



SQL

Luan Phan

- 
- git clone <https://github.com/luanphandinh/intro-database>
 - cd intro-database && docker-compose up



Tổng quan

1. Giới thiệu chung về database
2. Các kiểu dữ liệu cở bản trong MySQL
3. Mô phỏng dữ liệu thực tế
4. Database table (bảng dữ liệu) là gì, cách tạo table
5. Truy vấn (Query) dữ liệu
6. Join table
7. Sub Query
8. Tối ưu câu lệnh
9. Link github phần sql



Giới thiệu chung về Database

1. Database là gì
2. Các kiểu database
3. MySQL



Database là gì

- Database (hệ cơ sở dữ liệu) giúp ta lưu trữ, quản lý và truy xuất thông tin một cách hiệu quả



Các kiểu database phổ biến

- Relational Database: relational database(cơ sở dữ liệu quan hệ) phổ biến nhất, lưu trữ dữ liệu dưới dạng hàng và cột, sử dụng SQL để truy vấn và thay đổi dữ liệu, ví dụ: MySQL, Oracle, SQL Server, and PostgreSQL
 - On disk
- NoSQL Database: non-relational database, không sử dụng SQL, lưu trữ dữ liệu không cấu trúc, ví dụ: MongoDB, Cassandra, and Redis
 - In memory database
- Graph databases: sử dụng graph theory để lưu trữ và truy vấn dữ liệu, thường sử dụng để lưu trữ thông tin có mối quan hệ phức tạp giữa các đơn vị dữ liệu, ví dụ: Neo4j, ArangoDB, and OrientDB
-



Các kiểu dữ liệu cơ bản

1. Numeric (số)
 - a. [INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT](#)
 - b. [DECIMAL, NUMERIC](#)
 - c. [Floating-Point Types \(Approximate Value\) - FLOAT, DOUBLE](#)
2. Date/Time (thời gian)
 - a. [DATE, DATETIME, and TIMESTAMP](#)
3. String (chuỗi)
 - a. [The CHAR and VARCHAR Types](#)
 - b. [The BINARY and VARBINARY Types](#)
 - c. [The BLOB and TEXT Types](#)
 - d. [The ENUM Type](#)
 - e. [The SET Type](#)
4. Spatial Data (không gian)
5. JSON datatype (document)



Mô phỏng dữ liệu trường học

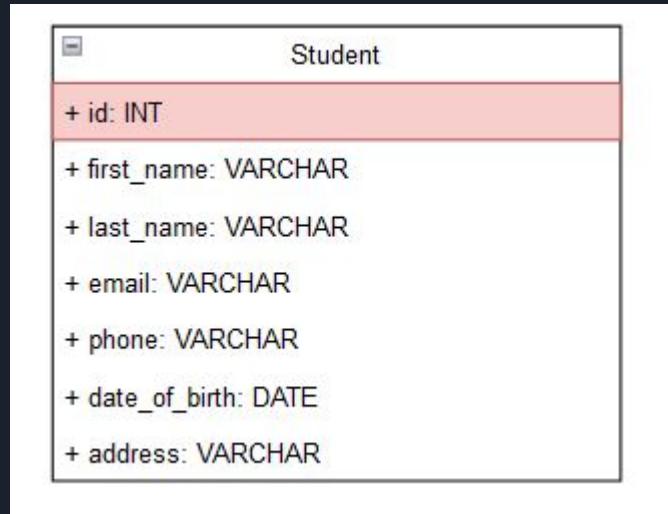
- Tạo một database lưu giữ thông tin của một trường học
- Các table cơ bản:
 - Student
 - Instructor

Student table

| Student | |
|---------|---------------------|
| + | id: INT |
| + | first_name: VARCHAR |
| + | last_name: VARCHAR |
| + | email: VARCHAR |
| + | phone: VARCHAR |
| + | date_of_birth: DATE |
| + | address: VARCHAR |

PRIMARY KEY

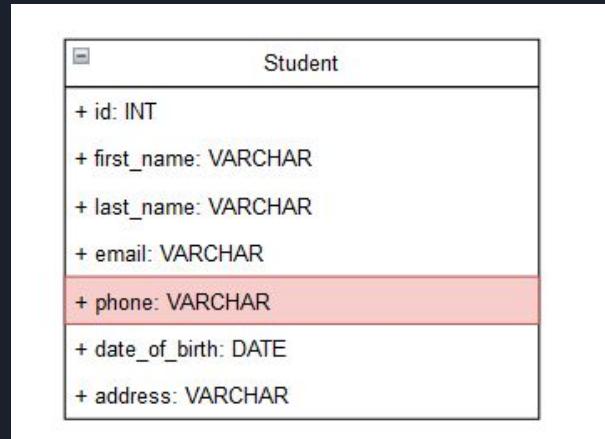
- Dùng để định danh duy nhất mỗi record **trong** table của cơ sở dữ liệu
 - Key tự nhiên (KTPM8989)
 - Key số
- Mỗi table chỉ có duy nhất một primary key



| id | first_name | last_name | email | phone_number | date_of_birth | address |
|-----------|-------------------|------------------|------------------------|---------------------|----------------------|----------------|
| 1 | Luan | Phan | luanphan@example.com | 991113 | 1995-10-10 | Somewhere |
| 1 | Luan | Phan | luanphan2@example.com | 991113 | 1995-10-11 | Somewhere 2 |
| 1 | Hung | Nguyen | hungnguyen@example.com | 991113 | 1995-10-10 | Somewhere |

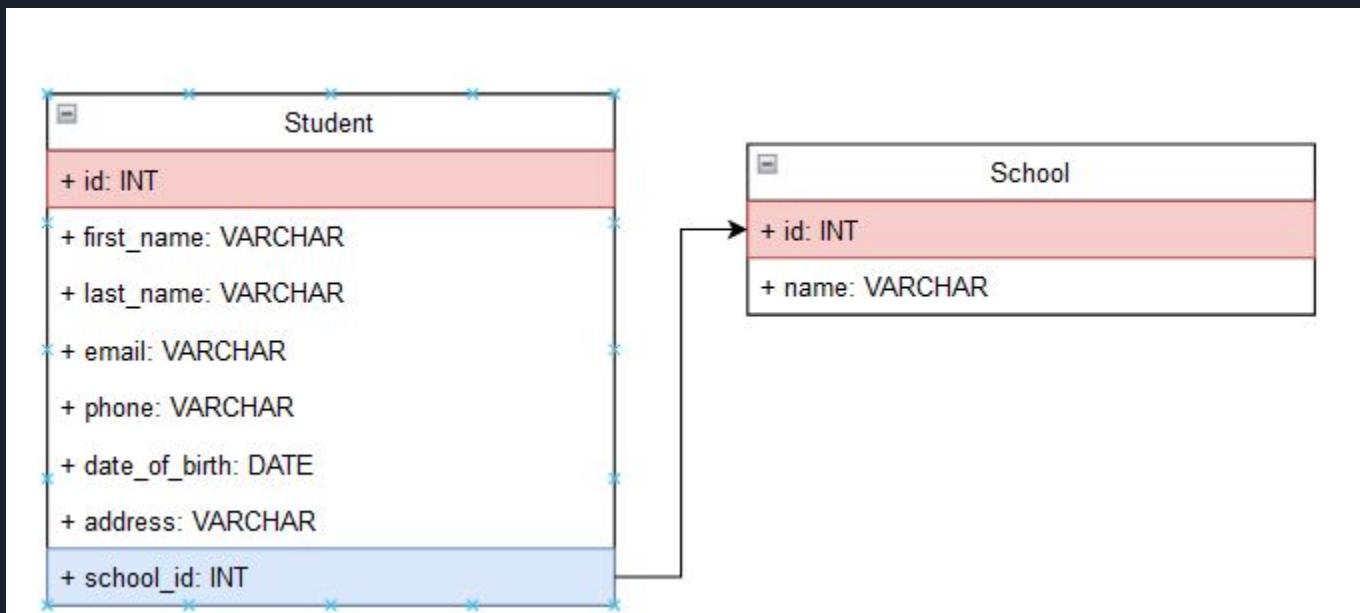
UNIQUE KEY

- Dùng để định danh duy nhất mỗi record **trong** table của cơ sở dữ liệu
- Mỗi table có thể có nhiều unique key
- Vì unique, nên chỉ có duy nhất 1 hàng có thể có NULL value

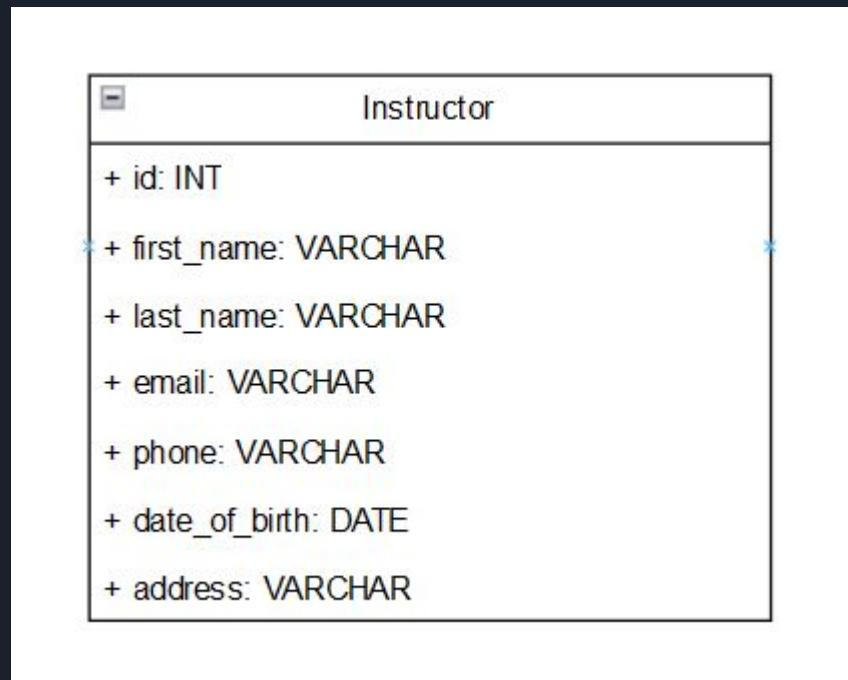


FOREIGN KEY

- Foreign key trong bảng là khóa tham chiếu đến khóa chính của table khác



Instructor table



Nếu ta muốn có thêm bảng hiệu trưởng?

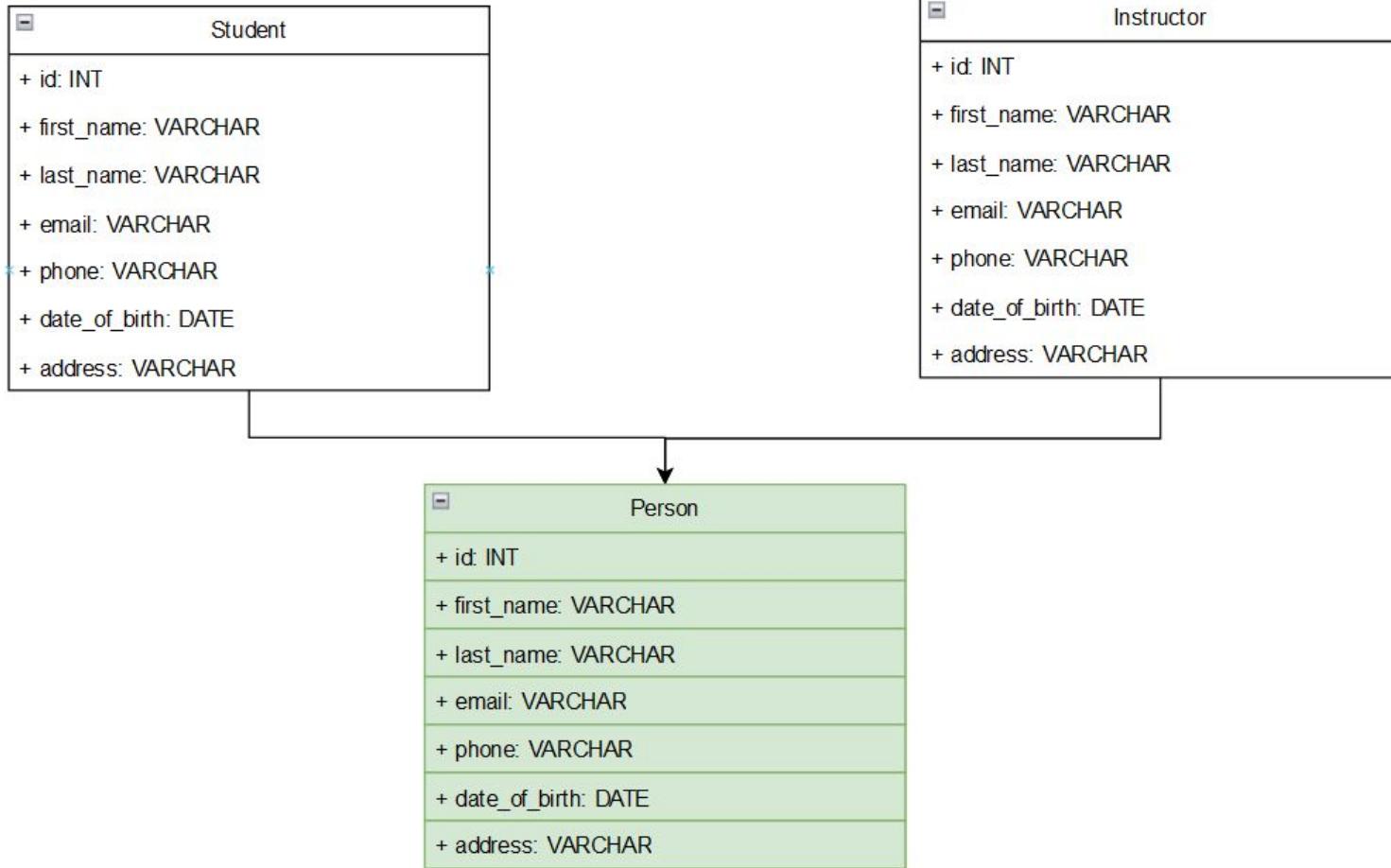
| President | |
|------------------|---------|
| + id: | INT |
| + first_name: | VARCHAR |
| + last_name: | VARCHAR |
| + email: | VARCHAR |
| + phone: | VARCHAR |
| + date_of_birth: | DATE |
| + address: | VARCHAR |



Nếu ta muốn có thêm bảng hiệu trưởng?

Ta thấy việc ứng với mỗi vị trí trong trường học, lại phải có một bảng.

Liệu ta có thể quy về chung một bảng để dễ dàng truy vấn hay không?



Làm sao để phân biệt instructor và student?

| Person | |
|------------------|-----------|
| + id: | INT |
| + first_name: | VARCHAR |
| * + last_name: | VARCHAR * |
| + email: | VARCHAR |
| + phone: | VARCHAR |
| + date_of_birth: | DATE |
| + address: | VARCHAR |
| + role: | VARCHAR |

- Nếu như có 1 bạn vừa là sinh viên, vừa là lớp trưởng, vừa là giảng viên thì sao?
 - Bảng phải có 3 hàng dành cho 1 người?
 - Role sẽ là string gồm cả 3 role? "Sinh viên, giang viên"
- Nếu sau này ta muốn đổi role name, từ giảng viên sang giáo viên thì sao?
 - Phải query hết tất cả các dòng và đổi?
- Khi đổi thông tin của một người, phải đổi tất cả các dòng?
 - Làm sao để biết tất cả các dòng là của một người?

