DATABASE

▼ SQL ≠ NO SQL

https://viblo.asia/p/nhung-diem-khac-biet-giua-sql-va-nosql-gGJ59b4rKX2

https://viblo.asia/p/sql-vs-nosql-dau-la-lua-chon-phu-hop-cho-du-an-cua-ban-Qbq5QWAzZD8

▼ Transaction, ACID

▼ **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 (hỏng) đồng nghĩa với việc dữ liệu phải rollback (trở lại) trạng thái ban đầu.

- Một transaction đòi hỏi phải có 4 tính chất ACID.
- ACID là viết tắt của cụm từ Atomicity (nguyên tử), Consistency (nhất quán), Isolation (Cô lập), và Durability (Lâu bền).
- Đây là các thuộc tính mà mọi transaction đều được đảm bảo bởi SQL Server.
    - **Atomicity**: Mọi thay đổi về mặt dữ liệu phải được thục hiện trọn vẹn khi transaction thực hiện thành công hoặc không có bất kì sự thay đổi nào về mặt dữ liệu nếu có xẩy ra sự cố. ***Giải thích thêm:***
        - Một giao dịch là đơn vị cơ bản của quá trình xử lý. Tất cả các hoạt động của nó đều được thực thi, hoặc không có hoạt động nào trong số chúng. Giả sử rằng hệ thống bị treo sau thao tác Ghi (A) (nhưng trước khi ghi (B).)
        - Cơ sở dữ liệu phải có khả năng khôi phục các giá trị cũ của A và B (hoặc hoàn thành toàn bộ giao dịch)
    - **Consistency**: Sau khi một transaction kết thúc thì tất cả dữ liệu phải được nhất quán dù thành công hay thất bại. ***Giải thích thêm:***

        - Việc thực hiện một giao dịch một mình phải di chuyển cơ sở dữ liệu từ trạng thái nhất quán này sang trạng thái nhất quán khác.
        - Tổng của A và B phải không thay đổi khi thực hiện giao dịch
    - **Isolation**: Các transaction khi đồng thời thực thi trên hệ thống thì không có bất kì ảnh hưởng gì tời nhau.
        - Một giao dịch không được để các giao dịch khác biết đến ảnh hưởng của nó cho đến khi nó được commit.
        - Nếu hai giao dịch thực hiện đồng thời, có vẻ như một giao dịch đã hoàn thành trước khi giao dịch kia bắt đầu. ***Giải thích thêm:***
        - Nếu một giao dịch khác đang thực hiện đồng thời đang đọc (và / hoặc ghi vào) tài khoản A và B, nó sẽ không thể đọc dữ liệu ở trạng thái không nhất quán (sau khi ghi vào A và trước khi ghi vào B)
    - **Durability**: Sau khi một transaction thành công thì tác dụng mà nó tạo ra phải bền vững trong cơ sở dữ liệu cho dù hệ thống có xẩy ra lỗi. ***Giải thích thêm:***
        - Sau khi giao dịch được cam kết, các thay đổi đối với cơ sở dữ liệu không thể bị mất do lỗi trong tương lai.
        - Khi giao dịch hoàn tất, chúng tôi sẽ luôn có các giá trị mới của A và B trong cơ sở dữ liệu

---

▼ **Transaction** là một loạt các câu lệnh SQL hoạt động giống như một đơn vị duy nhất.

- Nói một cách dễ hiểu, giao dịch là một đơn vị mà một chuỗi công việc được thực hiện để hoàn thành toàn bộ hoạt động.
    - Chúng ta có thể lấy ví dụ về Giao dịch ngân hàng để hiểu điều này.
- Khi chúng ta chuyển tiền từ tài khoản “A” sang tài khoản “B”, một giao dịch sẽ diễn ra.
- Mỗi giao dịch đều có bốn đặc điểm, chúng được gọi là thuộc tính ACID.
    - **Atomicity:** Mọi giao dịch đều tuân theo mô hình tính nguyên tử, có nghĩa là nếu một giao dịch được bắt đầu, thì giao dịch đó phải được

hoàn tất hoặc quay trở lại.

- Ví dụ: nếu một người đang chuyển số tiền từ tài khoản “A” sang tài khoản “B”, thì số tiền đó sẽ được ghi có vào tài khoản B sau khi hoàn tất giao dịch.
- Trong trường hợp nếu có bất kỳ lỗi nào xảy ra, sau khi ghi nợ số tiền từ tài khoản “A”, thay đổi sẽ được hoàn lại.

