**Bộ câu hỏi phỏng vấn SQL**

# 1.(Trending)Em có biết về NoSQL, sự khác nhau giữa NoSQL và SQL?

\* **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, dễ mở rộng .

- Thuật ngữ **NoSQL** được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các **lightweight open source relational database** (cơ sở dữ liệu quan hệ nguồn mở nhỏ) nhưng không sử dụng **SQL** cho truy vấn.

- Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ **NoSQL** trong một hội thảo về cơ sở dữ liệu nguồn mở phân tán.

- Thuật ngữ **NoSQL** đánh dấu bước phát triển của thế hệ database mới: **distributed** (phân tán) + **non-relational** (không ràng buộc). Đây là 2 đặc tính quan trọng nhất.

- 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 nhưng không sử dụng **SQL** cho truy vấn.

- 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.

- Đối với hệ thống cơ sở dữ liệu **NoSQL**, nó bao gồm một loạt các công nghệ cơ sở dữ liệu có thể lưu trữ dữ liệu có cấu trúc, bán cấu trúc, phi cấu trúc và đa hình.

**\* Sự khác nhau giữa NoSQL và SQL**

**a.Mô hình dữ liệu:**

- **SQL Databases:** Mô hình quan hệ chuẩn hóa dữ liệu vào bảng được hình thành từ hàng và cột. Sơ đồ quy định rõ ràng bảng, hàng, cột, chỉ mục, mối quan hệ giữa các bảng và các thành tố cơ sở dữ liệu khác. Cơ sở dữ liệu sẽ thực thi tính toàn vẹn tham chiếu trong mối quan hệ giữa các bảng.

- **NoSQL Databases:** Kiểu Document: JSON documents; Kiểu Key-value: key-value pairs; Kiểu column: bảng với hàng và cột thay đổi (dynamic); Graph: nodes và cạnh

**b.Lịch sử phát triển:**

**- SQL Databases:** Phát triển từ những năm 1970, tập trung vào việc giảm thiểu sự trùng lắp dữ liệu.

- **NoSQL Databases:** Phát triển sau năm 2000, mục đích chính phục vụ việc mở rộng nhanh (scale) và phát triển các ứng dụng theo phương pháp Agile và DevOps.

**c.Các CSDL tiêu biểu:**

**- SQL Databases:** Oracle, MySQL, Microsoft SQL Server,PostgreSQL

**- NoSQL Databases:** NoSQL Document: MongoDB, CouchDB,Key-value: Redis, DynamoDB, Column: Cassandra, HBase, Graph: Neo4j, Amazon Neptune

**d.Mục đích sử dụng**

**- SQL Databases:** Nhiều mục đích

**- NoSQL Databases:** NoSQL Document DB: nhiều mục đích, Key-value: dữ liệu lớn với các truy xuất đơn giản, Column: dữ liệu lớn với các mẫu truy xuất dữ liệu có thể dự đoán được (predictable query patterns), Graph: phân tích và duyệt các mối quan hệ giữa các dữ liệu được kết nối

**e.Schemas**

**- SQL Databases:** Cố định.

**- NoSQL Databases:** Uyển chuyển

**f.Mở rộng**

**- SQL Databases:** Theo chiều dọc. Cơ sở dữ liệu quan hệ thường tăng quy mô bằng cách tăng năng lực điện toán của phần cứng hoặc tăng quy mô bằng cách thêm bản sao của khối lượng công việc chỉ đọc.

**- NoSQL Databases:** Cho phép thay đổi quy mô theo chiều ngang (scale-out bằng cách phân tán trên nhiều server)

**g.Thuộc tính ACID**

**- SQL Databases:** Cơ sở dữ liệu quan hệ có các thuộc tính mang tính nguyên tố, nhất quán, tách biệt và bền vững (ACID)

**- NoSQL Databases:** tuân theo định lý Brewers CAP (Consistency, Availability, Partition tolerance).

**h.Data to Object Mapping**

**- SQL Databases:** Yêu cầu ORM (object-relational mapping)

**- NoSQL Databases:** Nhiều NoSQL database không yêu cầu ORMs.

**i.Hiệu năng**

**- SQL Databases:** Hiệu năng thường phụ thuộc vào hệ thống con của ổ đĩa. Thông thường, việc tối ưu hóa các truy vấn, chỉ mục và cấu trúc bảng bắt buộc phải được thực hiện để đạt mức hiệu năng tối đa.

**- NoSQL Databases:**Hiệu năng thường được xem là chức năng của kích cỡ cụm phần cứng ngầm, độ trễ mạng và ứng dụng đưa ra lệnh gọi.

**k.API**

**- SQL Databases:** Yêu cầu lưu trữ và truy xuất dữ liệu được truyền đạt bằng cách sử dụng các truy vấn nhất quán với ngôn ngữ truy vấn có cấu trúc (SQL). Các truy vấn này được phân tích và thực thi bởi cơ sở dữ liệu quan hệ.

**- NoSQL Databases:**API trên cơ sở đối tượng cho phép các nhà phát triển ứng dụng dễ dàng lưu trữ và truy xuất cấu trúc dữ liệu trong bộ nhớ. Khóa phân mảnh tìm kiếm các cặp khóa-giá trị, tập hợp cột hoặc văn bản có cấu trúc chưa hoàn chỉnh có chứa đối tượng và thuộc tính của ứng dụng được xếp theo chuỗi.

