ khi nào chúng tôi sử dụng hàm LIMIT, chúng tôi sẽ hiển thị các cột 'n' trên cùng trong kết quả của chúng tôi. Chúng ta có thể hiển thị số hàng chính xác bỏ qua một tập hợp hàng cụ thể từ trên cùng bằng cách sử dụng hàm offset.
  Vd . Select * from employees order by emp_id limit 3 offset 2;
  Câu lệnh này sẽ hiển thị kết quả của tất cả nhân viên từ bảng nhân viên được sắp xếp theo emp_id theo thứ tự tăng dần hiển thị 3 hàng trên cùng sau khi bỏ qua 2 hàng đầu tiên.
  Chúng tôi sắp xếp kết quả từ cơ sở dữ liệu của mình bất cứ khi nào chúng tôi muốn hiển thị kết quả từ định dạng không có cấu trúc sang định dạng có cấu trúc.
  Bất cứ khi nào chúng tôi hiển thị kết quả trong một trang web hoặc một trang web bằng cách trích xuất kết quả từ Cơ sở dữ liệu SQL, thì chúng tôi phải hiển

thị kết quả theo cách có thứ tự, ví dụ. hiển thị kết quả của học sinh.

Ở đó, chúng tôi sử dụng chức năng sắp xếp để hiển thị kết quả một cách có thứ tự.

Vì không phải tất cả dữ liệu đều được lưu trữ trong Cơ sở dữ liệu SQL, một số được lưu trữ trong hệ thống tệp và một số cũng được lưu trữ trong thư mục, vì vậy chúng ta cần tìm hiểu các thuật toán sắp xếp khác nhau để hiển thị kết quả theo thứ tự.

2.5.3.7. Liên kết các bảng

Bất cứ khi nào chúng ta muốn hiển thị kết quả từ nhiều bảng với các giá trị chung, chúng ta có thể nối các bảng trong SQL.

SQL hỗ trợ các kiểu nối sau trong bảng:

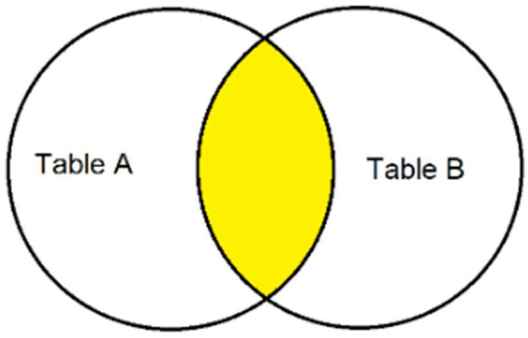
2.5.3.7.1. Inner Join

Tham gia bên trong hoặc liên kết ngầm là cách đơn giản nhất để nối hai bảng. Bất cứ khi nào chúng ta muốn nối hai bảng, chúng ta chỉ cần chọn các cột chung giữa hai bảng mà các bảng có thể được nối với nhau.

Phép nối bên trong trả về các bản ghi có giá trị phù hợp trong cả hai bảng.