First Normal Form (1NF)

- Mỗi ô trong bảng nên chứa một giá trị đơn
- Mỗi dòng cần phải duy nhất

| id | first_name | last_name | email | phone | date_of_birth | address | role |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|----------------------|
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 | Giang Vien, Hoc Sinh |

| id | first_name | last_name | email | phone | date_of_birth | address | role |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|-------------|
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 | Giang Vien |
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 | Hoc Sinh |

Second Normal Form (2NF)

- 1NF
- Tất cả các non-key attributes phải phụ thuộc vào primary key

| id | first_name | last_name | email | phone | date_of_birth | address | role |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|-------------|
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 | Giang Vien |
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 | Hoc Sinh |

- Xóa bỏ dữ liệu trùng lặp

2NF

- Tách bảng

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 1 | Luan | Phan | luanphan@example.com | 388654331 | 1941 | 123 Duong Van Qua, Quan 1 |

| person_id | role | role_detail |
|------------------|-------------|---------------------|
| 1 | Giang Vien | ngoi bam slide |
| 1 | Hoc Sinh | ngoi bam dien thoai |

Third Normal Form (3NF)

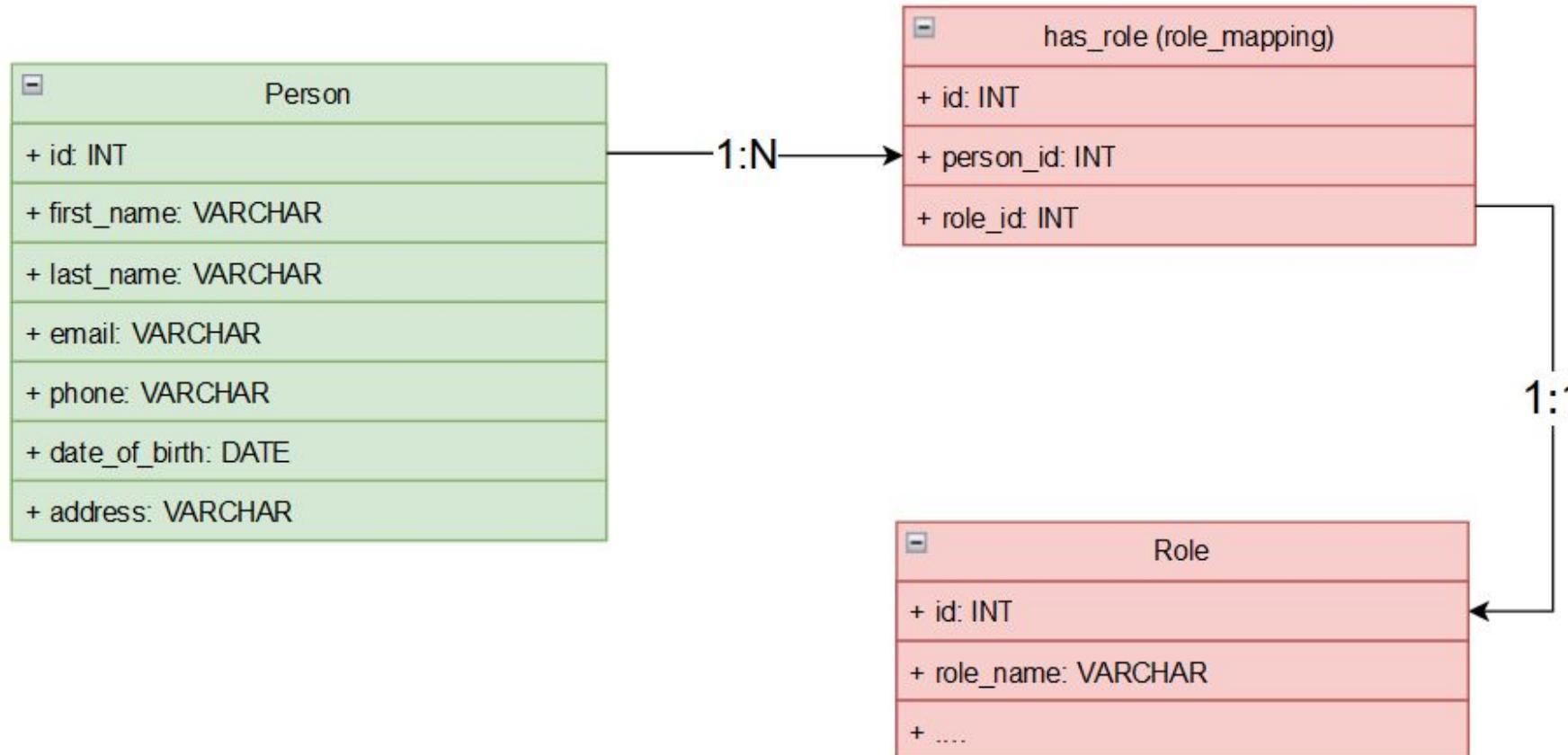
- 2NF
- No transitive dependency
 - Tất cả các key không phải primary phụ thuộc vào primary key
 - Và không phụ thuộc vào nhau

| person_id | role | role_detail |
|-----------|------------|---------------------|
| 1 | Giang Vien | ngoi bam slide |
| 1 | Hoc Sinh | ngoi bam dien thoai |
| 2 | Hoc Sinh | ngoi bam may tinh |



Third Normal Form (3NF)

- Tách bảng

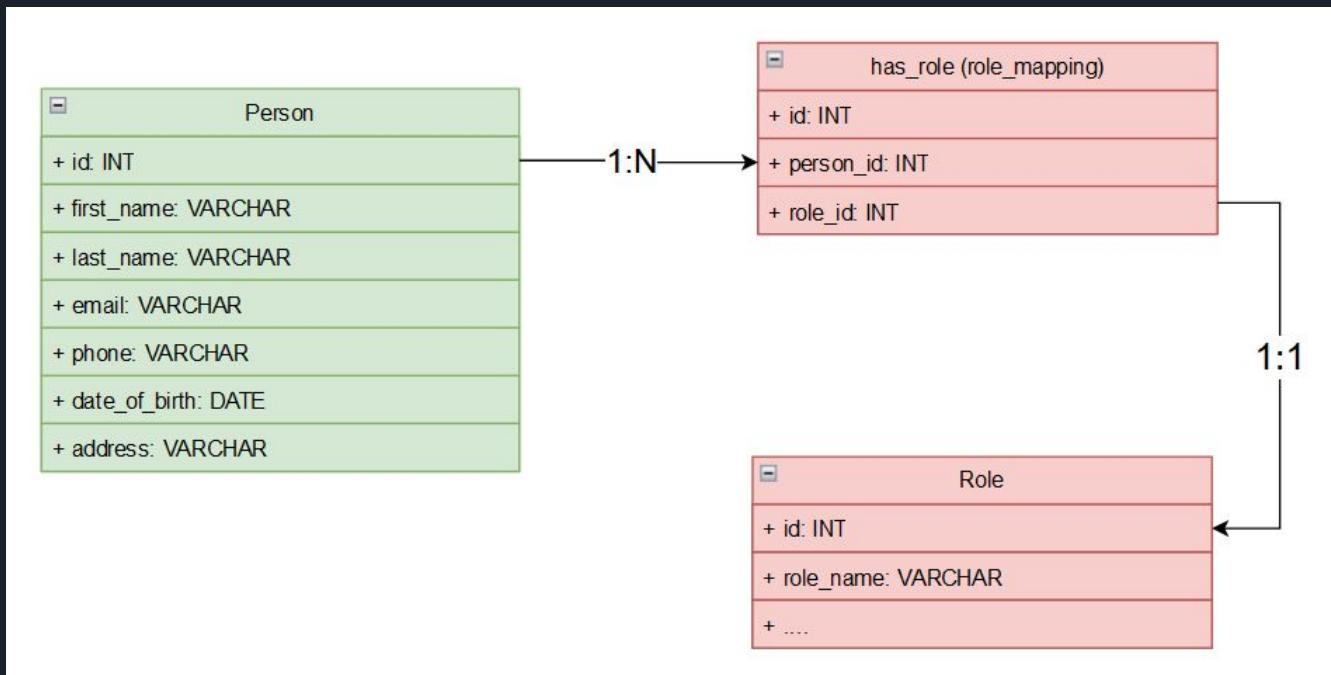


Làm sao để phân biệt instructor và student?

| Person | |
|------------------|-----------|
| + id: | INT |
| + first_name: | VARCHAR |
| * + last_name: | VARCHAR * |
| + email: | VARCHAR |
| + phone: | VARCHAR |
| + date_of_birth: | DATE |
| + address: | VARCHAR |
| + role: | VARCHAR |

- Nếu như có 1 bạn vừa là sinh viên, vừa là lớp trưởng, vừa là giảng viên thì sao?
 - Bảng phải có 3 hàng dành cho 1 người?
 - Role sẽ là string gồm cả 3 role?
 - -> Thêm has_role
- Nếu sau này ta muốn đổi role name, từ giảng viên sang giáo viên thì sao?
 - Phải query hết tất cả các dòng và đổi?
 - -> Update role_name in role table
- Khi đổi thông tin của một người, phải đổi tất cả các dòng?
 - Làm sao để biết tất cả các dòng là của một người?
 - -> Đổi duy nhất 1 dòng, vì đã đạt 3NF

Data trong bản person đã hoàn toàn normalize chưa?





Design Database

- Thêm các bảng để hỗ trợ
 - Phân chia lớp học (class)
 - các bạn sinh viên sẽ có 1 hoặc nhiều lớp
 - giảng viên sẽ giảng dạy 1 hoặc nhiều lớp
 - thông tin lớp học
 - Điểm danh
 - Bài kiểm tra và chấm điểm



Design Database

- Bảng và các loại khóa (primary, unique, foreign key)
- Các dạng chuẩn hóa normal form (1NF, 2NF, 3NF)
 - 1NF đảm bảo mỗi ô trong table là duy nhất -> từ đó đảm bảo sự tồn tại của primary key trong bảng
 - 2NF đảm bảo tất cả các thuộc tính trong bảng phụ thuộc vào primary key
 - 3NF đảm bảo tất cả các thuộc tính trong bảng phụ thuộc vào primary key và không phụ thuộc vào nhau
- -> Tránh tạo dữ liệu trùng lặp



Design Database

- Khi design hệ thống, không thể thiết kế hệ thống hoàn mỹ được
- Thiết kế và chỉnh sửa từ từ

CREATE TABLE

```
CREATE DATABASE IF NOT EXISTS school;

CREATE TABLE IF NOT EXISTS school.person (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    phone VARCHAR(20) UNIQUE,
    date_of_birth DATE,
    address VARCHAR(200)
);
```

```
1 CREATE TABLE IF NOT EXISTS school.role (
2     id INT PRIMARY KEY AUTO_INCREMENT,
3     role_name VARCHAR(50)
4 );
5
6 CREATE TABLE IF NOT EXISTS school.has_role (
7     id INT PRIMARY KEY AUTO_INCREMENT,
8     person_id INT,
9     role_id INT
10 );
11
```

INSERT DATA

```
INSERT INTO school.person (id, first_name, last_name, email, phone, date_of_birth, address)
VALUES
(1, 'Trung', 'Nguyen', 'trungnguyen@example.com', '0987654321', '1990-01-01', '123 Duong Le Loi, Quan 1'),
(2, 'Linh', 'Nguyen', 'linhnguyen@example.com', '0987654322', '1995-05-05', '456 Duong Nguyen Hue, Quan 3'),
(3, 'Hoa', 'Tran', 'hoatran@example.com', '0987654323', '1985-08-15', '789 Duong Vo Thi Sau, Quan 5'),
(4, 'Minh', 'Le', 'minhle@example.com', '0987654324', '1992-02-20', '321 Duong Cach Mang Thang Tam, Quan 10'),
(5, 'Phuong', 'Pham', 'phuongpham@example.com', '0987654325', '1980-11-11', '654 Duong Nam Ky Khoi, Quan 1'),
(6, 'An', 'Tran', 'antran@example.com', '0992654326', '1993-07-15', '456 Duong Tran Hung Dao, Quan 1'),
(7, 'Dat', 'Pham', 'datpham@example.com', '0987654327', '1997-03-20', '789 Duong Nam Ky Khoi Nghia, Quan 3'),
(8, 'Thu', 'Le', 'thule@example.com', '0987654328', '1989-12-01', '321 Duong Ly Thai To, Quan 10'),
(9, 'Tien', 'Vo', 'tienvo@example.com', '0963654329', '1991-06-10', '654 Duong Vo Van Tan, Quan 3'),
(10, 'Hien', 'Nguyen', 'hiennnguyen@example.com', '0963654330', '1998-11-28', '123 Duong Nguyen Hue, Quan 1'),
(11, 'Luan', 'Phan', 'luanphan@example.com', '0388654331', '1980-11-28', '123 Duong Van Qua, Quan 1');
```

INSERT DATA

```
INSERT INTO school.role (id, role_name)
VALUES
    (113, 'Hoc Sinh'),
    (911, 'Giang Vien'),
    (912, 'Tro Giang'),
    (2023, 'Hieu Truong');

INSERT INTO school.has_role (person_id, role_id)
VALUES
    (1, 113),
    (2, 113),
    (3, 113),
    (4, 113),
    (5, 113),
    (6, 113),
    (7, 113),
    (8, 113),
    (9, 113),
    (10, 113),
    (11, 113), # id: 11, vua la Hoc Sinh
    (11, 911); # Vua la giang vien
```



Truy vấn dữ liệu (SQL)

- **SELECT**
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING

SELECT (ALL)

Tìm tất cả thông tin của mọi người trong bảng person?

```
SELECT * FROM person;
```

| Result of SELECT * FROM person; | | | | | | |
|---------------------------------|------------|-----------|-------------------------|------------|---------------|--|
| id | first_name | last_name | email | phone | date_of_birth | address |
| 1 | Trung | Nguyen | trungnguyen@example.com | 0987654321 | 1990-01-01 | 123 Duong Le Loi, Quan 1 |
| 2 | Linh | Nguyen | linhnguyen@example.com | 0987654322 | 1995-05-05 | 456 Duong Nguyen Hue, Quan 3 |
| 3 | Hoa | Tran | hoatran@example.com | 0987654323 | 1985-08-15 | 789 Duong Vo Thi Sau, Quan 5 |
| 4 | Minh | Le | minhle@example.com | 0987654324 | 1992-02-20 | 321 Duong Cach Mang Thang Tam, Quan 10 |
| 5 | Phuong | Pham | phuongpham@example.com | 0987654325 | 1980-11-11 | 654 Duong Nam Ky Khoi, Quan 1 |
| 6 | An | Tran | antran@example.com | 0992654326 | 1993-07-15 | 456 Duong Tran Hung Dao, Quan 1 |
| 7 | Dat | Pham | datpham@example.com | 0987654327 | 1997-03-20 | 789 Duong Nam Ky Khoi Nghia, Quan 3 |
| 8 | Thu | Le | thule@example.com | 0987654328 | 1989-12-01 | 321 Duong Ly Thai To, Quan 10 |
| 9 | Tien | Vo | tienvo@example.com | 0963654329 | 1991-06-10 | 654 Duong Vo Van Tan, Quan 3 |
| 10 | Hien | Nguyen | hiennnguyen@example.com | 0963654330 | 1998-11-28 | 123 Duong Nguyen Hue, Quan 1 |
| 11 | Luan | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |

11 rows in set (0.00 sec)

SELECT (ALL) LIMIT

Tìm tất cả thông tin của mọi người trong bảng person?

```
SELECT * FROM person LIMIT 5;
```

```
mysql> SELECT * FROM person LIMIT 5;
+----+-----+-----+-----+-----+-----+-----+
| id | first_name | last_name | email           | phone      | date_of_birth | address
+----+-----+-----+-----+-----+-----+-----+
| 1  | Trung      | Nguyen     | trungnguyen@example.com | 0987654321 | 1990-01-01   | 123 Duong Le Loi, Quan 1
| 2  | Linh       | Nguyen     | linhnguyen@example.com | 0987654322 | 1995-05-05   | 456 Duong Nguyen Hue, Quan 3
| 3  | Hoa        | Tran       | hoatran@example.com  | 0987654323 | 1985-08-15   | 789 Duong Vo Thi Sau, Quan 5
| 4  | Minh       | Le          | minhle@example.com  | 0987654324 | 1992-02-20   | 321 Duong Cach Mang Thang Tam, Quan 10
| 5  | Phuong     | Pham       | phuongpham@example.com | 0987654325 | 1980-11-11   | 654 Duong Nam Ky Khoi, Quan 1
+----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql> |
```

SELECT (Some columns)

Lấy danh sách họ, tên, địa chỉ của tất cả mọi người trong bảng person?

```
SELECT first_name, last_name, email FROM person;
```

```
mysql> SELECT
    -> first_name, last_name, email
    -> FROM
    -> person;
+-----+-----+-----+
| first_name | last_name | email
+-----+-----+-----+
| Trung      | Nguyen   | trungnguyen@example.com
| Linh       | Nguyen   | linhnguyen@example.com
| Hoa        | Tran     | hoatran@example.com
| Minh       | Le        | minhle@example.com
| Phuong     | Pham     | phuongpham@example.com
| An         | Tran     | antran@example.com
| Dat        | Pham     | datpham@example.com
| Thu         | Le        | thule@example.com
| Tien        | Vo        | tienvo@example.com
| Hien        | Nguyen   | hiennnguyen@example.com
| Luan        | Phan     | luanphan@example.com
+-----+-----+-----+
11 rows in set (0.00 sec)
```

SELECT (Some columns and name them)

Lấy danh sách họ tên kèm địa chỉ của tất cả mọi người trong bảng person?

```
mysql> SELECT
    -> CONCAT(last_name, " ", first_name) AS full_name, email
    -> FROM
    -> person;
+-----+-----+
| full_name | email           |
+-----+-----+
| Nguyen Trung | trungnguyen@example.com |
| Nguyen Linh | linhnguyen@example.com |
| Tran Hoa | hoatran@example.com |
| Le Minh | minhle@example.com |
| Pham Phuong | phuongpham@example.com |
| Tran An | antran@example.com |
| Pham Dat | datpham@example.com |
| Le Thu | thule@example.com |
| Vo Tien | tienvo@example.com |
| Nguyen Hien | hiennguyen@example.com |
| Phan Luan | luanphan@example.com |
+-----+-----+
11 rows in set (0.01 sec)
```

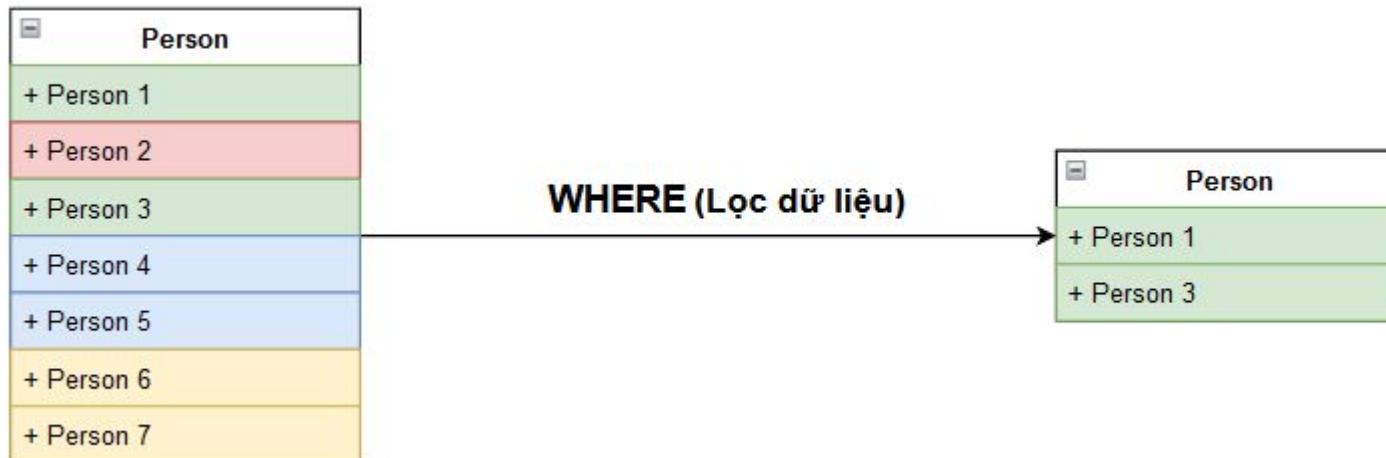


Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING

WHERE

Dùng để lọc dữ liệu trên toàn bộ table theo điều kiện nhất định



Where (mệnh đề điều kiện)

Tìm tất cả các bạn có họ là “Nguyen” trong bảng person?

```
SELECT * FROM person WHERE last_name = "Nguyen"
```

```
mysql> SELECT *
-> FROM person
-> WHERE last_name = "Nguyen";
+----+-----+-----+-----+-----+-----+-----+
| id | first_name | last_name | email           | phone      | date_of_birth | address        |
+----+-----+-----+-----+-----+-----+-----+
|  1 | Trung      | Nguyen     | trungnguyen@example.com | 0987654321 | 1990-01-01   | 123 Duong Le Loi, Quan 1 |
|  2 | Linh       | Nguyen     | linhnguyen@example.com | 0987654322 | 1995-05-05   | 456 Duong Nguyen Hue, Quan 3 |
| 10 | Hien       | Nguyen     | hiennguyen@example.com | 0987654330 | 1998-11-28   | 123 Duong Nguyen Hue, Quan 1 |
+----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> |
```



Where (mệnh đề điều kiện)

Tìm tất cả các bạn trong bảng person có độ dài last_name nhiều hơn 2 kí tự?

SELECT Cols WHERE

Lấy danh sách họ tên kèm địa chỉ của tất cả mọi người **có họ là “Nguyen”** trong bảng person?

```
mysql> SELECT
-> CONCAT(first_name, " ", last_name) as full_name, address
-> FROM person
-> WHERE last_name = "Nguyen";
+-----+-----+
| full_name | address          |
+-----+-----+
| Trung Nguyen | 123 Duong Le Loi, Quan 1 |
| Linh Nguyen | 456 Duong Nguyen Hue, Quan 3 |
| Hien Nguyen | 123 Duong Nguyen Hue, Quan 1 |
+-----+-----+
3 rows in set (0.00 sec)
```



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING

ORDER BY (sắp xếp)

Lấy danh sách tất cả các bạn trong bảng table, sắp xếp tên theo thứ tự tăng dần

```
mysql> SELECT *  
-> FROM person  
-> ORDER BY first_name ASC;  
+----+----+----+----+----+----+----+----+  
| id | first_name | last_name | email | phone | date_of_birth | address |  
+----+----+----+----+----+----+----+----+  
| 6  | An          | Tran       | antran@example.com | 0992654326 | 1993-07-15    | 456 Duong Tran Hung Dao, Quan 1  
| 7  | Dat          | Pham       | datpham@example.com | 0987654327 | 1997-03-20    | 789 Duong Nam Ky Khoi Nghia, Quan 3  
| 10 | Hien         | Nguyen     | hiennguyen@example.com | 0963654330 | 1998-11-28    | 123 Duong Nguyen Hue, Quan 1  
| 3  | Hoa          | Tran       | hoatran@example.com | 0987654323 | 1985-08-15    | 789 Duong Vo Thi Sau, Quan 5  
| 2  | Linh          | Nguyen     | linhnguyen@example.com | 0987654322 | 1995-05-05    | 456 Duong Nguyen Hue, Quan 3  
| 11 | Luan          | Phan       | luanphan@example.com | 0388654331 | 1980-11-28    | 123 Duong Van Qua, Quan 1  
| 4  | Minh          | Le          | minhle@example.com | 0987654324 | 1992-02-20    | 321 Duong Cach Mang Thang Tam, Quan 10  
| 5  | Phuong        | Pham       | phuongpham@example.com | 0987654325 | 1980-11-11    | 654 Duong Nam Ky Khoi, Quan 1  
| 8  | Thu           | Le          | thule@example.com | 0987654328 | 1989-12-01    | 321 Duong Ly Thai To, Quan 10  
| 9  | Tien          | Vo          | tienvo@example.com | 0963654329 | 1991-06-10    | 654 Duong Vo Van Tan, Quan 3  
| 1  | Trung         | Nguyen     | trungnguyen@example.com | 0987654321 | 1990-01-01    | 123 Duong Le Loi, Quan 1  
+----+----+----+----+----+----+----+----+  
11 rows in set (0.01 sec)
```

ORDER BY (sắp xếp)

Lấy danh sách tất cả các bạn trong bảng table, sắp xếp tên theo thứ tự giảm dần

```
mysql> SELECT *  
-> FROM person  
-> ORDER BY first_name DESC;  
+----+----+----+----+----+----+----+  
| id | first_name | last_name | email | phone | date_of_birth | address |  
+----+----+----+----+----+----+----+  
| 1  | Trung      | Nguyen    | trungnguyen@example.com | 0987654321 | 1990-01-01 | 123 Duong Le Loi, Quan 1  
| 9  | Tien       | Vo        | tienvo@example.com   | 0963654329 | 1991-06-10 | 654 Duong Vo Van Tan, Quan 3  
| 8  | Thu        | Le        | thule@example.com   | 0987654328 | 1989-12-01 | 321 Duong Ly Thai To, Quan 10  
| 5  | Phuong     | Pham      | phuongpham@example.com | 0987654325 | 1980-11-11 | 654 Duong Nam Ky Khoi, Quan 1  
| 4  | Minh       | Le        | minhle@example.com | 0987654324 | 1992-02-20 | 321 Duong Cach Mang Thang Tam, Quan 10  
| 11 | Luan       | Phan      | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1  
| 2  | Linh       | Nguyen    | linhnguyen@example.com | 0987654322 | 1995-05-05 | 456 Duong Nguyen Hue, Quan 3  
| 3  | Hoa        | Tran      | hoatran@example.com | 0987654323 | 1985-08-15 | 789 Duong Vo Thi Sau, Quan 5  
| 10 | Hien       | Nguyen    | hiennnguyen@example.com | 0963654330 | 1998-11-28 | 123 Duong Nguyen Hue, Quan 1  
| 7  | Dat        | Pham      | datpham@example.com | 0987654327 | 1997-03-20 | 789 Duong Nam Ky Khoi Nghia, Quan 3  
| 6  | An         | Tran      | antran@example.com  | 0992654326 | 1993-07-15 | 456 Duong Tran Hung Dao, Quan 1  
+----+----+----+----+----+----+----+
```

11 rows in set (0.00 sec)



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- **DISTINCT**
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING

DISTINCT (độc nhất)

Lấy danh sách tất cả các họ trong bảng person

```
mysql> SELECT last_name  
      -> FROM person;  
+-----+  
| last_name |  
+-----+  
| Nguyen   |  
| Nguyen   |  
| Tran     |  
| Le       |  
| Pham     |  
| Tran     |  
| Pham     |  
| Le       |  
| Vo       |  
| Nguyen   |  
| Phan     |  
+-----+  
11 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT(last_name)  
      -> FROM person;  
+-----+  
| last_name |  
+-----+  
| Nguyen   |  
| Tran     |  
| Le       |  
| Pham     |  
| Vo       |  
| Phan     |  
+-----+  
6 rows in set (0.00 sec)
```



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- **LIKE**
- AGGREGATIONS
- GROUP BY
- HAVING



LIKE (tương tự)

Dùng để truy vấn dữ liệu có tính tương đồng

- %abc
- abc%
- %abc%

Like

Lấy danh sách các bạn có tên kết thúc bằng chữ cái “n”?

```
mysql> SELECT *  
-> FROM person  
-> WHERE first_name LIKE "%n"  
-> ;  
+-----+-----+-----+-----+-----+-----+-----+  
| id | first_name | last_name | email | phone | date_of_birth | address |  
+-----+-----+-----+-----+-----+-----+-----+  
| 6 | An | Tran | antran@example.com | 0992654326 | 1993-07-15 | 456 Duong Tran Hung Dao, Quan 1 |  
| 9 | Tien | Vo | tienvo@example.com | 0963654329 | 1991-06-10 | 654 Duong Vo Van Tan, Quan 3 |  
| 10 | Hien | Nguyen | hiennguyen@example.com | 0963654330 | 1998-11-28 | 123 Duong Nguyen Hue, Quan 1 |  
| 11 | Luan | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |  
+-----+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

Like

Lấy danh sách các bạn có số điện thoại bắt đầu từ 0963?