# 2.Khi transaction thất bại, các table có tiến trình con thành công có giá trị index như thế nào nếu thêm record mới? Sẽ không có record được thêm mới

# 3.Cho 1 table có 1 colunm lưu url đầy đủ, làm sao để lưu 1 url rút gọn trỏ tới được record có url đầy đủ đó? Để lưu url rút gọn dùng: linkshort

# 4. Query: Trong 1 table có tên và điểm, lấy ra record có điểm xếp thứ 5.

select \* from (select s.point from student as s group by s.point order by s.point desc limit 5) as de order by point asc limit 1;

# 5. Cách xử lý các câu lệnh query liên tiếp nhau? Dùng subquery.

# 6. (Trending)**Transaction trong SQL?**

**Transaction** trong SQL là một nhóm các câu lệnh SQL.(là 1 tiến trình nhiều HĐ có điểm đầu và điểm cuối, thành công nếu tất cả HĐ thành công, thất bại nếu 1 HĐ trong tiến trình thất bại)

* Nếu một **transaction** được thực hiện thành công, tất cả các thay đổi dữ liệu được thực hiện trong transaction được lưu vào cơ sở dữ liệu.
* Nếu một **transaction** bị lỗi và được **rollback**, thì tất cả các sửa đổi dữ liệu sẽ bị xóa (dữ liệu được khôi phục về trạng thái trước khi thực hiện **transaction**).

# 7. (Trending) Phân biệt WHERE và HAVING. Khi nào dùng WHERE vs HAVING? Cho ví dụ minh họa?

- **Where**: Mệnh đề điều kiện, dùng để truy xuất dữ liệu theo 1 điều kiện nào đó (một số toán tử theo sau **where**: <, >, < >, =, or, and, not, between, like, in, …)

- **Having**: Mệnh đề điều kiện, thường sử dụng sau **Group by** và thường sử dụng để làm điều kiện cho các hàm thống kê count(), sum(), avg(), max(), min()…, sử dụng để lọc các bản ghi và chỉ lấy những bản ghi phù hợp với yêu cầu hoặc thực sự cần thiết tương tự như mệnh đề **Where**.

Tuy nhiên:

+ **Where** là câu lệnh điều kiện trả kết quả đối chiếu với từng dòng.

+ **Having** có thể thay thế where trong câu lệnh seclect, ngược lại thì không vì:

+ **Having** là câu lệnh điều kiện trả kết quả đối chiếu cho nhóm được tạo bởi mệnh đề **Group by** => Vì vậy mà sau **Group by** thì sẽ chỉ dùng được **Having** còn **Where** không dùng được sau **Group by**

# 8. Trigger, Procedure?

**Trigger:** là một thủ tục [**SQL**](https://topdev.vn/it-jobs/sql-kt28)được thực thi ở phía server khi có một sự kiện như **Insert, Delete, hay Update.**

* **Trigger** là một loại **stored procedure** đặc biệt (không có tham số) được thực thi (**execute**) một cách tự động khi có một sự kiện thay đổi dữ liệu (**data modification**).
* **Trigger** được lưu trữ và quản lý trong **Server DB**, được dùng trong trường hợp ta muốn kiểm tra các ràng buộc toàn vẹn trong **DB**.

**Procedure** (Thủ tục) là một chương trình trong cơ sở dữ liệu gồm nhiều câu lệnh mà bạn lưu lại cho những lần sử dụng sau.

* Trong **SQL Server**,có thể truyền các tham số (**IN,OUT,INOUT**) vào **procedure**, tuy nó không trả về một giá trị cụ thể như **function** (hàm) nhưng cho biết việc thực thi thành công hay thất bại.

# 9. Để map thuộc tính Entity vs cột trong SQL thì dùng annotation gì? @Entity, @Column

# 10.(Trending) So sánh DELETE & TRUNCATE:

**+Giống:** đều là xóa dữ liệu của bảng (khác với DROP là xóa bay luôn cả bảng).

**DELETE**

* Thuộc nhóm câu lệnh DML.
* Dùng để xóa table, xóa record (có thể dùng chung với điều kiện).
* Đối với id có sử dụng AUTO\_INCREMENT, Khi chạy lệnh DELETE thì SQL sẽ log lại từng dòng đã xóa vào transaction log, vì thế nên khi bạn tạo 1 record mới, giá trị của id sẽ không bắt đầu từ 1 mà sẽ có giá trị n+1 với n là giá trị của record cuối cùng được tạo.
* Dữ liệu bị xóa bởi DELETE có thể phục hồi (restored) được.

**TRUNCATE**

* Thuộc nhóm câu lệnh DDL.
* Chỉ có thể dùng cho table chứ không thể dùng cho từng record.
* Khi chạy lệnh TRUNCATE thì SQL sẽ xóa hết dữ liệu của bảng và reset transaction log, vì thế khi tạo 1 record mới, giá trị của id sẽ bắt đầu từ 1, đây cũng chính là khác biệt lớn nhất của DELETE và TRUNCATE.
* Xóa toàn bộ các dòng của bảng, giải phóng bộ nhớ và không thể phục hồi lại.
* TRUNCATE không thể xóa đc bảng có foreign key trỏ đến bảng khác.
* TRUNCATE sẽ luôn nhanh hơn vì nó tiêu tốn ít bộ nhớ hơn.