Truy vấn sau đây hiển thị liên kết bên trong:

```
1  SELECT Orders.OrderID, Customers.CustomerName
2  FROM Orders
3  INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```



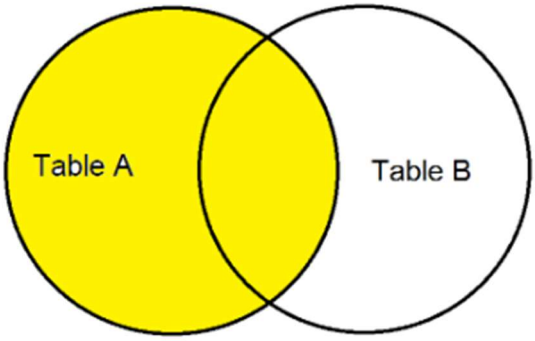
INNER JOIN

2.5.3.7.2. Left Join

Kết nối trái hoặc Nối ngoài trái là một kiểu nối ngoài trả về tất cả các bản ghi từ bảng bên trái và các bản ghi đã so khớp từ bảng bên phải.

Truy vấn sau đây hiển thị kết nối bên trái:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```



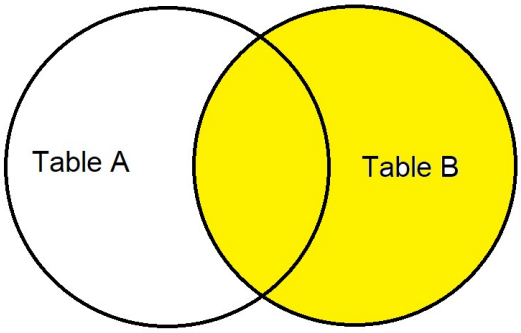
LEFT JOIN

2.5.3.7.3. Right Join

Phép nối bên phải hoặc phép nối ngoài bên phải là một loại phép nối trả về tất cả các bản ghi từ bảng bên phải và các bản ghi đã so khớp từ bảng bên trái

Truy vấn sau đây hiển thị Right Join:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```



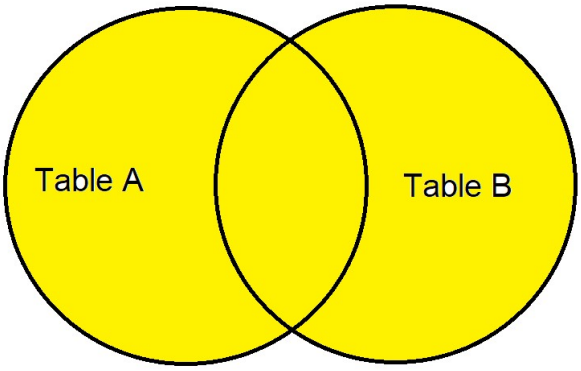
RIGHT JOIN

2.5.3.7.4. Full Join

Tham gia đầy đủ hoặc tham gia ngoài đầy đủ là một loại liên kết trả về tất cả các bản ghi khi có một kết quả phù hợp trong bảng bên trái hoặc bên phải

Truy vấn sau đây hiển thị tham gia đầy đủ:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```



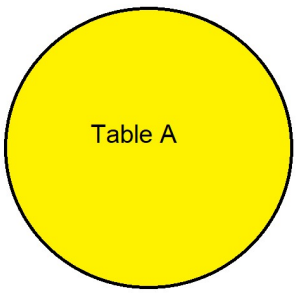
FULL OUTER JOIN

2.5.3.7.5. Self Join

Tự nối là một kiểu liên kết trong đó một bảng được nối với chính nó trong SQL.

Truy vấn sau đây hiển thị tự tham gia:

```
SELECT A.CustomerName AS CustomerName1, B.CustomerName AS   CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```



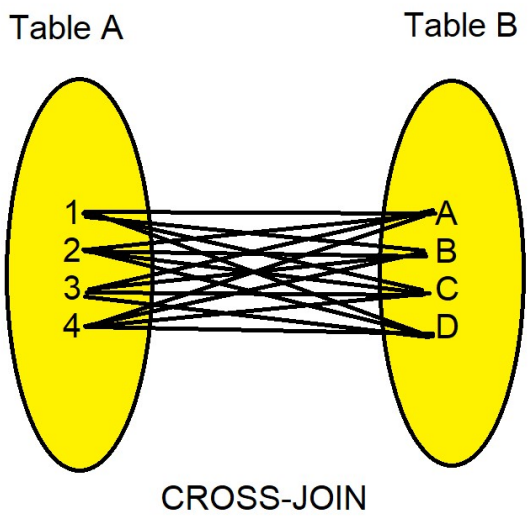
SELF JOIN

2.5.3.7.6. Cross Join

Nối chéo là kiểu liên kết trong SQL nơi mọi giá trị từ bảng 1 được nối với mọi giá trị khác trong bảng 2.

Truy vấn sau đây hiển thị kết hợp chéo:

```
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
CROSS JOIN company;
```



2.5.3.7.7. Equi-join

Trong SQL, Equi-join thực hiện phép nối với giá trị cột bằng nhau hoặc khớp với (các) cột của các bảng được liên kết. Trong phép nối tương đương, dấu bằng (=) được sử dụng như một toán tử so sánh trong mệnh đề where để chỉ sự bình đẳng của các bản ghi.