```
mysql> SELECT *  
-> FROM person  
-> WHERE phone LIKE "0963%";  
+----+----+----+----+----+----+  
| id | first_name | last_name | email | phone | date_of_birth | address |  
+----+----+----+----+----+----+  
| 9 | Tien | Vo | tienvo@example.com | 0963654329 | 1991-06-10 | 654 Duong Vo Van Tan, Quan 3 |  
| 10 | Hien | Nguyen | hiennguyen@example.com | 0963654330 | 1998-11-28 | 123 Duong Nguyen Hue, Quan 1 |  
+----+----+----+----+----+----+  
2 rows in set (0.00 sec)
```



Like

Lấy danh sách các bạn có email “@example”

Tổng hợp

Lấy danh sách họ tên, địa chỉ, số điện thoại của các bạn ở Quận 1 sắp xếp tên theo thứ tự tăng dần

```
mysql> SELECT CONCAT(first_name, " ", last_name) as full_name, address, phone
-> FROM person
-> WHERE address LIKE "%Quan 1"
-> ORDER BY last_name ASC;
```

| full_name | address | phone |
|--------------|---------------------------------|------------|
| Trung Nguyen | 123 Duong Le Loi, Quan 1 | 0987654321 |
| Hien Nguyen | 123 Duong Nguyen Hue, Quan 1 | 0963654330 |
| Phuong Pham | 654 Duong Nam Ky Khoi, Quan 1 | 0987654325 |
| Luan Phan | 123 Duong Van Qua, Quan 1 | 0388654331 |
| An Tran | 456 Duong Tran Hung Dao, Quan 1 | 0992654326 |

```
5 rows in set (0.00 sec)
```

Tổng hợp

Lấy danh sách họ tên, địa chỉ, số điện thoại của các bạn ở Quận 1 có họ Nguyen và số điện thoại bắt đầu bằng 0963

```
mysql> SELECT CONCAT(first_name, " ", last_name) as full_name, address, phone
-> FROM person
-> WHERE address LIKE "%Quan 1"
-> ORDER BY last_name ASC;
+-----+-----+
| full_name | address | phone |
+-----+-----+
| Trung Nguyen | 123 Duong Le Loi, Quan 1 | 0987654321 |
| Hien Nguyen | 123 Duong Nguyen Hue, Quan 1 | 0963654330 |
| Phuong Pham | 654 Duong Nam Ky Khoi, Quan 1 | 0987654325 |
| Luan Phan | 123 Duong Van Qua, Quan 1 | 0388654331 |
| An Tran | 456 Duong Tran Hung Dao, Quan 1 | 0992654326 |
+-----+-----+
5 rows in set (0.00 sec)
```



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING



Aggregations

- COUNT
- SUM
- AVG
- MIN
- MAX

COUNT

Đếm tất cả các bạn có họ là Nguyen

```
mysql> SELECT COUNT(*)
      -> FROM person
      -> WHERE last_name = "Nguyen";
+-----+
| COUNT(*) |
+-----+
|      3   |
+-----+
1 row in set (0.00 sec)
```

SUM

Tính tổng số tuổi của tất cả các bạn trong bảng person

```
mysql> SELECT date_of_birth, 2023 - YEAR(date_of_birth) as age
-> FROM person
-> ;
+-----+----+
| date_of_birth | age |
+-----+----+
| 1990-01-01    | 33  |
| 1995-05-05    | 28  |
| 1985-08-15    | 38  |
| 1992-02-20    | 31  |
| 1980-11-11    | 43  |
| 1993-07-15    | 30  |
| 1997-03-20    | 26  |
| 1989-12-01    | 34  |
| 1991-06-10    | 32  |
| 1998-11-28    | 25  |
| 1980-11-28    | 43  |
+-----+----+
11 rows in set (0.00 sec)
```

SUM

Dùng hàm YEAR(date type column) để lấy năm ra khỏi cột date_of_birth

```
mysql> SELECT date_of_birth, 2023 - YEAR(date_of_birth) as age
    -> FROM person
    -> ;
+-----+----+
| date_of_birth | age |
+-----+----+
| 1990-01-01    | 33  |
| 1995-05-05    | 28  |
| 1985-08-15    | 38  |
| 1992-02-20    | 31  |
| 1980-11-11    | 43  |
| 1993-07-15    | 30  |
| 1997-03-20    | 26  |
| 1989-12-01    | 34  |
| 1991-06-10    | 32  |
| 1998-11-28    | 25  |
| 1980-11-28    | 43  |
+-----+----+
11 rows in set (0.00 sec)
```

SUM

Tính tổng số tuổi của tất cả các bạn trong bảng person

```
mysql> SELECT SUM(2023 - YEAR(date_of_birth)) as sum_age
-> FROM
-> person;
+-----+
| sum_age |
+-----+
|      363 |
+-----+
1 row in set (0.00 sec)
```

SUM & COUNT -> AVG

Tính trung bình số tuổi của tất cả các bạn trong bảng person

```
mysql> SELECT SUM(2023 - YEAR(date_of_birth))/COUNT(*) as average_age  
      -> FROM  
      -> person;  
+-----+  
| average_age |  
+-----+  
|      33.0000 |  
+-----+  
1 row in set (0.00 sec)
```

AVG

Tính trung bình số tuổi của tất cả các bạn trong bảng person

```
mysql> SELECT AVG(2023 - YEAR(date_of_birth)) as average_age
-> FROM
-> person;
+-----+
| average_age |
+-----+
|      33.0000 |
+-----+
1 row in set (0.01 sec)
```

MAX

Tìm tuổi lớn nhất trong bảng person

```
mysql> SELECT MAX(2023 - YEAR(date_of_birth)) as max_age FROM person;
+-----+
| max_age |
+-----+
|      43 |
+-----+
1 row in set (0.00 sec)
```

MIN

Tìm tuổi nhỏ nhất trong bảng person

```
mysql> SELECT MIN(2023 - YEAR(date_of_birth)) as max_age FROM person;
+-----+
| max_age |
+-----+
|      25 |
+-----+
1 row in set (0.01 sec)
```

MIN

Tìm tuổi nhỏ nhất trong bảng person

```
mysql> SELECT MIN(2023 - YEAR(date_of_birth)) as max_age FROM person;
+-----+
| max_age |
+-----+
|      25 |
+-----+
1 row in set (0.01 sec)
```



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện) -> lọc dữ liệu
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- **GROUP BY**
- HAVING



GROUP BY

Lấy danh sách số lượng tất cả các bạn có cùng họ

- GROUP BY tất theo cột last_name (họ)
- Dùng COUNT để đếm

GROUP BY

Lấy danh sách số lượng tất cả các bạn có cùng họ

```
mysql> SELECT * FROM person;
+----+-----+-----+-----+-----+-----+-----+
| id | first_name | last_name | email           | phone      | date_of_birth | address
+----+-----+-----+-----+-----+-----+-----+
| 1  | Trung      | Nguyen    | trungnguyen@example.com | 0987654321 | 1990-01-01   | 123 Duong Le Loi, Quan 1
| 2  | Linh       | Nguyen    | linhnguyen@example.com | 0987654322 | 1995-05-05   | 456 Duong Nguyen Hue, Quan 3
| 3  | Hoa        | Tran      | hoatran@example.com  | 0987654323 | 1985-08-15   | 789 Duong Vo Thi Sau, Quan 5
| 4  | Minh       | Le         | minhle@example.com  | 0987654324 | 1992-02-20   | 321 Duong Cach Mang Thang Tam, Quan 10
| 5  | Phuong     | Pham      | phuongpham@example.com | 0987654325 | 1980-11-11   | 654 Duong Nam Ky Khoi, Quan 1
| 6  | An          | Tran      | antran@example.com  | 0992654326 | 1993-07-15   | 456 Duong Tran Hung Dao, Quan 1
| 7  | Dat         | Pham      | datpham@example.com | 0987654327 | 1997-03-20   | 789 Duong Nam Ky Khoi Nghia, Quan 3
| 8  | Thu          | Le         | thule@example.com  | 0987654328 | 1989-12-01   | 321 Duong Ly Thai To, Quan 10
| 9  | Tien         | Vo         | tienvo@example.com | 0963654329 | 1991-06-10   | 654 Duong Vo Van Tan, Quan 3
| 10 | Hien         | Nguyen    | hiennguyen@example.com | 0963654330 | 1998-11-28   | 123 Duong Nguyen Hue, Quan 1
| 11 | Luan         | Phan      | luanphan@example.com | 0388654331 | 1980-11-28   | 123 Duong Van Qua, Quan 1
+----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

GROUP BY

Lấy danh sách số lượng tất cả các bạn có cùng họ

```
mysql> SELECT last_name, COUNT(*)
-> FROM person
-> GROUP BY last_name;
+-----+-----+
| last_name | COUNT(*) |
+-----+-----+
| Le        |      2 |
| Nguyen    |      3 |
| Pham      |      2 |
| Phan      |      1 |
| Tran      |      2 |
| Vo         |      1 |
+-----+-----+
6 rows in set (0.00 sec)
```



GROUP BY

Tìm tuổi trong bình của tất cả các bạn có cùng họ?

- GROUP BY tất theo cột last_name (họ)
- Dùng AVG để tính trung bình

GROUP BY

Tìm tuổi trung bình của tất cả các bạn có cùng họ?

```
mysql> SELECT last_name, AVG(2023 - YEAR(date_of_birth))
-> FROM person
-> GROUP BY last_name;
+-----+-----+
| last_name | AVG(2023 - YEAR(date_of_birth)) |
+-----+-----+
| Le        |          32.5000 |
| Nguyen    |          28.6667 |
| Pham      |          34.5000 |
| Phan      |          43.0000 |
| Tran      |          34.0000 |
| Vo         |          32.0000 |
+-----+-----+
6 rows in set (0.00 sec)
```



Truy vấn dữ liệu (SQL)

- SELECT
 - * (Tất cả)
 - Columns (một vài cột)
 - Columns and rename (đổi tên cột)
- WHERE (mệnh đề điều kiện)
- ORDER BY
- DISTINCT
- LIKE
- AGGREGATIONS
- GROUP BY
- HAVING



HAVING

- HAVING được dùng vì WHERE không thể sử dụng cho với các hàm aggregations

HAVING

Lấy danh sách trung bình tuổi theo last_name trên 30?

```
mysql> SELECT last_name, AVG(2023 - YEAR(date_of_birth))
-> FROM person
-> GROUP BY last_name
-> HAVING AVG(2023 - YEAR(date_of_birth)) > 30;
+-----+-----+
| last_name | AVG(2023 - YEAR(date_of_birth)) |
+-----+-----+
| Le        |      32.5000 |
| Pham      |      34.5000 |
| Phan      |      43.0000 |
| Tran      |      34.0000 |
| Vo        |      32.0000 |
+-----+-----+
5 rows in set (0.00 sec)
```

HAVING

Lấy danh sách tuổi lớn nhất theo từng họ ở quận 1?

```
mysql> SELECT last_name, AVG(2023 - YEAR(date_of_birth))
-> FROM person
-> GROUP BY last_name
-> HAVING AVG(2023 - YEAR(date_of_birth)) > 30;
+-----+-----+
| last_name | AVG(2023 - YEAR(date_of_birth)) |
+-----+-----+
| Le        |      32.5000 |
| Pham      |      34.5000 |
| Phan      |      43.0000 |
| Tran      |      34.0000 |
| Vo        |      32.0000 |
+-----+-----+
5 rows in set (0.00 sec)
```

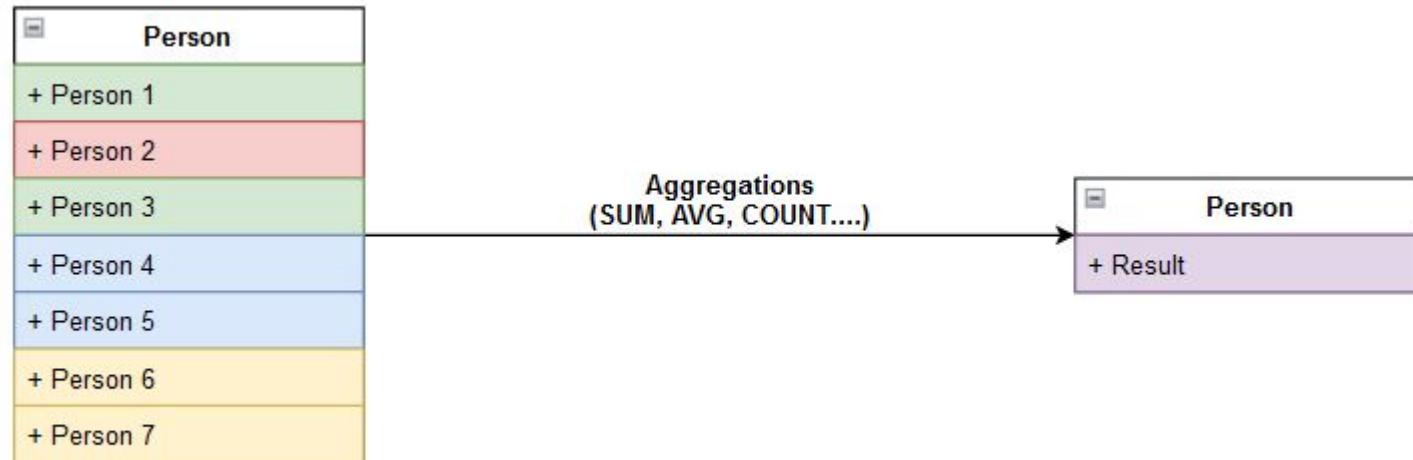
WHERE

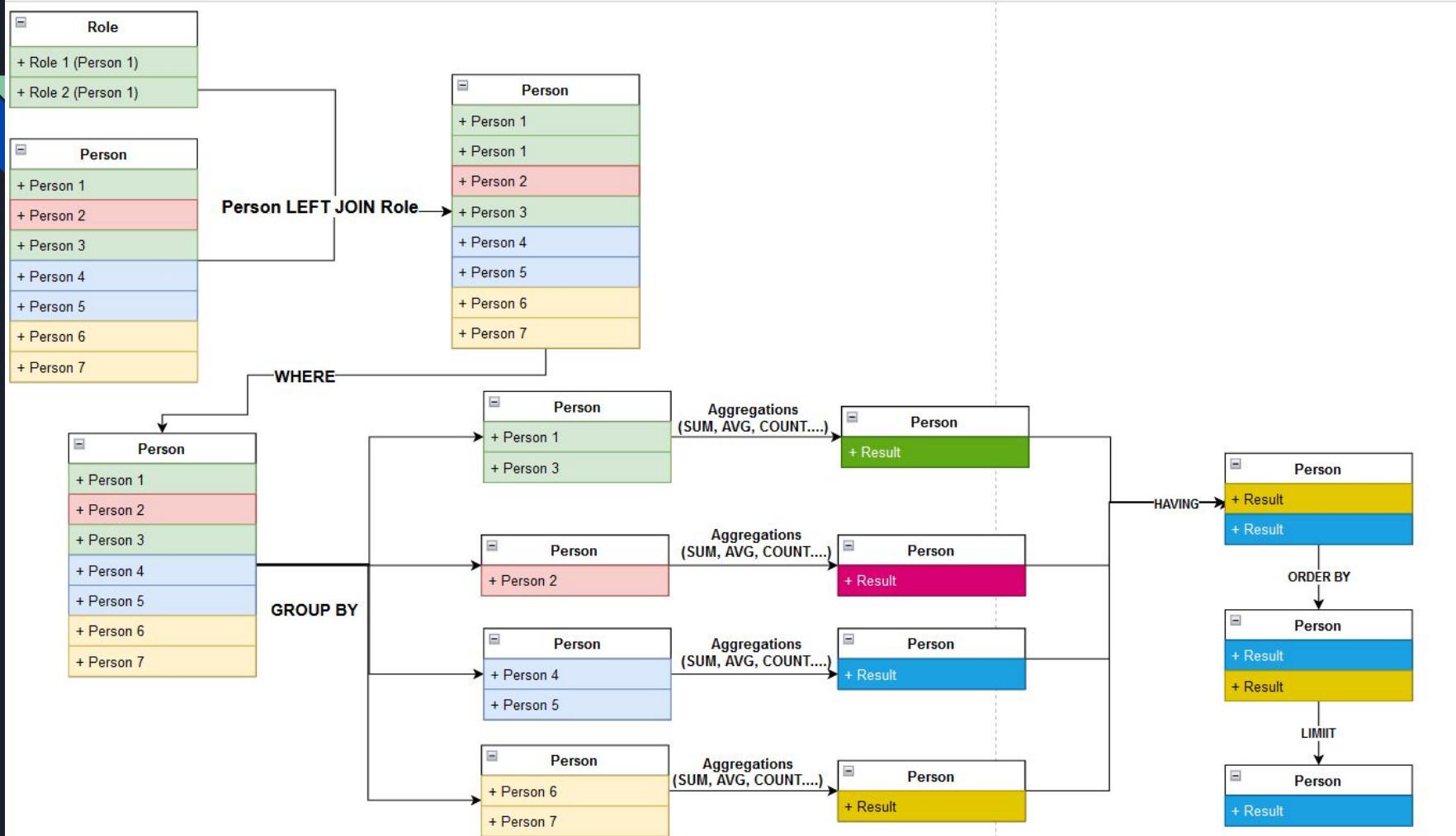
| Person |
|------------|
| + Person 1 |
| + Person 2 |
| + Person 3 |
| + Person 4 |
| + Person 5 |
| + Person 6 |
| + Person 7 |