# 11.Các câu lệnh trong MySQL

**- MySQL –** là hệ quản trị CSDL:

**+ MySql server –** dịch vụ để quản trị CSDL.

**+ MySql workbench –** là chương trình giúp lập trình viên có thể tương tác với CSDL.

**SQL (Structured Query Language) –** ngôn ngữ truy vấn cấu trúc:

**+ DDL (Data Definition Language) –** định nghĩa cấu trúc dữ liệu **(CREATE, ALTER, DROP, …).**

**+ DML (Data Manipulation Language) –** thao tác, truy vấn dữ liệu **(SELECT, INSERT, UPDATE, DELETE, …).**

**+ DCL (Data Control Language) –** điều khiển, phân quyền **(GRANT, REVOKE, …).**

**+ TCL (Transaction Control Language) –** tiến trình.

# 12. (Trending) Phân biệt inner join & left join?

**- Join –** dùng để lấy thông tin trên 2 hay nhiều bảng, kết nối với nhau thông qua các trường chung (bản chất là gộp dữ liệu theo cột). Bao gồm:

**+ Inner join –** lấy phần chung của 2 bảng.

**+ Outer join:**

- Left join – lấy phần chung & phần bên trái.

- Right join – lấy phần chung & phần bên phải.

- Full join – lấy phần chung, phần bên trái & phần bên phải.

- Self join – 1 bảng tự join chính nó thông qua 1 hay nhiều cột.

- **Cross join** - 2 bảng join vs nhau không qua cột chung,bảng mới sinh ra có số record bằng tích số record 2 bảng.

(MySql không hỗ trợ Full join, muốn sử dụng thì phải kết hợp giữa Left join, Right join và Union).

# 13.Khi nào dùng JPQL vs native query?

- Khi cần thao tác đến một database bất kỳ, chúng ta phải viết câu **SQL** dựa trên thông tin của các cột, các bảng trong **database** đó và có thể đối với mỗi loại **database** khác nhau, syntax của câu **SQL** cũng khác nhau. Để loại bỏ những nhược điểm này, các bạn có thể sử dụng **JPA Query Language** (**JPQL**).

- Đôi khi về câu lệnh **SQL** được tạo ra bởi **Hibernate** chậm hoặc cần viết câu lệnh phức tạp và thích tạo câu lệnh **SQL** (**SQL** gốc) của riêng mình hơn thì dùng **native query**.

# 14.Dùng DELETE để xóa dữ liệu thì làm thế nào để rollback lại dữ liệu đã xóa?

**Bước 1:** Bạn tạo một **database** mới, rồi **Import file .sql** cũ vào.

**Bước 2:** Bạn khôi phục lại từng bảng dữ liệu trong database.

- Chạy thực thi lệnh xóa tablespace của bảng cần khôi phục:

**ALTER TABLE** <Tên bảng cần khôi phục> **DISCARD TABLESPACE**

Sau khi chạy lệnh trên trong thư mục **Mysql/data/<database mới>** sẽ mất **file .ibd** của bảng bạn đang khôi phục.

- Bạn sao chép **file .ibd** của bảng trong thư mục của cơ sở dữ liệu bị lỗi vào **Mysql/data/<database mới>.**

- Chạy thực thi lệnh **import** lại **tablespace** của bảng cần khôi phục.

**ALTER TABLE** <Tên bảng cần khôi phục> **IMPORT TABLESPACE**

# 15.(Trending) Viết câu query JOIN 2 bảng có mối quan hệ n-n

Tạo một bảng trung gian **(table\_intermediary)** có khóa ngoại là khóa chính 2 bảng cần **JOIN**

**Query: select \* from table\_intermediary join table\_1 on table\_intermediary.table\_1\_id = table\_1.id**

**join table\_2 on table\_intermediary.table\_2\_id = table\_2.id;**

# 16.So sánh Native query và HQL (Hibernate Query Language)

1. **Native Query:** Truy vấn bảng hướng cơ sở dữ liệu, thao tác dữ liệu được lưu trữ trong các bảng và sửa đổi các hàng và cột của nó. Nhanh hơn.Không hỗ trợ các tính năng OOP như đa hình, kế thừa và liên kết.Chỉ dựa trên RDBMS.
2. **HQL:** Truy vấn hướng đối tượng. Quan tâm đến các đối tượng và thuộc tính của nó. Chậm hơn. Hỗ trợ các tính năng OOP như đa hình, kế thừa và liên kết. Sự kết hợp của OOP với cơ sở dữ liệu quan hệ.

# 17. (Trending) Các mối quan hệ trong SQL? One to One; One to Many; Many to One; Many to Many

# 18.Em hiểu Port là gì? Mysql chạy port bao nhiêu?

* **Port** là giao thức bit 16 đứng đầu các tập tin thuộc hai giao thức [**UDP và TCP**](https://vietnix.vn/udp-va-tcp/). Ngoài ra, port còn được biết đến là cổng quy định đối với các gói dữ liệu được phân định.
* Để thiết bị của bạn có thể nhận và xuất tập tin thì phải cần đăng ký port – thuật toán đã được lập trình trước.
* Bên cạnh đó, port sẽ được quy đổi tương tự như bit của một mã dữ liệu bất kỳ. Hiểu đơn giản thì nó sẽ phân tích và quyết định việc dữ liệu nào được phép ra hoặc vào thiết bị tương tự như một người gác cổng cho toàn hệ thống máy tính. **-My SQL chạy port 3306**

# 19.Sự khác nhau giữa các câu lệnh TRUNCATE, DELETE và DROP?

1. **DELETE:** Xóa một hay tất cả dòng trong một bảng theo một điều kiện nhất định, dữ liệu có thể phục hồi lại. Thêm mới đối với id tự động tăng sẽ đc thêm ở i+1
2. **TRUNCATE:** Xóa toàn bộ các dòng của bảng, giải phóng bộ nhớ và không thể phục hồi lại. Thêm mới đối với id tự động tăng sẽ bắt đầu lại từ 0
3. **DROP:** Xóa một bảng khỏi database, dữ liệu không thể phục hồi lại.