- **Consistency:** Tính nhất quán nói rằng sau khi hoàn thành một giao dịch, các thay đổi được thực hiện trong quá trình giao dịch phải nhất quán.
  - Ví dụ: nếu tài khoản “A” đã được ghi nợ 200 USD thì sau khi hoàn thành giao dịch, tài khoản “B” sẽ được ghi có 200 USD
  - Nó có nghĩa là các thay đổi phải nhất quán.
- **Isolation:** nói rằng mọi giao dịch nên được cách ly với nhau, không nên có bất kỳ sự can thiệp nào giữa hai giao dịch.
- **Durability:** có nghĩa là một khi giao dịch được hoàn thành, tất cả các thay đổi sẽ là vĩnh viễn, có nghĩa là trong trường hợp hệ thống bị lỗi, các thay đổi sẽ không bị mất

▼ Master & Slave

> 💡 *Hệ thống sẽ có 2 (hoặc nhiều hơn) database giống hệt nhau, mỗi database được cài trên 1 server khác nhau. Trong số các db đó, có 1 db được gọi là master (ông chủ), các db còn lại được gọi là slave (nô lệ). Khi db master xảy ra một sự kiện làm thay đổi dữ liệu (thêm, sửa, xóa) thì db master sẽ log ra một file, các db slave thì liên tục lắng nghe file log này, và thực hiện việc thay đổi dữ liệu trên db slave như thay đổi trên db master.*

- https://viblo.asia/p/gioi-thieu-ve-mysql-replication-master-slave-bxjvZYwNkJZ

▼ Các mô hình database phổ biến

- **Replication:** Hiểu nôm na thì đây là kiến trúc nhân bản. Chúng ta có 1 server Master và 1 hoặc nhiều server Slave. Master dùng chủ yếu để ghi dữ liệu (có

thể dùng để đọc trong trường hợp cần thiết), Slave dùng để dọc dữ liệu và không thể ghi dữ liệu.

- **Partitioning:** đây là phương pháp giúp tối ưu truy vấn khi dữ liệu trong một bảng quá lớn hàng tram triệu bản ghi. Để dễ hình dung thì 1 bảng dữ liệu gióng như một chiếc hộp nhiều ngăn. Mỗi partition là một ngăn, mỗi một ngăn sẽ chưa một số lượng bản ghi theo một quy luật nào đó. Các cách chia thường theo id của bản ghi, hoặc theo thời gian tạo bản ghi theo ngày tháng
- **Cluster:** là mô hình dư liệu phân tán, kết hợp replication + sharding. Dữ liệu đươc lưu ở các datanode, truy vấn qua các sql node, và có 1 node là manager quản lý toàn bộ các sqlnode và datanode
- **Sharding:** là mô hình tương tự như partition, partition chia dữ liệu theo chiều dọc thì sharding chia dữ liệu theo chiều ngang. Mỗi một partition giờ sẽ là 1 server.

https://kipalog.kaopiz.com/posts/Cac-mo-hinh-database-nen-biet

▼ Index

- Link

▼ Store Procedure

- **Stored Procedure** là một tập hợp các câu lệnh SQL dùng để thực thi một nhiệm vụ nhất định. Nó hoạt động giống như một *hàm* trong các ngôn ngữ lập trình khác.
- Stored procedure là một khái niệm khá phổ biến và được hầu hết các hệ quản trị cơ sở dữ liệu (DBMS) hỗ trợ, tuy nhiên không phải tất cả đều hỗ trợ Stored Procedure.
- Lợi ích:
  - Module hóa: bạn chỉ cần viết SP 1 lần, sau đó có thể gọi nhiều lần trong ứng dụng
  - Hiệu suất: SP thực thi mã nhanh hơn và giảm tải băng thông.
  - Bảo mật:

https://viblo.asia/p/gioi-thieu-stored-procedure-trong-sql-server-m68Z0VpM5kG

▼ Trigger

- Hiểu đơn giản thì Trigger là một stored procedure không có tham số. Trigger thực thi một cách tự động khi một trong ba câu lệnh Insert, Update, Delete làm thay đổi dữ liệu trên bảng có chứa trigger
- Tác dụng:
  - Trigger thường được sử dụng để kiểm tra ràng buộc (check constraints) trên nhiều quan hệ (nhiều bảng/table) hoặc trên nhiều dòng (nhiều record) của bảng.
  - Ngoài ra việc sử dụng Trigger để chương trình có những hàm chạy ngầm nhằm phục vụ nhưng trường hợp hữu hạn và thường không sử dụng cho mục đích kinh doanh hoặc giao dịch

https://viblo.asia/p/su-dung-trigger-trong-sql-qua-vi-du-co-ban-aWj538APK6m

https://www.ibm.com/docs/en/informix-servers/12.10?topic=triggers-when-use

▼ **function ≠ procedure**

- Cả stored procedure và function đều là các đối tượng cơ sở dữ liệu chứa một tập các câu lệnh SQL để hoàn thành một tác vụ.
- Một stored procedure (thủ tục lưu trữ) có thể sử dụng lại nhiều lần. Vì vậy, nếu bạn có một truy vấn SQL mà bạn có ý định sử dụng nhiều lần thì hãy lưu nó dưới dạng một SP, sau đó chỉ cần gọi nó để nó thực thi truy vấn SQL của bạn. Ngoài ra, bạn củng có thể truyền tham số cho một SP
- Một function (hàm) được biên dịch và thực thi mỗi khi hàm đó được gọi. Hàm phải trả về giá trị...
- Khác nhau:
  - Thủ tục lưu trữ có thể trả về giá trị zero, một hoặc nhiều giá trị. Trong khi hàm phải trả về một giá trị duy nhất (có thể là bảng).
  - Các hàm chỉ có thể có các tham số đầu vào cho nó trong khi thủ tục lưu trữ có thể có các tham số đầu vào hoặc đầu ra.
  - Hàm có thể được gọi từ thủ tục lưu trữ trong khi thủ tục lưu trữ không thể được gọi từ hàm.
  - Một ngoại lệ có thể được xử lý bằng **try-catch** trong thủ tục lưu trữ, đối với hàm thì không thể.

    - Có thể sử dụng **transaction** trong thủ tục lưu trữ, với hàm thì không thể.

https://itworld.forumvi.net/t688-topic

▼ Aggregate

- Aggregation có thể hiểu là sự tập hợp. Các **Aggregation** operation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất.
- Trong SQL, count(*) và GROUP BY là tương đương với Aggregation trong MongoDB.

▼ distinct ≠ group by

- **DISTINCT** trong SQL được sử dụng kết hợp với câu lệnh SELECT để loại bỏ tất cả các bản ghi trùng lặp và chỉ lấy các bản ghi duy nhất.
- **GROUP BY** trong SQL được sử dụng hợp tác với câu lệnh SELECT để sắp xếp dữ liệu giống nhau thành các nhóm. Mệnh đề GROUP BY này tuân theo mệnh đề WHERE trong câu lệnh SELECT và đứng trước mệnh đề ORDER BY.
- Cả hai mệnh đề đều làm giảm số lượng record trả về bằng cách loại bỏ các bản ghi trùng lặp
- Nhưng khi dùng **group by** ta có thể dùng kết hợp thêm một số aggregate như sum, count, ... còn **distinct** thì không

▼ Inner, outer, left, right join

- **JOIN** trong SQL được sử dụng để kết hợp các bản ghi từ hai hay nhiều bảng trong cơ sở dữ liệu. JOIN là một phương tiện để kết hợp các trường từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng.
- **INNER JOIN**: Trả về các bản ghi có những giá trị phù hợp trong cả hai bảng.
- **LEFT (OUTER) JOIN**: Trả về tất cả bản ghi từ bảng bên trái và bản ghi trùng với bảng bên phải.
- **RIGHT (OUTER) JOIN**: Trả về tất cả bản ghi từ bảng bên phải và bản ghi trùng với bản bên trái.

- **FULL (OUTER) JOI**N: Trả về tất cả bản ghi khi có một kết quả phù hợp trong bảng bên trái hoặc bên phải.