WHERE (Lọc dữ liệu)

| Person |
|------------|
| + Person 1 |
| + Person 3 |

AGGREGATIONS





Question

Lấy danh sách tuổi trung bình của các bạn ở Quận 1 có cùng họ và tuổi trung bình lớn hơn 30

```
mysql> SELECT CONCAT(first_name, " ", last_name) as full_name, address, phone, (2023 - YEAR(date_of_birth)) as age
-> FROM person
-> WHERE address LIKE "%Quan 1";
+-----+-----+-----+
| full_name | address           | phone      | age   |
+-----+-----+-----+
| Trung Nguyen | 123 Duong Le Loi, Quan 1 | 0987654321 | 33   |
| Phuong Pham | 654 Duong Nam Ky Khoi, Quan 1 | 0987654325 | 43   |
| An Tran     | 456 Duong Tran Hung Dao, Quan 1 | 0992654326 | 30   |
| Hien Nguyen | 123 Duong Nguyen Hue, Quan 1  | 0963654330 | 25   |
| Luan Phan   | 123 Duong Van Qua, Quan 1   | 0388654331 | 43   |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Question

Lấy danh sách tuổi trung bình của các bạn ở Quận 1 có cùng họ và tuổi trung bình lớn hơn 30

```
mysql> SELECT last_name, AVG(2023 - YEAR(date_of_birth)) as age
-> FROM person
-> WHERE address LIKE "%Quan 1"
-> GROUP BY last_name
-> HAVING AVG(2023 - YEAR(date_of_birth)) > 30;
+-----+-----+
| last_name | age      |
+-----+-----+
| Pham      | 43.0000 |
| Phan      | 43.0000 |
+-----+-----+
2 rows in set (0.01 sec)
```

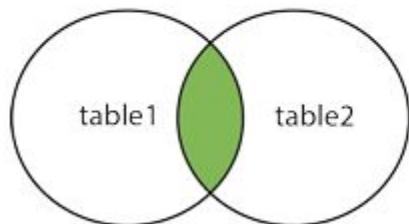


Tổng quan

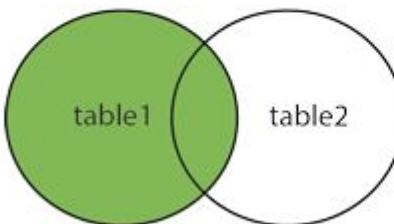
1. Giới thiệu chung về database
2. Các kiểu dữ liệu cở bản trong MySQL
3. Mô phỏng dữ liệu thực tế
4. Database table (bảng dữ liệu) là gì, cách tạo table
5. Truy vấn (Query) dữ liệu
6. **Join table**
7. Sub Query
8. Tối ưu câu lệnh
9. Link github phần sql

JOINS

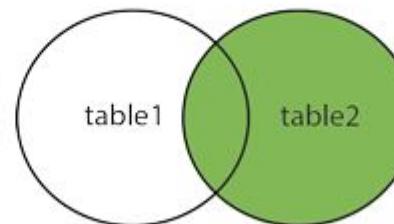
INNER JOIN



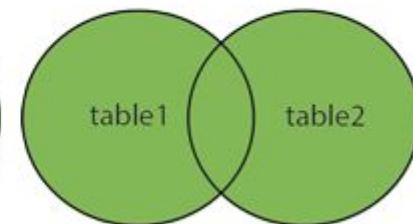
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN

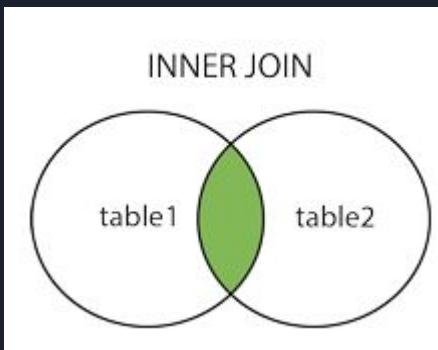


INNER JOIN

```
INSERT INTO school.role (id, role_name)
VALUES
    (113, 'Hoc Sinh'),
    (911, 'Giang Vien'),
    (912, 'Tro Giang'),
    (2023, 'Hieu Truong');

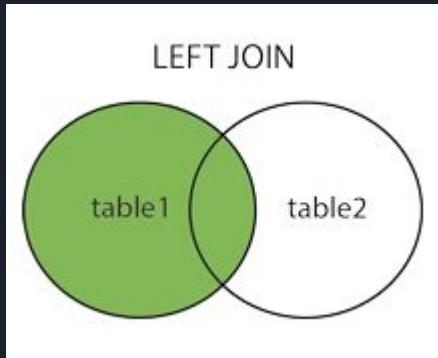
INSERT INTO school.has_role (person_id, role_id)
VALUES
    (1, 113),
    (2, 113),
    (3, 113),
    (4, 113),
    (5, 113),
    (6, 113),
    (7, 113),
    (8, 113),
    (9, 113),
    (10, 113),
    (11, 113), # id: 11, vua la Hoc Sinh
    (11, 911); # Vua la giang vien
```

(INNER) JOIN



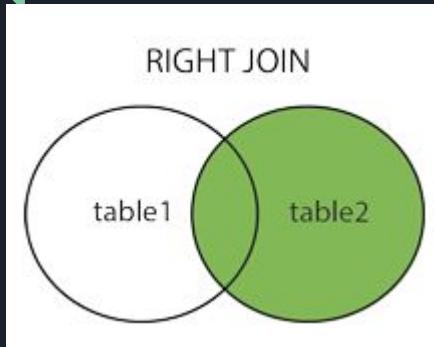
```
mysql> SELECT * FROM
-> role INNER JOIN has_role
-> ON role.id = has_role.role_id;
+----+-----+----+-----+----+
| id | role_name | id | person_id | role_id |
+----+-----+----+-----+----+
| 113 | Hoc Sinh | 1 | 1 | 113 |
| 113 | Hoc Sinh | 2 | 2 | 113 |
| 113 | Hoc Sinh | 3 | 3 | 113 |
| 113 | Hoc Sinh | 4 | 4 | 113 |
| 113 | Hoc Sinh | 5 | 5 | 113 |
| 113 | Hoc Sinh | 6 | 6 | 113 |
| 113 | Hoc Sinh | 7 | 7 | 113 |
| 113 | Hoc Sinh | 8 | 8 | 113 |
| 113 | Hoc Sinh | 9 | 9 | 113 |
| 113 | Hoc Sinh | 10 | 10 | 113 |
| 113 | Hoc Sinh | 11 | 11 | 113 |
| 911 | Giang Vien | 12 | 11 | 911 |
+----+-----+----+-----+----+
12 rows in set (0.00 sec)
```

LEFT JOIN



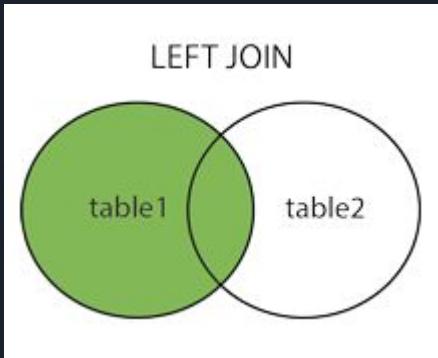
```
mysql> SELECT * FROM
-> role LEFT JOIN has_role
-> ON role.id = has_role.role_id;
+-----+-----+-----+-----+
| id   | role_name | id   | person_id | role_id |
+-----+-----+-----+-----+
| 113  | Hoc Sinh  | 1    | 1         | 113    |
| 113  | Hoc Sinh  | 2    | 2         | 113    |
| 113  | Hoc Sinh  | 3    | 3         | 113    |
| 113  | Hoc Sinh  | 4    | 4         | 113    |
| 113  | Hoc Sinh  | 5    | 5         | 113    |
| 113  | Hoc Sinh  | 6    | 6         | 113    |
| 113  | Hoc Sinh  | 7    | 7         | 113    |
| 113  | Hoc Sinh  | 8    | 8         | 113    |
| 113  | Hoc Sinh  | 9    | 9         | 113    |
| 113  | Hoc Sinh  | 10   | 10        | 113    |
| 113  | Hoc Sinh  | 11   | 11        | 113    |
| 911  | Giang Vien | 12   | 11        | 911    |
| 912  | Tro Giang   | NULL | NULL      | NULL   |
| 2023 | Hieu Truong | NULL | NULL      | NULL   |
+-----+-----+-----+-----+
14 rows in set (0.00 sec)
```

RIGHT JOIN



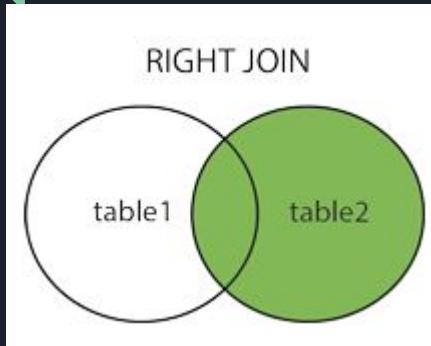
```
mysql> SELECT * FROM
-> role RIGHT JOIN has_role
-> ON role.id = has_role.role_id;
+-----+-----+-----+-----+
| id   | role_name | id | person_id | role_id |
+-----+-----+-----+-----+
| 113  | Hoc Sinh  | 1  |          1 |    113  |
| 113  | Hoc Sinh  | 2  |          2 |    113  |
| 113  | Hoc Sinh  | 3  |          3 |    113  |
| 113  | Hoc Sinh  | 4  |          4 |    113  |
| 113  | Hoc Sinh  | 5  |          5 |    113  |
| 113  | Hoc Sinh  | 6  |          6 |    113  |
| 113  | Hoc Sinh  | 7  |          7 |    113  |
| 113  | Hoc Sinh  | 8  |          8 |    113  |
| 113  | Hoc Sinh  | 9  |          9 |    113  |
| 113  | Hoc Sinh  | 10 |         10 |    113  |
| 113  | Hoc Sinh  | 11 |         11 |    113  |
| 911  | Giang Vien | 12 |         11 |    911  |
+-----+-----+-----+-----+
12 rows in set (0.00 sec)
```

LEFT JOIN



```
mysql> SELECT *  
-> FROM  
-> has_role LEFT JOIN role  
-> ON role.id = has_role.role_id;  
+----+-----+-----+----+-----+  
| id | person_id | role_id | id | role_name |  
+----+-----+-----+----+-----+  
| 1 | | 113 | 113 | Hoc Sinh |  
| 2 | | 113 | 113 | Hoc Sinh |  
| 3 | | 113 | 113 | Hoc Sinh |  
| 4 | | 113 | 113 | Hoc Sinh |  
| 5 | | 113 | 113 | Hoc Sinh |  
| 6 | | 113 | 113 | Hoc Sinh |  
| 7 | | 113 | 113 | Hoc Sinh |  
| 8 | | 113 | 113 | Hoc Sinh |  
| 9 | | 113 | 113 | Hoc Sinh |  
| 10 | | 113 | 113 | Hoc Sinh |  
| 11 | | 113 | 113 | Hoc Sinh |  
| 12 | | 911 | 911 | Giang Vien |  
+----+-----+-----+----+-----+  
12 rows in set (0.00 sec)
```

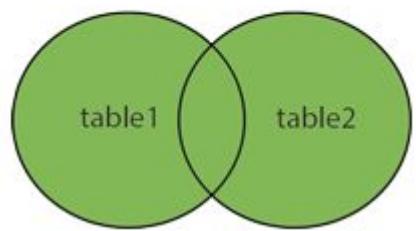
RIGHT JOIN



```
mysql> SELECT *
-> FROM
-> has_role RIGHT JOIN role
-> ON role.id = has_role.role_id;
+----+-----+-----+----+-----+
| id | person_id | role_id | id | role_name |
+----+-----+-----+----+-----+
| 1 | 1 | 113 | 113 | Hoc Sinh |
| 2 | 2 | 113 | 113 | Hoc Sinh |
| 3 | 3 | 113 | 113 | Hoc Sinh |
| 4 | 4 | 113 | 113 | Hoc Sinh |
| 5 | 5 | 113 | 113 | Hoc Sinh |
| 6 | 6 | 113 | 113 | Hoc Sinh |
| 7 | 7 | 113 | 113 | Hoc Sinh |
| 8 | 8 | 113 | 113 | Hoc Sinh |
| 9 | 9 | 113 | 113 | Hoc Sinh |
| 10 | 10 | 113 | 113 | Hoc Sinh |
| 11 | 11 | 113 | 113 | Hoc Sinh |
| 12 | 11 | 911 | 911 | Giang Vien |
| NULL | NULL | NULL | 912 | Tro Giang |
| NULL | NULL | NULL | 2023 | Hieu Truong |
+----+-----+-----+----+-----+
14 rows in set (0.00 sec)
```

FULL (OUTER) JOIN

FULL OUTER JOIN

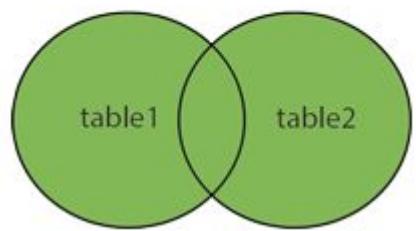


```
mysql> SELECT role_name, person_id, role_id
    -> FROM role FULL JOIN has_role;
+-----+-----+-----+
| role_name | person_id | role_id |
+-----+-----+-----+
| Hoc Sinh | 1 | 113 |
| Giang Vien | 1 | 113 |
| Tro Giang | 1 | 113 |
| Hieu Truong | 1 | 113 |
| Hoc Sinh | 2 | 113 |
| Giang Vien | 2 | 113 |
| Tro Giang | 2 | 113 |
| Hieu Truong | 2 | 113 |
| Hoc Sinh | 3 | 113 |
| Giang Vien | 3 | 113 |
| Tro Giang | 3 | 113 |
| Hieu Truong | 3 | 113 |
| Hoc Sinh | 4 | 113 |
| Giang Vien | 4 | 113 |
| Tro Giang | 4 | 113 |
| Hieu Truong | 4 | 113 |
| Hoc Sinh | 5 | 113 |
| Giang Vien | 5 | 113 |
| Tro Giang | 5 | 113 |
| Hieu Truong | 5 | 113 |
| Hoc Sinh | 6 | 113 |
| Giang Vien | 6 | 113 |
| Tro Giang | 6 | 113 |
| Hieu Truong | 6 | 113 |
| Hoc Sinh | 7 | 113 |
| Giang Vien | 7 | 113 |
| Tro Giang | 7 | 113 |
| Hieu Truong | 7 | 113 |
| Hoc Sinh | 8 | 113 |
| Giang Vien | 8 | 113 |
| Tro Giang | 8 | 113 |
| Hieu Truong | 8 | 113 |
| Hoc Sinh | 9 | 113 |
| Giang Vien | 9 | 113 |
| Tro Giang | 9 | 113 |
| Hieu Truong | 9 | 113 |
| Hoc Sinh | 10 | 113 |
| Giang Vien | 10 | 113 |
| Tro Giang | 10 | 113 |
| Hieu Truong | 10 | 113 |
| Hoc Sinh | 11 | 113 |
| Giang Vien | 11 | 113 |
| Tro Giang | 11 | 113 |
| Hieu Truong | 11 | 113 |
| Hoc Sinh | 11 | 911 |
| Giang Vien | 11 | 911 |
| Tro Giang | 11 | 911 |
| Hieu Truong | 11 | 911 |
+-----+-----+-----+
```