# 20.(Trending) Khóa chính (PK) và khóa phụ (FK) là gì?

- **Khóa chính (hay ràng buộc khóa chính: Primary Key)** là một cột hoặc tập hợp các cột trong một bảng.Được sử dụng để định danh duy nhất mỗi record trong table của cơ sở dữ liệu,duy nhất và không được null.

- Khóa ngoại (**Foreign Key**) là một cột hoặc tập hợp các cột trong một bảng, được xem như con trỏ tham chiếu đến khoá chính của một bảng khác trong cùng cơ sở dữ liệu. Khoá ngoại được sử dụng để xác định mối quan hệ giữa hai bảng. Nó cho phép tạo ra các liên kết giữa các bảng thông qua khoá chính và khoá phụ, giúp đảm bảo tính toàn vẹn dữ liệu và cập nhật dữ liệu một cách hiệu quả.

# 21.Bây giờ có 1 danh sách học sinh gồm 1000 học sinh. Muốn lấy ra 100 record từ 300 đến 400 mà không sử dụng câu lệnh Where theo kiểu Where id > 300 and id < 400. Thì làm như thế nào? =>Sử dụng limit 300,100.

# 22.Phân biệt 1 to many, many to one?

**- One to many (1-n):** bảng phía 1 sẽ có liên kết với nhiều bảng phía n (khóa chính bảng phía 1 được sử dụng để làm khóa ngoại ở các bảng phía n).

**- Many to one (n-1):** bảng phía n chỉ liên kết với 1 bảng phía 1 (khóa ngoại sẽ được đặt ở phía n, tham chiếu đến bảng phía 1).

# 23.Hỏi về GROUP BY, ORDER BY, UNION, UNION ALL?

**- Mệnh đề GROUP BY trong SQL** Server dùng để nhóm các bảng ghi trên 1 hoặc nhiều cột

* Thông thường, **GROUP BY** được sử dụng với các hàm tính toán, thống kê.

**- Mệnh đề ORDER BY** trong **SQL** được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần, dựa trên một hoặc nhiều cột.

* Một số cơ sở dữ liệu sắp xếp các kết quả truy vấn theo thứ tự tăng dần theo mặc định.
* Lệnh **ASC** được sử dụng để sắp xếp tăng dần và **DESC** được sử dụng để sắp xếp giảm dần**.**

**- Toán tử UNION** được sử dụng để kết hợp tập hợp kết quả của hai hoặc nhiều câu lệnh **SELECT.**

**+** Mỗi câu lệnh **SELECT** với **UNION** phải có cùng số lượng cột, các cột phải có cùng kiểu dữ liệu, các cột trong mỗi câu lệnh **SELECT** phải có cùng trật tự.

**- Toán tử UNION ALL** được dùng để kết hợp bộ kết quả từ 2 hoặc nhiều lệnh **SELECT** trong **SQL** Server**.**

* Khác với toán tử **UNION**, toán tử **UNION ALL t**rả về tất cả các hàng từ truy vấn và không xóa các hàng trùng lặp.
* Mỗi lệnh **SELECT** trong toán tử **UNION** phải có cùng số cột trong bộ kết quả với kiểu dữ liệu tương ứng.

# 24.Phân biệt DISTINCT và GROUP BY?

Điểm khác biệt giữa **DISTINCT** và **GROUP BY** là:

* **DISTINCT** được sử dụng để loại bỏ các bản sao của các giá trị trong kết quả truy vấn, trong khi **GROUP BY** được sử dụng để nhóm các bản ghi dựa trên một hoặc nhiều cột.
* **DISTINCT** sẽ trả về một danh sách các giá trị duy nhất từ một cột hoặc nhiều cột trong bảng, trong khi **GROUP BY** sẽ trả về một bảng tạm thời được nhóm lại với các giá trị duy nhất của các cột được chỉ định.
* **DISTINCT** không thể áp dụng các hàm tổng hợp như **SUM, COUNT, AVG.**.., trong khi **GROUP BY** cho phép áp dụng các hàm này cho mỗi nhóm dòng trong bảng tạm thời.

# 25. Câu lệnh đầy đủ của SQL, họ cho một cái bảng bảo lấy theo điều kiện HAVING thì làm thế nào?

SELECT \*, AVG (price) FROM product GROUP BY category HAVING AVG (price) >= 50000

# 26. Unique Key? Ràng buộc trong MySQL không được trùng lặp (duy nhất).

# 27. Cho biết một câu truy vấn đầy đủ?

**SELECT** <Danh sách các cột> **FROM** <Danh sách Bảng> **WHERE** <Các điều kiện> **GROUP** **BY** <Tên cột> **HAVING** <Điều kiện dựa trên **GROUP** **BY**> **ORDER** **BY** <Danh sách cột> **LIMIT**<number>

# 28. Cho bảng Bill (gồm Mã Hoá Đơn, Ngày bán), bảng chi tiết hóa đơn (Mã Hóa Đơn, Mã Sản phẩm, Số lượng): a. Đếm số lượng hóa đơn bán trong ngày hôm qua? b. Tìm số lượng sp có Mã sản phẩm "Laptop" bán được trong ngày hôm qua?