▼ **where ≠ having clause**

- WHERE - filter kết quả theo dòng
- HAVING - filter kết quả theo GROUP

- Where : Là câu lệnh điều kiện trả kết quả đối chiếu với từng dòng
- Having : Là câu lệnh điều kiện trả kết quả đối chiếu cho nhóm (Sum,AVG,COUNT,…)

⇒ Vì vậy mà sau GROUP BY thì sẽ chỉ dùng được Having còn Where thì KHÔNG dùng được sau GROUP BY

(HAVING có thể thay thế vị trí dùng cho WHERE nhưng ngược lại WHERE thì KHÔNG thể thay thế vị trí cho HAVING)

https://daynhauhoc.com/t/su-khac-nhau-giua-where-va-having/21676/2

▼ **delete ≠ truncate ≠ drop**

- DELETE : Xóa một hay tất cả dòng trong một bảng theo một điều kiện nhất định, dữ liệu có thể phục hồi lại
- TRUNCATE : Xóa toàn bộ các dòng của bảng, giải phóng bộ nhớ và không thể phục hồi lại
- DROP : Xóa một bảng khỏi database
- Truncate và drop không thể dùng mệnh đề Where

▼ **Subqueries?**

- Subquery hay còn gọi là truy vấn con, đây là cách viết một câu lệnh SQL mà trong đó có lồng thêm một hoặc nhiều câu truy vấn khác, và thường được sử dụng trong bốn lệnh: SELECT, INSERT, UPDATE hoặc DELETE. Cùng với đó là các toán tử ví dụ như =, <, >, >=, <=, IN, BETWEEN…
- Có một vài quy tắc mà Sub query phải tuân theo:
  - Sub query phải được đặt trong dấu ngoặc đơn.

    - Một sub query có thể chỉ có một cột trong mệnh đề SELECT, trừ khi nhiều cột trong truy vấn chính cho sub query để so sánh các cột đã chọn của nó.
    - Không thể sử dụng lệnh ORDER BY trong sub query, mặc dù truy vấn chính có thể sử dụng ORDER BY. Lệnh GROUP BY có thể được sử dụng để thực hiện chức năng giống như ORDER BY trong một sub query.
    - Sub query trả về nhiều hơn một hàng chỉ có thể được sử dụng với toán tử nhiều giá trị như toán tử IN.
    - Danh sách SELECT không được bao gồm bất kỳ tham chiếu nào đến các giá trị đánh giá BLOB, ARRAY, CLOB hoặc NCLOB.
    - Một sub query không thể được chứa trực tiếp một chức năng set.
    - Toán tử BETWEEN không thể được sử dụng với một sub query. Tuy nhiên, toán tử BETWEEN có thể được sử dụng trong sub query.

▼ Union?

- 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.

- Mệnh đề **UNION** trong SQL được sử dụng để kết hợp các kết quả của hai hoặc nhiều câu lệnh SELECT mà không cần trả về bất kỳ hàng trùng lặp nào.
- Để sử dụng mệnh đề UNION này, mỗi câu lệnh SELECT cần phải có
    - Cùng một số cột được chọn
    - Cùng một số biểu thức cột
    - Cùng kiểu dữ liệu
    - Có chúng trong cùng một trật tự

- **UNION** kết hợp lại nhưng loại bỏ trùng nhau
- **UNION ALL** kết hợp lại nhưng không loại bỏ trùng nhau

▼ Connection pool

- Connection pool (vùng kết nối) : là kỹ thuật cho phép tạo và duy trì 1 tập các kết nối dùng chung nhằm tăng hiệu suất cho các ứng dụng bằng cách sử dụng lại các kết nối khi có yêu cầu thay vì việc tạo kết nối mới.
- Connection Pool Manager (CPM) là trình quản lý vùng kết nối, một khi ứng dụng được chạy thì Connection pool tạo ra một vùng kết nối, trong vùng kết nối đó có các kết nối do chúng ta tạo ra sẵn. Và như vậy, một khi có một request đến thì CPM kiểm tra xem có kết nối nào đang rỗi không? Nếu có nó sẽ dùng kết nối đó còn không thì nó sẽ đợi cho đến khi có kết nối nào đó rỗi hoặc kết nối khác bị timeout. Kết nối sau khi sử dụng sẽ không đóng lại ngay mà sẽ được trả về CPM để dùng lại khi được yêu cầu trong tương lai.

▼ Data warehouse ≠ Data lake

---

https://techtfq.com/blog/top-20-sql-interview-questions?ref=morioh.com&utm_source=morioh.com