Chúng tôi cũng có thể thực hiện phép nối tương đương bằng cách sử dụng từ khóa JOIN và sau đó là từ khóa ON, sau đó chỉ định tên của các cột cùng với các bảng được liên kết của chúng để kiểm tra tính bình đẳng.

Truy vấn sau đây hiển thị liên kết tương đương:

```
SELECT column_list FROM table1, table2....WHERE table1.column_name =
table2.column_name;
```

✓ Equi-join vs Self-join:

Equijoin là kiểu nối với điều kiện nối có chứa toán tử đẳng (=). Một phép nối tương đương chỉ trả về những hàng có giá trị tương đương cho tập hợp cột được chỉ định.

Trong khi đó, phép nối bên trong là phép nối của hai hoặc nhiều bảng chỉ trả về các hàng đó (được so sánh bằng cách sử dụng toán tử so sánh) thỏa mãn điều kiện nối.

2.5.3.7.8. Natural Join

Trong SQL, Natural Join là một loại Equijoin và được cấu trúc theo cách mà các cột có cùng tên với các bảng được liên kết sẽ chỉ xuất hiện một lần.

Natural join và Inner join:

Sự khác biệt giữa phép nối bên trong và phép nối tự nhiên là số cột được trả về.

Giá trị NULL

Trong SQL, một cột có thể chứa giá trị NULL. Điều này có nghĩa là không tồn tại giá trị cho bản ghi cụ thể đó trong bảng. Chúng ta có thể kiểm tra sự tồn tại của giá trị NULL trong SQL bằng các lệnh sau:

- IS NULL - Truy vấn này trả về true nếu giá trị cột là NULL.
- IS NOT NULL - Truy vấn này trả về true nếu giá trị cột không phải là NULL. Ngược lại, nếu giá trị cột là NULL, giá trị false được trả về.
- <=> - Toán tử này so sánh các giá trị và trả về true cho 2 giá trị KHÔNG ĐỦ.

Truy vấn sau đây hiển thị các giá trị NULL trong SQL:

```
1 | SELECT * FROM table1 WHERE tutorial_count = NULL;
```

Tại đây, truy vấn trả về tất cả các bản ghi đó trong đó giá trị của cột tutorial_count = NULL

REGEXPS trong SQL :

Trong SQL, chúng ta có thể sử dụng Regexps để so khớp chuỗi với một chuỗi cụ thể hoặc một tập hợp con của chuỗi. Nó được sử dụng để đối sánh với mẫu để tìm một chuỗi cụ thể với mẫu đó.

Có 3 cách để sử dụng so sánh regexp trong SQL:

- Like
- Similar to
- Bộ so sánh Posix

❖ Like và Similar to : được sử dụng cho các so sánh cơ bản trong đó một chuỗi cụ thể được sử dụng để khớp với chuỗi phù hợp. Nếu mẫu được khớp, kết quả là chuỗi tương ứng được trả về.

✓ Truy vấn sau đây cho thấy việc sử dụng toán tử LIKE:

```
1 | Select * from employees where emp_name LIKE 'a%a';
```

❖ Với toán tử Posix, chúng tôi sử dụng các toán tử sau để khớp với kết quả:

✓ ~: Khớp, phân biệt chữ hoa chữ thường
✓ ~ *: Khớp, không phân biệt chữ hoa chữ thường
✓ ! ~: Không khớp, phân biệt chữ hoa chữ thường
✓ ! ~ *: Không trùng khớp, không phân biệt chữ hoa chữ thường
✓ Chúng tôi cũng có thể sử dụng toán tử REGEXP để khớp với mẫu. Dưới đây là bảng xác định danh sách các toán tử được sử dụng với các toán tử REGEXP để khớp với chuỗi mong muốn.
✓ ^: Bắt đầu của chuỗi
✓ $: Cuối chuỗi
✓ . : Bất kỳ ký tự đơn nào
✓ […]: Bất kỳ ký tự nào được liệt kê giữa 2 dấu ngoặc vuông
✓ [^…]: Bất kỳ ký tự nào không được liệt kê giữa 2 dấu ngoặc vuông
✓ p1 | p2 | p3: Thay thế, khớp với bất kỳ mẫu nào p1 hoặc p2 hoặc p3
✓ *: 0 hoặc nhiều trường hợp của phần tử trước
✓ +: 1 hoặc nhiều trường hợp của phần tử trước
✓ {x}: x phiên bản của phần tử trước
✓ {x, y}: x đến y các trường hợp của phần tử trước
✓ Chúng ta có thể sử dụng truy vấn SQL sau để hiển thị việc sử dụng toán tử REGEXP:

```sql
SELECT name FROM person_tbl WHERE name REGEXP '^st';

SELECT name FROM person_tbl WHERE name REGEXP 'ok$';

SELECT name FROM person_tbl WHERE name REGEXP 'mar';

SELECT FirstName FROM intque.person_tbl WHERE FirstName REGEXP '^[aeiou].*ok$';
```