1. **SELECT** **COUNT** (cthd.so\_luong) as so\_luong hoa don ban **FROM** chi\_tiet\_hoa\_don cthd **JOIN** bill b **ON** cthd.ma\_hoa\_don = b.ma\_hoa\_don **GROUP BY** b.ngay\_ban **HAVING** b.ngay\_ban = "Ngày hôm qua"
2. **SELECT COUNT** (cthd.so\_luong) as so\_luong hoa don ban FROM chi\_tiet\_hoa\_don cthd **JOIN** bill b **ON** cthd.ma\_hoa\_don = b.ma\_hoa\_don **GROUP BY** cthd.ma\_san\_pham **HAVING** cthd.ma\_san\_pham = "Laptop" **AND** b.ngay\_ban = "Ngày hôm qua"

# 29. Từ khóa GROUP BY dùng để làm gì? Viết 1 câu SQL có sử dụng GROUP BY

Group by – nhóm dữ liệu trên 1 cột, sắp xếp các hàng của truy vấn theo nhóm (thường dùng với các hàm count(), sum(), avg(), max(), min(),…)

SELECT \*, AVG (price) FROM product GROUP BY category HAVING AVG (price) >= 50000;

# 30. Phân biệt JOIN và LEFT JOIN? ở trên có rồi.

# 31. Vì sao GROUP BY sử dụng chung với HAVING mà không dùng với WHERE?

1. Mệnh đề WHERE dùng để kiểm tra các điều kiện cho các thuộc tính của bảng, trả kết quả đối chiếu với từng dòng
2. Mệnh đề HAVING được dùng để giới hạn nhóm các hàng trả về trong bảng, trả kết quả đối chiếu cho các nhóm (Sum, Count, Average, …) được tạo bởi Group by.
3. Mệnh đề HAVING được thêm vào SQL vì mệnh đề WHERE không thể sử dụng được với các aggregate functions.

# 32. Trigger là gì? Dùng để làm gì?

1. Trigger là một thủ tục SQL được thực thi ở phía server khi có một sự kiện như Insert, Delete, hay Update.
2. Trigger là một loại stored procedure đặc biệt (không có tham số) được thực thi (execute) một cách tự động khi có một sự kiện thay đổi dữ liệu (data modification).
3. Trigger được lưu trữ và quản lý trong Server DB, được dùng trong trường hợp ta muốn kiểm tra các ràng buộc toàn vẹn trong DB

# 33. Transaction trong db và mối liên hệ với transaction trong java, triển khai transaction trong java như thế nào?

1. Mối liên hệ giữa transaction trong cơ sở dữ liệu và trong Java là khi thực hiện các hoạt động dữ liệu trong Java, ta có thể muốn đảm bảo rằng các thay đổi được thực hiện trong một giao dịch đều được thực hiện hoặc không được thực hiện.
2. Do đó, ta sử dụng các API quản lý transaction để triển khai các giao dịch trong Java.
3. Để triển khai transaction trong Java:
   * Xác định transaction scope: Xác định những hoạt động nào sẽ được thực hiện trong transaction, như thêm, sửa, xóa hoặc truy vấn dữ liệu.
   * Bắt đầu transaction: Khởi tạo transaction bằng cách sử dụng API quản lý transaction, ví dụ như JTA hoặc JDBC.
4. Thực hiện các hoạt động trong transaction: Thực hiện các hoạt động dữ liệu đã xác định trong transaction.
5. Kết thúc transaction: Nếu tất cả các hoạt động dữ liệu được thực hiện thành công, ta có thể commit transaction.
6. Nếu có lỗi, ta có thể rollback transaction để quay trở lại trạng thái trước khi bắt đầu transaction.
7. Transaction trong spring boot: @Transactional.

# 34. Tổ chức database, các mối quan hệ.?

Database nhóm dữ liệu có tổ chức dưới dạng bảng. Các mối quan hệ: 1-1, 1-N, N-N.

# 35.(Trending) Database view?

Bảng ảo là kết quả các truy vấn (**select**) đươc lưu lại dưới dạng bảng và sau đó có thể truy vấn dữ liệu như là bảng thật. **Mục đích:** - Tái sử dụng mã nguồn. - Bảo mật thông tin cho bảng chính.

# 36. SQL phỏng vấn nhiều, quan hệ và truy vấn?

- **Quan hệ (relationship)** thường được sử dụng để mô tả mối liên hệ giữa các đối tượng trong hệ thống cơ sở dữ liệu.

* Một quan hệ bao gồm các bảng (table) và các thuộc tính (attribute) của chúng.
* Các bảng được sử dụng để lưu trữ thông tin về các đối tượng, trong khi các thuộc tính mô tả các đặc tính của các đối tượng đó.

- **Truy vấn (query)** là quá trình truy xuất dữ liệu từ cơ sở dữ liệu để tìm kiếm thông tin cụ thể.

* Truy vấn có thể được thực hiện bằng cách sử dụng ngôn ngữ truy vấn cơ sở dữ liệu (SQL) hoặc các công cụ quản lý cơ sở dữ liệu khác như MongoDB, Elasticsearch, Cassandra, và Redis.
* Trong truy vấn, người dùng có thể chỉ định các tiêu chí để tìm kiếm thông tin. Các tiêu chí này có thể là các điều kiện, phép toán logic, hoặc các thuật toán xử lý dữ liệu. Kết quả của truy vấn là tập hợp các dữ liệu phù hợp với các tiêu chí đã chỉ định.

# 37. SQL rollback?

Rollback () – quay lại trạng thái ban đầu, khôi phục lại toàn bộ các thay đổi dữ liệu trong transaction đã thực hiện.

# 38. Làm sao select nhiều table? Join bảng Union UnionAll

# 39. Hiểu thế nào về SELECT lồng nhau?

SELECT lồng nhau sub-query là một kỹ thuật truy vấn trong SQL cho phép thực hiện truy vấn SELECT bên trong một truy vấn SELECT khác. Kết quả của truy vấn bên trong sẽ được sử dụng như một bảng ảo để thực hiện truy vấn bên ngoài.