```
Hieu Truong | 4 | 113 |
Hoc Sinh | 5 | 113 |
Giang Vien | 5 | 113 |
Tro Giang | 5 | 113 |
Hieu Truong | 5 | 113 |
Hoc Sinh | 6 | 113 |
Giang Vien | 6 | 113 |
Tro Giang | 6 | 113 |
Hieu Truong | 6 | 113 |
Hoc Sinh | 7 | 113 |
Giang Vien | 7 | 113 |
Tro Giang | 7 | 113 |
Hieu Truong | 7 | 113 |
Hoc Sinh | 8 | 113 |
Giang Vien | 8 | 113 |
Tro Giang | 8 | 113 |
Hieu Truong | 8 | 113 |
Hoc Sinh | 9 | 113 |
Giang Vien | 9 | 113 |
Tro Giang | 9 | 113 |
Hieu Truong | 9 | 113 |
Hoc Sinh | 10 | 113 |
Giang Vien | 10 | 113 |
Tro Giang | 10 | 113 |
Hieu Truong | 10 | 113 |
Hoc Sinh | 11 | 113 |
Giang Vien | 11 | 113 |
Tro Giang | 11 | 113 |
Hieu Truong | 11 | 113 |
Hoc Sinh | 11 | 911 |
Giang Vien | 11 | 911 |
Tro Giang | 11 | 911 |
Hieu Truong | 11 | 911 |
+-----+-----+-----+
48 rows in set (0.00 sec)
```

FULL (OUTER) JOIN

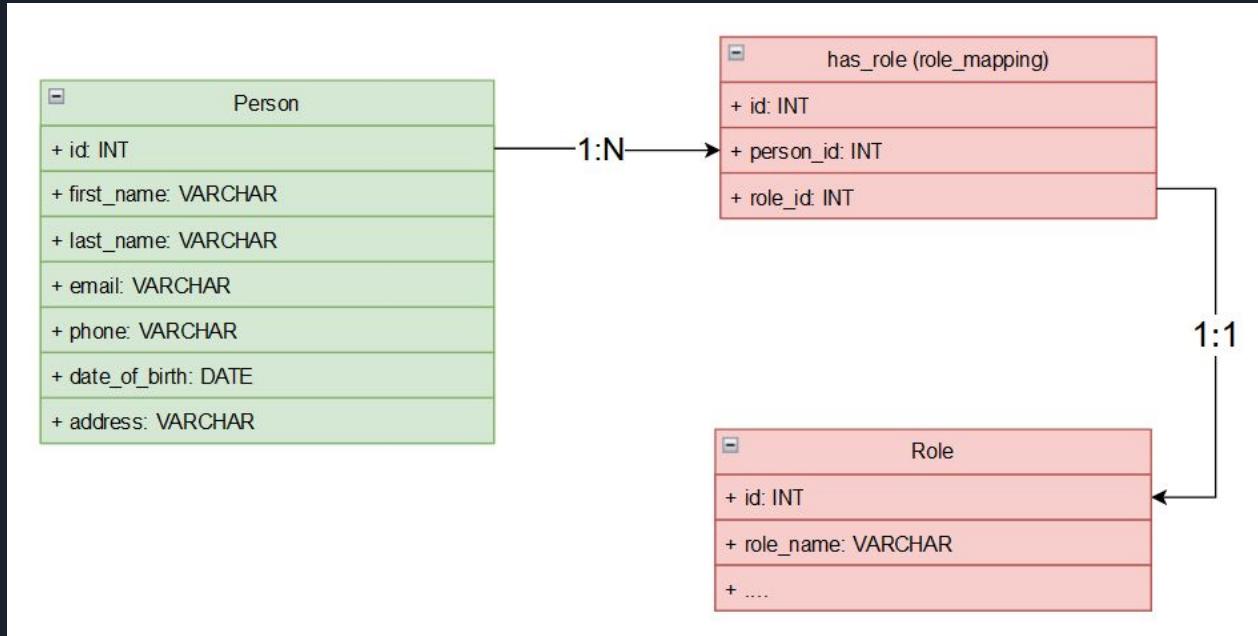
FULL OUTER JOIN



```
mysql> SELECT role_name, person_id, role_id
    -> FROM role FULL JOIN has_role;
+-----+-----+-----+
| role_name | person_id | role_id |
+-----+-----+-----+
| Hoc Sinh | 1 | 113 |
| Giang Vien | 1 | 113 |
| Tro Giang | 1 | 113 |
| Hieu Truong | 1 | 113 |
| Hoc Sinh | 2 | 113 |
| Giang Vien | 2 | 113 |
| Tro Giang | 2 | 113 |
| Hieu Truong | 2 | 113 |
| Hoc Sinh | 3 | 113 |
| Giang Vien | 3 | 113 |
| Tro Giang | 3 | 113 |
| Hieu Truong | 3 | 113 |
| Hoc Sinh | 4 | 113 |
| Giang Vien | 4 | 113 |
| Tro Giang | 4 | 113 |
| Hieu Truong | 4 | 113 |
| Hoc Sinh | 5 | 113 |
| Giang Vien | 5 | 113 |
| Tro Giang | 5 | 113 |
| Hieu Truong | 5 | 113 |
| Hoc Sinh | 6 | 113 |
| Giang Vien | 6 | 113 |
| Tro Giang | 6 | 113 |
| Hieu Truong | 6 | 113 |
| Hoc Sinh | 7 | 113 |
| Giang Vien | 7 | 113 |
| Tro Giang | 7 | 113 |
| Hieu Truong | 7 | 113 |
| Hoc Sinh | 8 | 113 |
| Giang Vien | 8 | 113 |
| Tro Giang | 8 | 113 |
| Hieu Truong | 8 | 113 |
| Hoc Sinh | 9 | 113 |
| Giang Vien | 9 | 113 |
| Tro Giang | 9 | 113 |
| Hieu Truong | 9 | 113 |
| Hoc Sinh | 10 | 113 |
| Giang Vien | 10 | 113 |
| Tro Giang | 10 | 113 |
| Hieu Truong | 10 | 113 |
| Hoc Sinh | 11 | 113 |
| Giang Vien | 11 | 113 |
| Tro Giang | 11 | 113 |
| Hieu Truong | 11 | 113 |
| Hoc Sinh | 11 | 911 |
| Giang Vien | 11 | 911 |
| Tro Giang | 11 | 911 |
| Hieu Truong | 11 | 911 |
+-----+-----+-----+
48 rows in set (0.00 sec)
```

JOIN

Lấy danh sách tất cả các bạn sinh viên



JOIN

Lấy danh sách tất cả mọi người trong hệ thống kèm với chức danh

JOIN & WHERE

Lấy danh sách tất cả các bạn Học Sinh

```
mysql> SELECT *
-> FROM
-> person JOIN has_role JOIN role
-> ON person.id = has_role.person_id AND role.id = has_role.role_id
-> WHERE role_name = 'Hoc Sinh';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | first_name | last_name | email | phone | date_of_birth | address |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Trung | Nguyen | trungnguyen@example.com | 0987654321 | 1990-01-01 | 123 Duong Le Loi, Quan 1 |
| 2 | Linh | Nguyen | linhnguyen@example.com | 0987654322 | 1995-05-05 | 456 Duong Nguyen Hue, Quan 3 |
| 3 | Hoa | Tran | hoatran@example.com | 0987654323 | 1985-08-15 | 789 Duong Vo Thi Sau, Quan 5 |
| 4 | Minh | Le | minhle@example.com | 0987654324 | 1992-02-20 | 321 Duong Cach Mang Thang Tam, Quan 10 |
| 5 | Phuong | Pham | phuongpham@example.com | 0987654325 | 1980-11-11 | 654 Duong Nam Ky Khoi, Quan 1 |
| 6 | An | Tran | antran@example.com | 0992654326 | 1993-07-15 | 456 Duong Tran Hung Dao, Quan 1 |
| 7 | Dat | Pham | datpham@example.com | 0987654327 | 1997-03-20 | 789 Duong Nam Ky Khoi Nghia, Quan 3 |
| 8 | Thu | Le | thule@example.com | 0987654328 | 1989-12-01 | 321 Duong Ly Thai To, Quan 10 |
| 9 | Tien | Vo | tienvo@example.com | 0963654329 | 1991-06-10 | 654 Duong Vo Van Tan, Quan 3 |
| 10 | Hien | Nguyen | hiennnguyen@example.com | 0963654330 | 1998-11-28 | 123 Duong Nguyen Hue, Quan 1 |
| 11 | Luan | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```



JOINS

- Web server
 - Query tất cả các bạn học sinh
 - SELECT id FROM role WHERE role_name = "Hoc Sinh"
 - SELECT person_id FROM has_role WHERE role_id = id
 - ORDER BY ID ASC
 - LIMIT 10
 - <1, 2, 3, 4, 5, 6>
 - SELECT first_name, last_name FROM person



JOINS

- Liệt kê tổng số sinh viên trên hệ thống
- Liệt kê danh sách số lượng người theo chức danh trong hệ thống
- Liệt kê các bạn sinh viên có tuổi lớn hơn 20
 - lớn hơn 30



Tổng quan

1. Giới thiệu chung về database
2. Các kiểu dữ liệu cở bản trong MySQL
3. Mô phỏng dữ liệu thực tế
4. Database table (bảng dữ liệu) là gì, cách tạo table
5. Truy vấn (Query) dữ liệu
6. Join table
7. Sub Query
8. Tối ưu câu lệnh
9. Link github phần sql



Sub Query

```
SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
WHERE column_name OPERATOR  
  (SELECT column_name [, column_name ]  
   FROM table1 [, table2 ]  
   [WHERE])
```

Sub Query

Lấy danh sách họ tên các bạn học sinh từ bảng person

```
mysql> SELECT first_name, last_name
    -> FROM person
    -> WHERE id IN (
    ->     SELECT person_id
    ->     FROM has_role JOIN role
    ->     ON has_role.role_id = role.id
    ->     WHERE role_name = 'Hoc Sinh'
    -> )
    -> ;
+-----+-----+
| first_name | last_name |
+-----+-----+
| Trung      | Nguyen   |
| Linh       | Nguyen   |
| Hoa        | Tran     |
| Minh       | Le        |
| Phuong     | Pham     |
| An         | Tran     |
| Dat        | Pham     |
| Thu         | Le        |
| Tien        | Vo        |
| Hien        | Nguyen   |
| Luan        | Phan     |
+-----+-----+
11 rows in set (0.00 sec)
```



Sub Query

Lấy danh sách họ tên các bạn học sinh từ bảng person chỉ sử dụng sub query



Tối ưu Query

- Dùng index
- Hạn chế sử dụng %
- Hạn chế sử dụng full outer join
- Hạn chế sử dụng “SELECT *”
- Hạn chế sử dụng DISTINCT, YEAR(Curr()) - YEAR(date_of_birth)



Index

Tại sao phải sử dụng index?

- Tăng tốc độ truy vấn

Create INDEX

```
1 CREATE INDEX phone_number_idx ON school.person (phone);
2 CREATE INDEX last_name_idx ON school.person (last_name);
3 CREATE INDEX email_idx ON school.person (email);
4 |
```



Indexed vs Not indexed

Khi data đủ lớn, ta sẽ thấy được sự khác biệt khi sử dụng index và không index

Tạo DB với 10 million records, query để kiểm tra performance

- git pull
- git checkout 10m

Indexed vs Not indexed

```
intro_db_index_10m_1 | 2023-05-10 06:38:32+00:00 [Note] [Entrypoint]: Temporary server started.
intro_db_10m_1       | 2023-05-10 06:38:32+00:00 [Note] [Entrypoint]: Temporary server started.
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Creating database example_database
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Creating user example_user
intro_db_10m_1        | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Creating database example_database
intro_db_10m_1         | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Creating user example_user
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Giving user example_user access to schema example_database
intro_db_10m_1          | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Giving user example_user access to schema example_database
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Securing system users (equivalent to running mysql_secure_installation)
intro_db_10m_1           | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: Securing system users (equivalent to running mysql_secure_installation)
intro_db_index_10m_1 |
intro_db_10m_1          |
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/0_database.sql
intro_db_10m_1            | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/0_database.sql
intro_db_index_10m_1 |
intro_db_index_10m_1 |
intro_db_index_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/1_person.sql
intro_db_10m_1             |
intro_db_10m_1             |
intro_db_10m_1 | 2023-05-10 06:38:34+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/1_person.sql
```

Indexed vs Not indexed

Query data không có trong table
Full scan vs scan with index

SQL command

```
SELECT *
FROM person
WHERE first_name = "luanphan1000"
```

No rows.
(0.003 s) Edit, Explain, Export

```
SELECT *
FROM person
WHERE first_name = "luanphan1000"
```

Execute Limit rows: Stop on error Show only errors
[History](#)

```
SELECT *
FROM person
WHERE first_name = "luanphan1000"
```

No rows.
(14.732 s) Edit, Explain, Export

```
SELECT *
FROM person
WHERE first_name = "luanphan1000"
```

Execute Limit rows: Stop on error Show only errors
[History](#)

Indexed vs Not indexed

Query data có trong table

Full scan vs scan with index

```
SELECT *
FROM person
WHERE first_name = "Luan1000"
```

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|--------------------------|--------------|----------------------|------------------|
| 1000 | Luan1000 | Phan | luanphan1000@exampl.come | 09631000 | 1990-01-01 | Some Where in VN |

1 row (13.106 s) Edit, Explain, Export

| id? | select_type? | table? | partitions? | type? | possible_keys? | key? | key_len? | ref? | rows? | Extra? |
|------------|---------------------|---------------|--------------------|--------------|-----------------------|-------------|-----------------|-------------|--------------|---------------|
| 1 | SIMPLE | person | NULL | ALL | NULL | NULL | NULL | NULL | 9979225 | Using where |

```
SELECT *
FROM person
WHERE first_name = "Luan1000"
```

Indexed vs Not indexed

Query data có trong table
Full scan vs scan with index

```
SELECT *
FROM person
WHERE first_name = "Luan1000"
```

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|--------------------------|--------------|----------------------|------------------|
| 1000 | Luan1000 | Phan | luanphan1000@exampl.come | 09631000 | 1990-01-01 | Some Where in VN |

1 row (0.003 s) [Edit](#), [Explain](#), [Export](#)

```
SELECT *
FROM person
WHERE first_name = "Luan1000"
```



How does database index?