LỆNH ALTER :

Lệnh Alter trong SQL dùng để thay đổi cấu trúc cơ sở dữ liệu. Chúng ta có thể thêm hoặc bớt các cột. Chúng ta có thể thay đổi lược đồ của cột bảng trong SQL bằng cách sử dụng lệnh Alterna. Lệnh Alter cũng có thể được sử dụng để đổi tên bảng. Với sự trợ giúp của lệnh thay đổi, chúng ta cũng có thể định vị lại các cột.

- Thêm cột:
  Với sự trợ giúp của truy vấn sau, chúng ta có thể thêm một cột trong bảng SQL:
  Alter table employee add name varchar(30);
  Lệnh này thêm một cột tên kiểu dữ liệu varchar có độ dài 30 ký tự vào bảng nhân viên.
- Xóa cột:
  Với sự trợ giúp của truy vấn sau, chúng ta có thể bỏ một cột từ các bảng SQL:
  Alter table employee drop name;
  Lệnh này xóa cột tên khỏi bảng nhân viên
- Định vị lại vị trí cột:
  Với sự trợ giúp của các truy vấn sau, chúng ta có thể định vị lại một cột cụ thể trong bảng SQL:
  Alter table employee add salary int first;
  Lệnh này thêm cột lương kiểu int vào bảng nhân viên ở vị trí đầu tiên.
  Alter table employee add salary int after name;
  Lệnh này thêm cột lương kiểu int vào bảng nhân viên sau cột tên
- Cài đặt giá trị mặc định:
  Lệnh sau giúp đặt giá trị mặc định cho một cột cụ thể nếu giá trị rõ ràng không được cung cấp.
  Alter table employee alter salary set default 1000;
  Lệnh này đặt giá trị mặc định của cột lương thành 1000 nếu giá trị rõ ràng của cột lương không được cung cấp.
- Thay đổi định nghĩa cột:
  Alter table employee modify name varchar(20);
  Lệnh này thay đổi cột tên trong bảng nhân viên và sửa đổi kiểu dữ liệu của nó thành varchar (20) bằng cách ghi đè kiểu dữ liệu trước đó của nó.

- Đổi tên một cột:
  Lệnh sau thay đổi tên của bảng trong cơ sở dữ liệu SQL.
  Alter table employee rename to emp;
  Lệnh này thay đổi tên của bảng nhân viên và đổi tên nó thành emp.

  Lệnh Alter chỉ cần cập nhật cấu trúc của bảng cơ sở dữ liệu. Nó có thể thêm, bớt, đặt lại vị trí, đổi tên các cột hoặc đơn giản là đổi tên toàn bộ bảng của cơ sở dữ liệu.

CÁC CHỈ SỐ TRONG SQL

Chỉ mục SQL là cấu trúc dữ liệu được sử dụng để cải thiện tốc độ hoạt động trong bảng. Chỉ mục có thể được tạo trên một hoặc nhiều cột, do đó cung cấp cơ sở cho việc tra cứu bảng nhanh chóng và hiệu quả.
Chỉ mục cũng là một loại bảng lưu giữ bản ghi các con trỏ trỏ đến mỗi bản ghi trong bảng thực tế.
Chỉ mục có thể có hai loại:

- ✓ Simple Index
- ✓ Unique Index

Lập chỉ mục đơn giản cho phép 2 hàng có cùng giá trị chỉ mục trong khi Unique không cho phép 2 hàng của bảng không có cùng giá trị chỉ mục.
Lệnh sau giúp tạo chỉ mục trong bảng cơ sở dữ liệu.