# 40. Query ra sản phẩm có giá lớn nhất.

SELECT \* FROM products ORDER BY price DESC LIMIT 1

# 41. Query sp đồng giá có bao nhiêu sp?

SELECT COUNT(\*) FROM products WHERE price = 'đơn giá sản phẩm cần tìm'

# 42. Index là gì? Có mấy loại index? Có thể group by index được không?

**1/Index (chỉ mục)** – là 1 bảng tra cứu đặc biệt có thể sự dụng để tăng hiệu quả tìm kiếm dữ liệu

**Cách đánh index** - Đảm bảo rằng tất cả các cột khóa chính đều được lập index.

- Không nên sử dụng trong các bảng nhỏ, ít bản ghi.

- Đánh index vs các trường thường xuyên tìm kiếm vd ngày.

- Không nên sử dụng Index trong bảng mà các hoạt động UPDATE, INSERT xảy ra thường xuyên với tần suất lớn.

- Không nên sử dụng cho các cột mà chứa một số lượng lớn giá trị NULL

**Mục đích của index**: + Sử dụng để tăng nhanh thời gian và hiệu suất truy xuất dữ liệu

**2/Trong SQL, có 4 loại chính của chỉ mục (index):**

**Primary Key Index:** Là chỉ mục đặc biệt được tạo để duy trì tính nguyên vẹn của khóa chính (Primary Key) trong bảng.

+ Nó đảm bảo rằng không có hai bản ghi nào trong bảng có giá trị khóa chính trùng nhau

+ Primary Key Index thường được tạo tự động khi tạo khóa chính cho bảng.

**Unique Index:** Là chỉ mục được tạo để đảm bảo rằng không có hai bản ghi trong bảng có cùng giá trị của cột chỉ mục.

+ Unique Index cho phép tìm kiếm nhanh các bản ghi trong bảng dựa trên giá trị của cột chỉ mục.

+ Unique Index có thể được tạo cho một hoặc nhiều cột trong bảng.

**Clustered Index:** Là chỉ mục được tạo để sắp xếp và lưu trữ các bản ghi trong bảng dựa trên giá trị của cột chỉ mục.

+ Các bản ghi trong bảng được lưu trữ trên đĩa dựa trên thứ tự của cột chỉ mục.

+ Do đó, Clustered Index cho phép truy xuất nhanh các bản ghi trong bảng bằng cách sử dụng các phương pháp tìm kiếm nhị phân.

**Non-clustered Index:** Là chỉ mục được tạo để đánh chỉ mục các bản ghi trong bảng dựa trên giá trị của một hoặc nhiều cột.

+ Tương tự như Unique Index, Non-clustered Index cho phép tìm kiếm nhanh các bản ghi trong bảng dựa trên giá trị của các cột chỉ mục.

+ Non-clustered Index không sắp xếp và lưu trữ các bản ghi trên đĩa dựa trên thứ tự của cột chỉ mục, mà chỉ đánh chỉ mục các bản ghi và lưu trữ chỉ mục này trên đĩa.

**3/** **Không thể sử dụng GROUP BY trên một chỉ số (index) cụ thể trong bảng**. + Để nhóm các bản ghi trong bảng dựa trên chỉ số của chúng, ta có thể sử dụng các hàm như ROW\_NUMBER(), RANK(), DENSE\_RANK() để xác định chỉ số và sau đó sử dụng GROUP BY trên kết quả của các hàm này.

# 43. Trigger có dùng vs Truncated được không?

Khi chúng ta sử dụng câu lệnh TRUNCATE TABLE, các Trigger không được tự động kích hoạt. Điều này có nghĩa là Trigger sẽ không được thực thi và không thể chạy các hành động được định nghĩa bởi Trigger.

# 44. Nếu trong database không dùng các mối quan hệ 1-n, 1-1, n-n thì làm cách nào để đảm bảo được tính đúng đắn của DL?

- Sử dụng các ràng buộc kiểm soát dữ liệu: Điều này đảm bảo rằng các giá trị được nhập vào bảng phải tuân theo một số quy tắc cụ thể, ví dụ như không được phép nhập giá trị rỗng hoặc giá trị trùng lặp.

+ Các ràng buộc kiểm soát dữ liệu có thể được thiết lập bằng cách sử dụng các khóa chính, khóa duy nhất hoặc các ràng buộc CHECK.

- Sử dụng các quy tắc kinh nghiệm: Các quy tắc kinh nghiệm là các quy tắc được áp dụng dựa trên kinh nghiệm của người thiết kế cơ sở dữ liệu và các quy tắc doanh nghiệp.

+ Các quy tắc này được thiết kế để đảm bảo tính đúng đắn của dữ liệu.

- Sử dụng các quy trình kiểm tra và báo cáo: Các quy trình kiểm tra và báo cáo được thiết kế để kiểm tra tính đúng đắn của dữ liệu và báo cáo các lỗi hoặc không phù hợp với quy tắc.

- Ngoài ra, việc sử dụng các quy trình kiểm tra, bảo trì và cập nhật thường xuyên cũng là một phương pháp quan trọng để đảm bảo tính đúng đắn của dữ liệu trong cơ sở dữ liệu.

# 45. Transaction là gì ? Quy trình hoạt động ? Định nghĩa \* commit.

1/**Transaction** là một tập hợp các hoạt động có điểm đầu và điểm cuối được thực hiện trên cơ sở dữ liệu. Một transaction thành công khi tât cả hoạt động thành công, thất bại khi 1 trong các hoạt động thất bại.