Simple DB with 2 functions

Key value

Set key => value, Get key => value

```
1: db.sh
 1 #!/bin/bash
 2
 3 function set() {
 4     echo "$1,$2" >> db.txt
 5 }
 6
 7 function get() {
 8     grep "$1" db.txt | tail -n1 | cut -d ',' -f2
 9 }
10
11 $"@
```

buffers

Simple DB

```
Luanphan@LUAN-PC:~/dev/intro-database$ ./db.sh set dog 1000
Luanphan@LUAN-PC:~/dev/intro-database$ ./db.sh set dog 1999
Luanphan@LUAN-PC:~/dev/intro-database$ ./db.sh set cat 2000
Luanphan@LUAN-PC:~/dev/intro-database$ ./db.sh get dog
1999
Luanphan@LUAN-PC:~/dev/intro-database$ ./db.sh get cat
2000
Luanphan@LUAN-PC:~/dev/intro-database$ █
```



Simple DB

- Insert nhanh, append only file

Nhưng khi database size growth (10 million records)

- Get
 - Get key exist -> scan full file (DB without index)
 - Get key not exist -> scan full file (DB without index)
- Duplicate data
 - Khi set append data only -> duplicate
 - File dữ liệu sẽ lớn dần theo thời gian

Simple DB compaction

Giải quyết vấn đề file dữ liệu lớn dần theo thời gian

Data file segment

| | | | | | |
|------------|------------|------------|-----------|------------|-----------|
| mew: 1078 | purr: 2103 | purr: 2104 | mew: 1079 | mew: 1080 | mew: 1081 |
| purr: 2105 | purr: 2106 | purr: 2107 | yawn: 511 | purr: 2108 | mew: 1082 |



Compaction process

Compacted segment

| | | |
|-----------|-----------|------------|
| yawn: 511 | mew: 1082 | purr: 2108 |
|-----------|-----------|------------|

Hash index

- Tạo một hashmap
 - Lưu data xuống file
 - Update hashmap với key, value mới nhất

```
1 const fs = require('fs/promises');
2
3 let index = new Map();
4
5 async function set(key, val) {
6   let row = `${key},${val}\n`;
7   try {
8     await fs.appendFile('./db.txt', row);
9   } catch (err) {
10   console.log(err);
11 }
12 index.set(key, val);
13 }
14
15 function get(key) {
16   let val = index.get(key);
17   if (val == "") {
18     console.log("try to search on file");
19   }
20   console.log(val);
21 }
22
23
24 get("dog");
25 set("dog", "hasmapset").then(() => {
26   get("dog");
27 })
```

Hash index

- Tạo một hashmap
 - Lưu data xuống file
 - Update hashmap với key, value mới nhất

```
Luanphan@LUAN-PC:~/dev/intro-database$ node db.js
undefined
hasmapset
Luanphan@LUAN-PC:~/dev/intro-database$ cat db.txt
dog,hasmapset
Luanphan@LUAN-PC:~/dev/intro-database$ █
```



Hash index

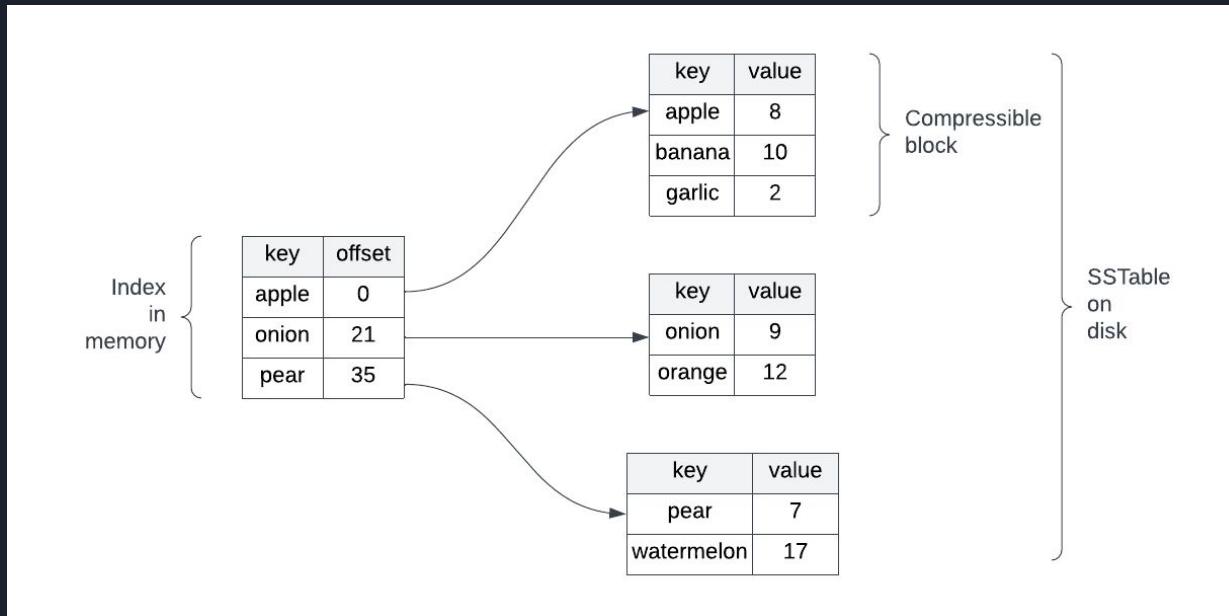
- Search by key mà không phải quét hết file (có index trên memory)

Nhược điểm:

- DB crash -> in memory hash table will be gone
 - Scan all DB to get build has table again
- Không thể thực hiện range query
 - Key được lưu trên hashmap -> random -> scan all table
- Data ko có trong hashmap
 - Tìm data không có trong table -> scan toàn bộ file
- Dữ liệu lớn -> gặp vấn đề khi memory không đủ

SSTable (Sorted string table)

- Cũng in-memory structure
- Các key được store trong in memory index phải được **SORT**
 - AVL, Red-black Tree (Balanced search tree) -> memtable





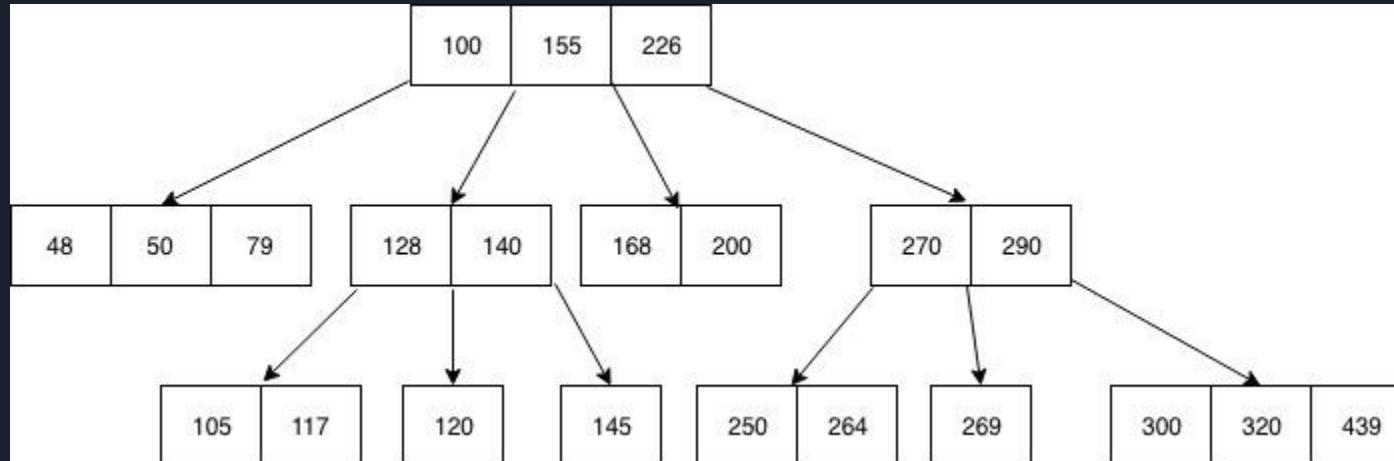
SSTable

- Write throughput high
- Read throughput depend

BTree

On disk index

- MySQL InnoDB Primary được index (clustered index)
- Tương tự binary tree
- Ổ cứng được chia thành các trang (page) với một size cố định (thường là 16kb)
- Parent page chia theo range
 - Leaf page sẽ lưu dữ liệu



Clustered index

- Tất cả các cột trong một dòng sẽ được lưu chung với primary key trên disk

```
SHOW INDEX FROM person
```

| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Ignored |
|--------|------------|------------------|--------------|-------------|-----------|-------------|----------|--------|------|------------|---------|---------------|---------|
| person | 0 | PRIMARY | 1 | id | A | 9949297 | NULL | NULL | | BTREE | | | NO |
| person | 0 | phone | 1 | phone | A | 9949297 | NULL | NULL | YES | BTREE | | | NO |
| person | 1 | phone_number_idx | 1 | phone | A | 9949297 | NULL | NULL | YES | BTREE | | | NO |
| person | 1 | last_name_idx | 1 | last_name | A | 2 | NULL | NULL | YES | BTREE | | | NO |
| person | 1 | first_name_idx | 1 | first_name | A | 9949297 | NULL | NULL | YES | BTREE | | | NO |
| person | 1 | email | 1 | email | A | 9949297 | NULL | NULL | YES | BTREE | | | NO |

6 rows (0.001 s) [Edit](#), [Export](#)

```
SHOW INDEX FROM person;
```

Secondary index

- Chỉ lưu index, dữ liệu của leaf node sẽ trả về primary index

```
SELECT database_name, table_name, index_name,
ROUND(stat_value * @@innodb_page_size / 1024 / 1024, 2) size_in_mb
FROM mysql.innodb_index_stats
WHERE stat_name = 'size'
ORDER BY size_in_mb DESC
```

| database_name | table_name | index_name | size_in_mb |
|---------------|----------------|------------------|------------|
| school | person | PRIMARY | 1155.00 |
| school | person | email | 423.98 |
| school | person | phone | 414.95 |
| school | person | first_name_idx | 246.88 |
| school | person | phone_number_idx | 246.88 |
| school | person | last_name_idx | 170.70 |
| mysql | gtid_slave_pos | PRIMARY | 0.02 |
| school | role | PRIMARY | 0.02 |
| school | class | PRIMARY | 0.02 |
| school | has_role | PRIMARY | 0.02 |

10 rows (0.000 s) [Edit](#), [Explain](#), [Export](#)

```
SELECT database_name, table_name, index_name,
ROUND(stat_value * @@innodb_page_size / 1024 / 1024, 2) size_in_mb
FROM mysql.innodb_index_stats
WHERE stat_name = 'size'
ORDER BY size_in_mb DESC;
```

```
SELECT
    table_name AS `Table`,
    round(((data_length + index_length) / 1024 / 1024), 2) `Size in MB`
FROM information_schema.TABLES
WHERE table_schema = "school"
    AND table_name = "person"
```

| Table | Size in MB |
|--------|------------|
| person | 2658.39 |

1 row (0.001 s) [Edit](#), [Explain](#), [Export](#)

```
SELECT
    table_name AS `Table`,
    round(((data_length + index_length) / 1024 / 1024), 2) `Size in MB`
FROM information_schema.TABLES
WHERE table_schema = "school"
    AND table_name = "person";
```

Index

Tối ưu câu query sau đây

```
SELECT MAX(date_of_birth) FROM person LIMIT 20;
```

```
CREATE INDEX date_of_birth_idx ON school.person (date_of_birth);
```

SQL command

```
SELECT MAX(date_of_birth)
FROM person
LIMIT 20

MAX(date_of_birth)
1990-01-01

1 row (12.331 s) Edit, Explain, Export
```

```
SELECT MAX(date_of_birth)
FROM person
LIMIT 20;
```



Tối ưu Query

- Dùng index
- Hạn chế sử dụng %
- Hạn chế sử dụng full outer join
- Hạn chế sử dụng “SELECT *”
 - Chỉ query những cột cần dùng
 - Network throughput và latency
- Hạn chế sử dụng DISTINCT, YEAR(NOW())- YEAR(date_of_birth)...
 - Client có thể tự tính, không cần thiết tốn database resource

Hạn chế dùng %

Nếu dùng thì nên dùng ở dưới cuối

```
SELECT *
FROM person
WHERE first_name = 'Luan1000'
```

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|--------------------------|--------------|----------------------|------------------|
| 1000 | Luan1000 | Phan | luanphan1000@exampl.come | 09631000 | 1990-01-01 | Some Where in VN |

1 row (0.001 s) [Edit](#), [Explain](#), [Export](#)

| id? | select_type? | table? | partitions? | type? | possible_keys? | key? | key_len? | ref? | rows? | Extra? |
|------------|---------------------|---------------|--------------------|--------------|-----------------------|----------------|-----------------|-------------|--------------|-----------------------|
| 1 | SIMPLE | person | NULL | ref | first_name_idx | first_name_idx | 203 | const | 1 | Using index condition |

```
SELECT COUNT(*)
FROM person
WHERE first_name LIKE "Luan1000%"
```

| COUNT(*) |
|----------|
| 1112 |

1 row (0.001 s) [Edit](#), [Explain](#), [Export](#)

| id? | select_type? | table? | partitions? | type? | possible_keys? | key? | key_len? | ref? | rows? | Extra? |
|------------|---------------------|---------------|--------------------|--------------|-----------------------|----------------|-----------------|-------------|--------------|--------------------------|
| 1 | SIMPLE | person | NULL | range | first_name_idx | first_name_idx | 203 | NULL | 1112 | Using where; Using index |

```
SELECT COUNT(*)
FROM person
WHERE first_name LIKE "%Luan1000%"
```

| COUNT(*) |
|----------|
| 1 |

1 row (3.955 s) [Edit](#), [Explain](#), [Export](#)

| id? | select_type? | table? | partitions? | type? | possible_keys? | key? | key_len? | ref? | rows? | Extra? |
|------------|---------------------|---------------|--------------------|--------------|-----------------------|----------------|-----------------|-------------|--------------|--------------------------|
| 1 | SIMPLE | person | NULL | index | NULL | first_name_idx | 203 | NULL | 9733025 | Using where; Using index |

Hạn chế join liên tục

- Có thể filter dữ liệu thông qua sub query, sau đó join dữ liệu với nhau
- Dùng inner join nếu có thể

```
SELECT first_name, last_name
FROM person JOIN has_role JOIN role
ON person.id = has_role.person_id
AND has_role.role_id = role.id
LIMIT 50000, 10
```

| first_name | last_name |
|-------------|-----------|
| Luan1017233 | Phan |
| Luan1017234 | Phan |
| Luan1017235 | Phan |
| Luan1017236 | Phan |
| Luan1017237 | Phan |
| Luan1017238 | Phan |
| Luan1017239 | Phan |
| Luan1017240 | Phan |
| Luan1017241 | Phan |
| Luan1017242 | Phan |

10 rows (0.054 s) [Edit](#), [Explain](#), [Export](#)

Hạn chế join liên tục

- Có thể filter dữ liệu thông qua sub query, sau đó join dữ liệu với nhau
- Dùng inner join nếu có thể

```
SELECT first_name, last_name
FROM person
JOIN (
    SELECT person_id
    FROM has_role
    WHERE role_id = (
        SELECT id FROM role WHERE role_name = 'Học Sinh'
    )
    LIMIT 50000, 10
) as student
ON person.id = student.person_id
```

| first_name | last_name |
|-------------|-----------|
| Luan1017233 | Phan |
| Luan1017234 | Phan |
| Luan1017235 | Phan |
| Luan1017236 | Phan |
| Luan1017237 | Phan |
| Luan1017238 | Phan |
| Luan1017239 | Phan |
| Luan1017240 | Phan |
| Luan1017241 | Phan |
| Luan1017242 | Phan |