Simple Index :

```
1 | Create index my_index on employee( emp_name, salary);
```

Unique Index :

```
1 | Create unique index my_index on employee(emp_name, salary);
```

Các truy vấn trên tạo một chỉ mục my_index trên bảng nhân viên với các cột tên_và và lương;

Chúng ta có thể sử dụng lệnh thay đổi để thêm hoặc xóa các chỉ mục trên một bảng hiện có.

- Alter table employee add primary key (emp_name, salary)
  Lệnh này thêm chỉ mục khóa chính trên bảng nhân viên

- Alter table employee add unique my_index (emp_name, salary)
  Lệnh này thêm chỉ mục duy nhất my_index trên bảng nhân viên trong đó 2 hàng không được có cùng giá trị chỉ mục (lập chỉ mục duy nhất)
- Alter table employee add index my_ind (emp_name, salary)
  Lệnh này thêm một chỉ mục bình thường my_ind trên bảng nhân viên trong đó 2 hàng có thể có cùng giá trị chỉ mục (lập chỉ mục đơn giản)
- Alter table employee drop index my_index
  Lệnh này loại bỏ chỉ mục my_index khỏi bảng nhân viên.
  Hiển thị các chỉ mục:
  Lệnh sau có thể được sử dụng để hiển thị tất cả các chỉ mục trong bảng SQL.
  Hiển thị chỉ mục từ nhân viên \ G
  Ở đây \ G được sử dụng để hiển thị các kết quả theo thứ tự dọc, do đó tránh các kết quả dài theo chiều ngang.

GIAO DỊCH TRONG CƠ SỞ DỮ LIỆU

Trong cơ sở dữ liệu, một giao dịch là một nhóm tuần tự các câu lệnh, được thực hiện như thể nó là một đơn vị duy nhất.

Cơ sở dữ liệu phải tuân theo các thuộc tính giao dịch sau: **ACID**

1. (A)tomicity: Nó đảm bảo rằng các giao dịch được hoàn thành đầy đủ hoặc hoàn toàn bị hủy bỏ. Không nên có giao dịch từng phần diễn ra trong SQL. Ví dụ. Nếu chuyển tiền được thực hiện từ tài khoản A sang tài khoản B, thì quá trình chuyển tiền hoàn tất sẽ được thực hiện hoặc không được thực hiện. Không được có giao dịch từng phần diễn ra ghi nợ tiền từ tài khoản A và không ghi có tiền vào tài khoản B.
2. (C)onsistency(tính nhất quán): Cơ sở dữ liệu phải luôn duy trì ở trạng thái nhất quán. Cơ sở dữ liệu phải nhất quán sau truy vấn SQL vì nó nhất quán trước truy vấn SQL. Ví dụ. Khi chuyển tiền diễn ra từ tài khoản A sang tài khoản B, thì tổng số tiền trong tài khoản A và tài khoản B phải luôn bằng nhau trước và sau khi giao dịch.
3. (I)solation: Nó đảm bảo rằng nếu nhiều giao dịch đang chạy thành công trong cơ sở dữ liệu, thì mỗi giao dịch sẽ không bị ảnh hưởng bởi các giao dịch khác đang chạy trong cơ sở dữ liệu. Ví dụ. nếu có 2 thao tác ghi được thực hiện trong một cơ sở dữ liệu, thì sẽ không có sự can thiệp nào giữa hai thao tác đang được thực hiện. Chúng nên được thực hiện một cách cô lập.
4. (D)urability: Cơ sở dữ liệu phải chống được lỗi. Các thay đổi được thực hiện trong hệ thống cơ sở dữ liệu phải được lưu trữ vĩnh viễn ngay cả khi hệ thống bị lỗi phần mềm. Ví dụ. Nếu tài khoản A đã được ghi có với số tiền Rs. 50000, thì thay đổi đó sẽ tồn tại vĩnh viễn trong tài khoản ngay cả khi hệ thống hoặc phần mềm bị lỗi vì nó chứa thông tin nhạy cảm liên quan đến tài khoản của khách hàng.