2/Quy trình hoạt động của một transaction sẽ bao gồm các bước sau:

**+ Bắt đầu giao dịch (Begin Transaction):** Bắt đầu transaction bằng cách sử dụng lệnh BEGIN TRANSACTION.

+ **Thực hiện các thao tác trên cơ sở dữ liệu.**Các thao tác như SELECT, INSERT, UPDATE, DELETE được thực hiện trên cơ sở dữ liệu.

+ **Xác nhận (Commit):** Nếu tất cả các thao tác đều thành công, xác nhận transaction bằng cách sử dụng lệnh COMMIT để lưu dữ liệu trên cơ sở dữ liệu.

+ **Hoàn tác (Rollback):** Nếu một trong các thao tác thất bại, transaction sẽ bị huỷ và dữ liệu được hoàn tác bằng cách sử dụng lệnh ROLLBACK.

Transaction đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu trong cơ sở dữ liệu. Nó đảm bảo rằng dữ liệu được lưu trữ trong cơ sở dữ liệu luôn đúng và đáng tin cậy, ngay cả khi xảy ra các lỗi trong quá trình thực hiện các thao tác trên cơ sở dữ liệu.

3/Trong SQL, **COMMIT** là một câu lệnh dùng để lưu các thay đổi được thực hiện trên cơ sở dữ liệu. Khi một giao dịch được bắt đầu, các thay đổi được thực hiện trên cơ sở dữ liệu chỉ được lưu tạm thời và chưa được áp dụng cho cơ sở dữ liệu chính thức.

Việc áp dụng các thay đổi này cho cơ sở dữ liệu chính thức chỉ xảy ra khi sử dụng câu lệnh **COMMIT**.

Câu lệnh **COMMIT** trong SQL có tác dụng áp dụng tất cả các thay đổi được thực hiện trong giao dịch đang chạy vào cơ sở dữ liệu.

Nếu không có câu lệnh **COMMIT**, các thay đổi này sẽ không được lưu vào cơ sở dữ liệu và sẽ bị mất khi giao dịch kết thúc.

Cú pháp của câu lệnh COMMIT như sau: COMMIT; Khi câu lệnh COMMIT được thực thi, tất cả các thay đổi trong giao dịch sẽ được áp dụng cho cơ sở dữ liệu và giao dịch sẽ kết thúc.

# 46.(Trending) Cách đánh INDEX. Mục đích của INDEX?

**1/ Index (chỉ mục)** – là 1 bảng tra cứu đặc biệt có thể sự dụng để tăng hiệu quả tìm kiếm dữ liệu

**2/ Cách đánh index:**

* 1. Đảm bảo rằng tất cả các cột khóa chính đều được lập index.
  2. Không nên sử dụng trong các bảng nhỏ, ít bản ghi. - Hầu hết các cột ngày là những ứng cử viên tốt, cùng với các cột chuỗi ngắn (3 đến 50 ký tự).
  3. Không nên sử dụng Index trong bảng mà các hoạt động UPDATE, INSERT xảy ra thường xuyên với tần suất lớn.
  4. Không nên sử dụng cho các cột mà chứa một số lượng lớn giá trị NULL

**3/ Mục đích của index:** + sử dụng để tăng nhanh thời gian và hiệu suất truy xuất dữ liệu

# 47. Truy vấn SQL (COUNT - GROUP BY). Mối quan hệ n-n trong SQL. Ví dụ thực tế dựa trên project đã làm trong CV.

Đém số lượng bài đăng theo trạng thái: select \*,count(\*) as quanlity from post group by status\_post

# 48. Ví dụ transaction, nếu 1 câu lệnh trong transaction fail, thì câu lệnh kia làm sao để có thể hoạt động.

- Ta có thể sử dụng cơ chế "retry" (thử lại) để thực hiện lại các câu lệnh trong transaction.

- Khi một câu lệnh trong transaction thất bại, ta có thể thực hiện lại câu lệnh đó và tiếp tục thực hiện các câu lệnh tiếp theo trong transaction.

- Sử dụng cấu trúc BEGIN...EXCEPTION...END để bắt lỗi và xử lý khi có lỗi xảy ra trong transaction.

- Nếu câu lệnh thất bại, ta sẽ rollback transaction và thực hiện lại câu lệnh với giá trị khác.

- Sau đó, ta tiếp tục thực hiện các câu lệnh tiếp theo trong transaction và commit transaction khi hoàn thành thành công.

# 49. Tại sao sửa dụng index lại tăng tốc độ truy vấn? Cơ chế hoạt động như thế nào? (nhờ thuật toán tìm kiếm nhị phân)

**1/** Một Index là một con trỏ chỉ tới từng giá trị xuất hiện trong bảng/cột được đánh chỉ mục.

+ Index trong Database có ý nghĩa tương tự như các mục trong xuất hiện trong mục lục của một cuốn sách.

+ Khi bạn đánh chỉ mục cho bảng. Database Enginer sẽ tìm đến chỗ lưu chỉ mục để có thể so sánh và lấy ra vị trí chính xác của hàng dữ liệu phù hợp với điều kiện truy vấn.

Qua đó việc đánh INDEX sẽ giúp việc truy vấn dữ liệu trở lên nhanh hơn và tốn ít tài nguyên hơn so với việc quét toàn bộ bản ghi trong bảng.

**2/ Cơ chế hoạt động:**

Bước 1: kiểm tra phần cần tìm với phần tử giữa mảng.

Bước 2: nếu phần tử cần tìm = = phần tử giữa mảng => trả về index.

Bước 3: nếu phần tử cần tìm < phần tử giữa mảng => xét mảng con bên trái.