2.6. So sánh SQL với MySQL

Mô hình quan hệ lần đầu tiên được mô tả trong một bài báo năm 1970 bởi Edgar F. Codd. Một trong những ngôn ngữ lập trình thương mại đầu tiên liên quan đến mô hình, SQL, được phát triển ngay sau đó tại IBM. Trong một thời gian, SQL là ngôn ngữ cơ sở dữ liệu được sử dụng rộng rãi nhất, được chấp nhận như một tiêu chuẩn ANSI vào năm 1986 và trong ISO một năm sau đó.

SQL bao gồm bốn ngôn ngữ con, mỗi ngôn ngữ có một phạm vi khác nhau.

- DQL: Ngôn ngữ truy vấn dữ liệu (DQL) là ngôn ngữ quen thuộc nhất và được sử dụng để chạy các truy vấn trên cơ sở dữ liệu và trích xuất thông tin từ dữ liệu được lưu trữ. Ví dụ: chọn và trả về giá trị lớn nhất trong một cột.
- DDL: Một ngôn ngữ định nghĩa dữ liệu (DDL) được sử dụng để mã hóa các cấu trúc và lược đồ cụ thể của cơ sở dữ liệu. Tạo bảng hoặc xác định kiểu dữ liệu là một ví dụ.
- DCL: Một ngôn ngữ kiểm soát dữ liệu (DCL) xác định quyền truy cập, ủy quyền và quyền cho người dùng và các quá trình truy cập cơ sở dữ liệu, bao gồm cả việc cấp đặc quyền quản trị viên hoặc hạn chế người dùng chỉ có các đặc quyền chỉ đọc.
- DML: Và cuối cùng, ngôn ngữ thao tác dữ liệu (DML) được sử dụng để thực hiện sửa đổi trên các thành phần hiện có của cơ sở dữ liệu, như chèn bản ghi, cập nhật giá trị trong ô hoặc xóa dữ liệu.

Công ty Thụy Điển MySQL AB lần đầu tiên phát hành MySQL vào năm 1995. Giống như hầu hết các phần mềm cơ sở dữ liệu theo sau sự nổi lên ban đầu của các hệ thống quan hệ, MySQL chỉ đơn giản là một phần mở rộng của tiêu chuẩn SQL ban đầu , bổ sung thêm nhiều tính năng, hỗ trợ, lập trình thủ tục, cơ chế luồng điều khiển, và nhiều hơn nữa.

2.7. Đám mây và tương lai của mySQL

MySQL ban đầu được hình dung để quản lý cơ sở dữ liệu khổng lồ, nhanh hơn các phần mềm cơ sở dữ liệu hiện có. Được sử dụng trong các môi trường hoạt động, giao dịch và sản xuất đòi hỏi khắt khe trong nhiều thập kỷ, MySQL đã phát triển cùng với việc chuyển tính toán và lưu trữ vào đám mây .

Mặc dù thường được cài đặt trên các máy riêng lẻ, MySQL hiện bao gồm hỗ trợ sâu cho các ứng dụng phân tán và đưa vào hầu hết các nền tảng dữ liệu đám mây.

So với nhiều giải pháp lưu trữ và xử lý dữ liệu trên thị trường hiện nay, MySQL là một công nghệ cũ hơn, nhưng nó không có dấu hiệu gắn cờ về mức độ phổ biến hay tiện ích. Trên thực tế, MySQL đã có một sự hồi sinh gần đây so với các hệ thống lưu trữ hiện đại chuyên biệt hơn, do tốc độ, độ tin cậy, dễ sử dụng và khả năng tương thích rộng của nó.

3. Đánh giá thành viên

3.1. Thời gian thực hiện của nhóm

Thời gian / Công đoạn	31/03/2022	04/04/2022	05/04/2022	06/04/2022	07/04/2022	Thành viên tham gia
Thu thập thông tin						Tất cả
Hoàn thiện word và ppt						Thật, Khôi
Demo các thao tác						Tất cả

3.2. Đánh giá mức độ đóng góp cho đồ án

Thành viên	Mức độ đóng góp cho đồ án (%)
19200349 - Nguyễn Đình Khôi	33
19200495 - Huỳnh Chí Thật	33
19200421 - Nguyễn Hồng Phát	33