Bước 4: nếu phần tử cần tìm > phần tử giữa mảng => xét mảng con bên phải. Bước 5: lặp lại Bước 1 -> Bước 4.

# 50. Query tìm lớp có đông học sinh nhất (khóa ngoại class\_id nằm ở bảng student)

SELECT class\_id, class\_name, COUNT(student\_id) AS num\_students FROM class JOIN student ON class.class\_id = student.class\_id GROUP BY class\_id ORDER BY num\_students DESC LIMIT 1;

# 51. Xây dựng DB theo yêu cầu, mô tả các mối quan hệ giữa các bảng.

Xác định các thực thể (Entity) => Xác định các thuộc tính của Entity => Xác định mối quan hệ giữa các Entity (1-1, 1-n, n-n).

# 52. (Trending) MySQL là gì?

-**MySQL:** là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (open-source) được phát triển bởi tập đoàn Oracle.

MySQL sử dụng ngôn ngữ truy vấn SQL (Structured Query Language) để truy vấn và quản lý dữ liệu.

MySQL là một trong những hệ quản trị cơ sở dữ liệu phổ biến nhất trên thế giới và được sử dụng trong nhiều ứng dụng khác nhau, bao gồm các hệ thống quản lý nội dung, các ứng dụng web và các hệ thống phân tích dữ liệu.

MySQL cung cấp các tính năng như đa người dùng, đa nền tảng, tối ưu hóa hiệu suất và bảo mật dữ liệu. + **MySql server** – dịch vụ để quản trị CSDL. +

**MySql work bench** – là chương trình giúp lập trình viên có thể tương tác với CSDL.

# 53. Query các username giống nhau?

SELECT username, COUNT(\*) as count FROM table\_name GROUP BY username HAVING count > 1;

# 54. Có nên xóa 1 bản ghi trong database không?

Không nên xóa hẳn 1 bản ghi trong database, vì dữ liệu là rất quan trọng và đáng giá, mà chỉ nên xóa mềm, để sau này có thể dùng đến bản ghi đó.

# 55. Câu lệnh SQL Group By dùng để làm gì? Khi nào thì sử dụng Having?

- Group by: nhóm dữ liệu trên 1 cột, sắp xếp các hàng của truy vấn theo nhóm (thường dùng với các hàm count(), sum(), avg(), max(), min(),…).

- Having: sử dụng sau Group by và thường sử dụng để làm điều kiện cho các hàm thống kê, sử dụng để lọc các bản ghi và chỉ lấy những bản ghi phù hợp với yêu cầu.

# 56. Cho câu lệnh SQL và sắp xếp thứ tự chạy?

Trong một truy vấn SQL đơn giản, các lệnh thường được thực hiện theo thứ tự sau:

- SELECT: Lệnh SELECT được thực hiện đầu tiên trong truy vấn để lựa chọn các cột dữ liệu cần truy vấn.

- FROM: Lệnh FROM được thực hiện sau lệnh SELECT để chỉ định bảng hoặc các bảng cần truy vấn dữ liệu.

- WHERE: Lệnh WHERE được thực hiện sau lệnh FROM để lọc các bản ghi dữ liệu phù hợp với các điều kiện cho trước.

- GROUP BY: Lệnh GROUP BY được thực hiện sau lệnh WHERE để nhóm các bản ghi dữ liệu theo một hoặc nhiều cột.

- HAVING: Lệnh HAVING được thực hiện sau lệnh GROUP BY để lọc các bản ghi dữ liệu nhóm phù hợp với các điều kiện cho trước. - ORDER BY: Lệnh ORDER BY được thực hiện cuối cùng trong truy vấn để sắp xếp các bản ghi dữ liệu theo một hoặc nhiều cột.

- LIMIT: giới hạn số lượng record hiển thị. - VD: SELECT \* FROM staft WHERE age> 30 ORDER BY age DESC LIMIT 5;

# 57. SQL Injection là gì? cho ví dụ?

* SQL Injection là một kỹ thuật tấn công web phổ biến, trong đó kẻ tấn công lợi dụng lỗ hổng trong ứng dụng web để thực thi các truy vấn SQL không được cho phép hoặc không được dự đoán.
* Khi thành công, kẻ tấn công có thể truy xuất, thay đổi hoặc xóa dữ liệu trong cơ sở dữ liệu của ứng dụng web.VD:
* Giả sử ứng dụng web của bạn chấp nhận các thông tin đăng nhập từ người dùng và thực hiện một truy vấn SQL để xác minh thông tin đăng nhập. Truy vấn có thể có dạng như sau:

SELECT \* FROM users WHERE username='[username]' AND password='[password]'

* Kẻ tấn công có thể sử dụng SQL Injection để thay đổi truy vấn này để có thể đăng nhập bằng tài khoản của người dùng khác hoặc thậm chí xóa tất cả các bảng trong cơ sở dữ liệu.
* Nếu kẻ tấn công nhập "admin' --" vào ô nhập tên đăng nhập, truy vấn SQL sẽ trở thành:

SELECT \* FROM users WHERE username='admin' --' AND password='[password]'

* Trong trường hợp này, dấu "--" là cú pháp SQL để bỏ qua phần còn lại của câu lệnh SQL, cho phép kẻ tấn công đăng nhập vào tài khoản của quản trị viên mà không cần mật khẩu.
* Hoặc: SELECT \* FROM users WHERE username='admin' AND password='' OR 1=1 --'
* Kết quả truy vấn này sẽ trả về tất cả các bản ghi trong bảng "users" vì điều kiện "1=1" luôn đúng. Như vậy, kẻ tấn công có thể đăng nhập vào tài khoản admin mà không cần biết mật khẩu.

.