10 rows (0.007 s) [Edit](#), [Explain](#), [Export](#)

Hạn chế sử dụng distinct

Language: English

MySQL » intro_db_index_10m » school » SQL command Logout

Adminer 4.8.1

DB: school

SQL command Import
Export Create table

select class
select has_role
select person
select role

SELECT COUNT(DISTINCT(first_name))
FROM person

COUNT(DISTINCT(first_name))
10000000

1 row (5.090 s) Edit, Explain, Export

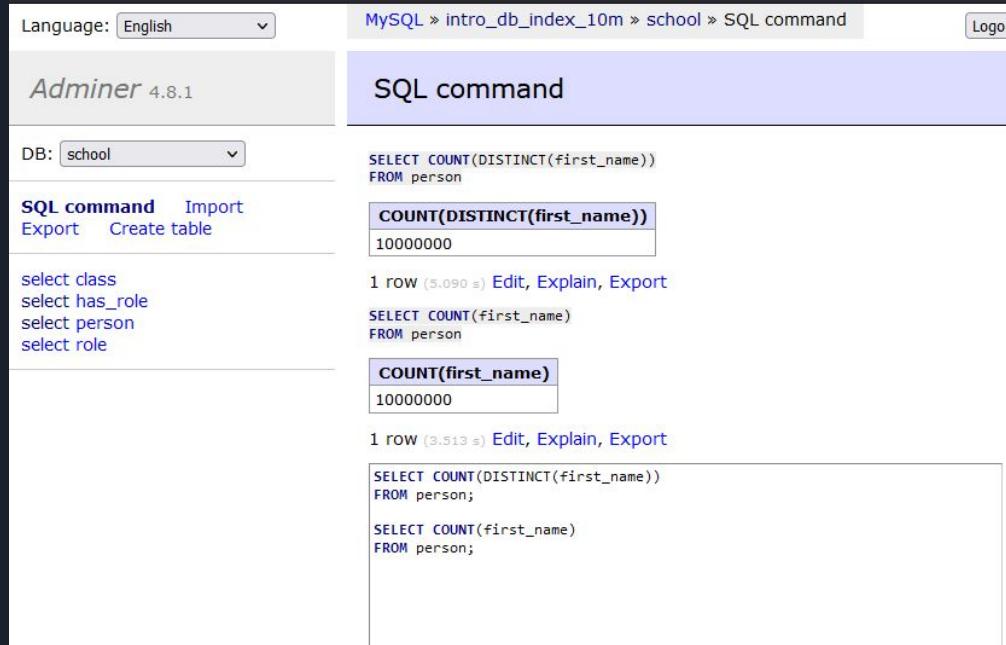
SELECT COUNT(first_name)
FROM person

COUNT(first_name)
10000000

1 row (3.513 s) Edit, Explain, Export

SELECT COUNT(DISTINCT(first_name))
FROM person;

SELECT COUNT(first_name)
FROM person;





Scale patterns

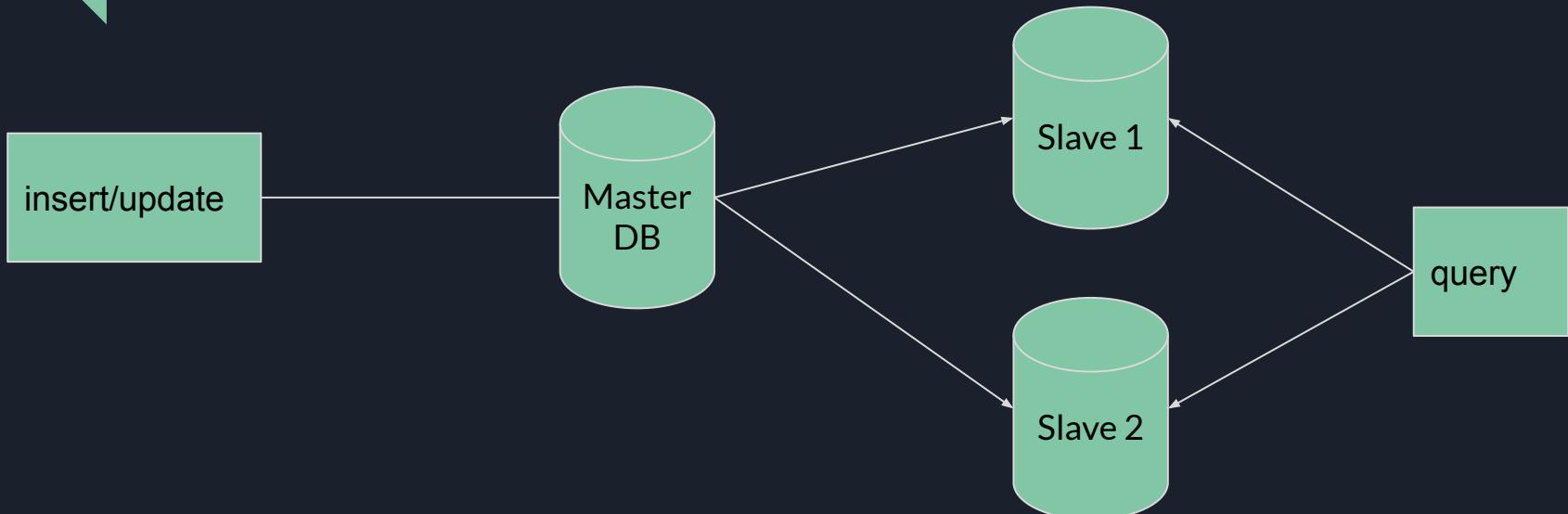
- Khi Database phải phục vụ truy vấn, lưu trữ dữ liệu ngày càng lớn theo thời gian, ta phải làm gì?



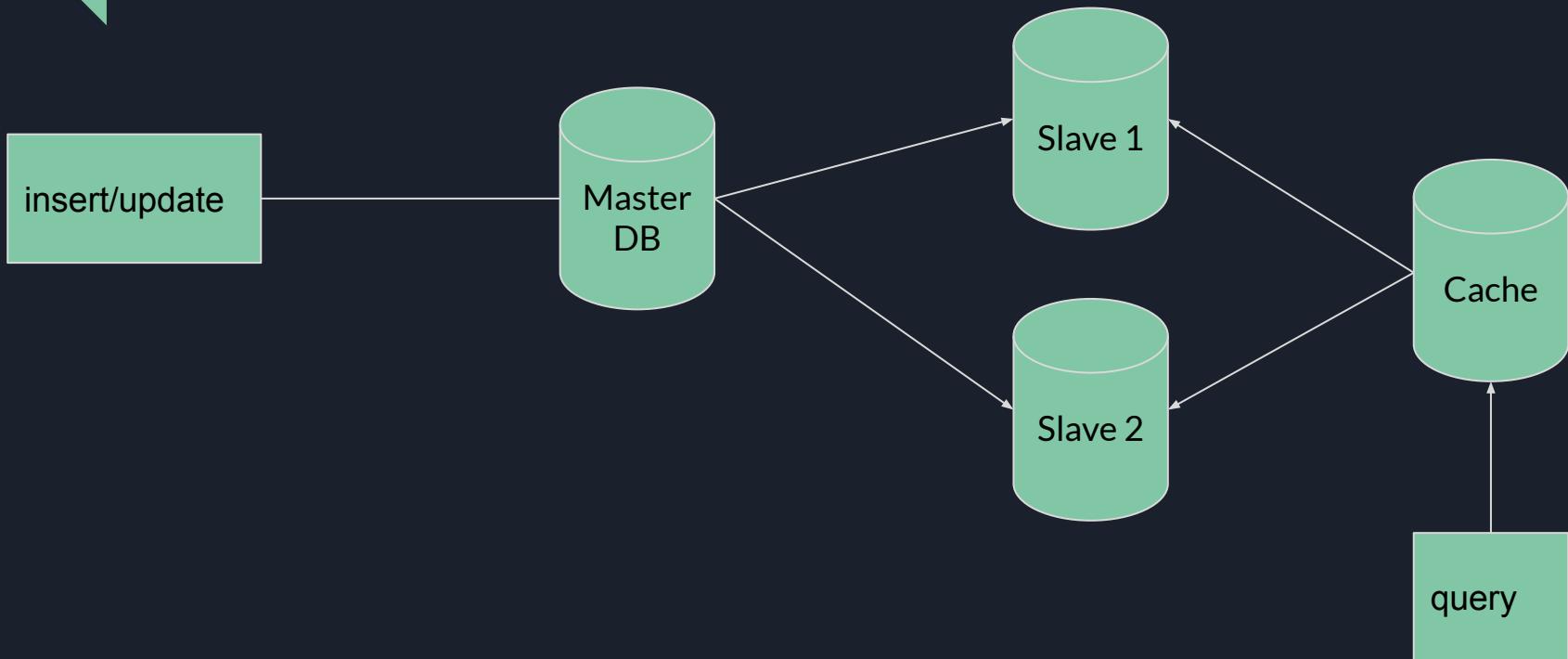
Scale patterns

- Vertical scaling DB
 - Increase disk space, memory
- Master/Slave (Write/Read) databases
 - Scaling multiple read (slave) DBs
- Cache layer (memache, redis cache)
- Sharding

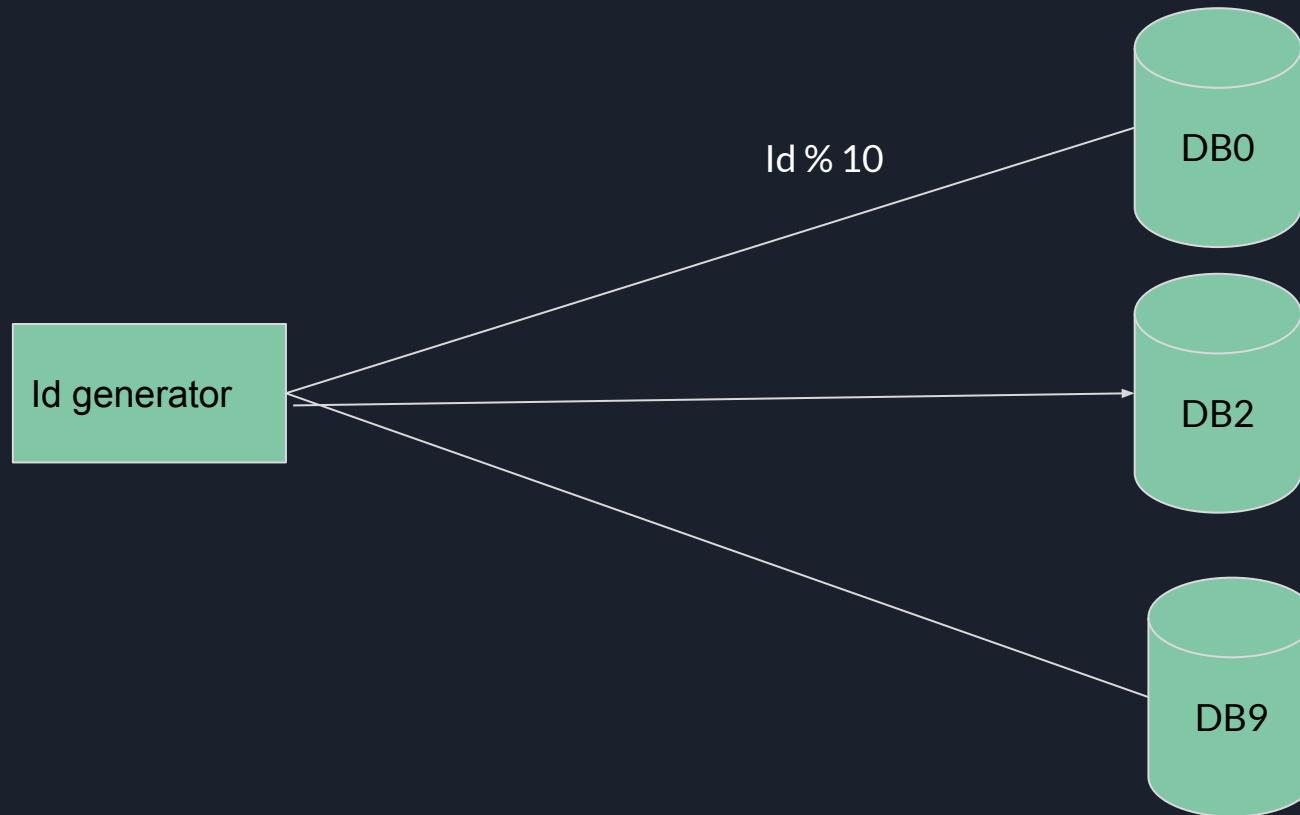
Master/Slave (Write/Read) DBs



Master/Slave (Write/Read) DBs



Sharding





Transaction

ACID

- Atomicity
- Consistency
- Isolation
- Durability



Transaction -> Atomicity

- Transaction là một hoặc một chuỗi các thao tác để thay đổi dữ liệu trong database
 - Tất cả thành công, mọi thay đổi được lưu vào database
 - Tất cả thất bại, mọi thay đổi được hủy bỏ

Transaction -> Atomicity

```
START TRANSACTION;
SELECT * FROM person WHERE email = 'luanphan@example.com';
UPDATE person SET first_name = 'Luan UPDATED' WHERE email = 'luanphan@example.com';
SELECT * FROM person WHERE email = 'luanphan@example.com';
ROLLBACK;

SELECT * FROM person WHERE email = 'luanphan@example.com';
```

Transaction -> Atomicity

SQL command

START TRANSACTION

Query executed OK, 0 rows affected. (0.000 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11 | Luan | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |

1 row (0.000 s) Edit, Explain, Export

UPDATE person SET first_name = 'Luan UPDATED' WHERE email = 'luanphan@example.com'

Query executed OK, 1 row affected. (0.000 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11 | Luan UPDATED | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |

1 row (0.000 s) Edit, Explain, Export

ROLLBACK

Query executed OK, 0 rows affected. (0.001 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'

| id | first_name | last_name | email | phone | date_of_birth | address |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11 | Luan | Phan | luanphan@example.com | 0388654331 | 1980-11-28 | 123 Duong Van Qua, Quan 1 |

Transaction -> Atomicity

```
START TRANSACTION;
SELECT * FROM person WHERE email = 'luanphan@example.com';
UPDATE person SET first_name = 'Luan UPDATED' WHERE email = 'luanphan@example.com';
SELECT * FROM person WHERE email = 'luanphan@example.com';
COMMIT;

SELECT * FROM person WHERE email = 'luanphan@example.com';
```

Transaction -> Atomicity

```
START TRANSACTION

Query executed OK, 0 rows affected. (0.000 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'



| <b>id</b> | <b>first_name</b> | <b>last_name</b> | <b>email</b>         | <b>phone</b> | <b>date_of_birth</b> | <b>address</b>            |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11        | Luan              | Phan             | luanphan@example.com | 0388654331   | 1980-11-28           | 123 Duong Van Qua, Quan 1 |


1 row (0.000 s) Edit, Explain, Export

UPDATE person SET first_name = 'Luan UPDATED' WHERE email = 'luanphan@example.com'

Query executed OK, 1 row affected. (0.000 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'



| <b>id</b> | <b>first_name</b> | <b>last_name</b> | <b>email</b>         | <b>phone</b> | <b>date_of_birth</b> | <b>address</b>            |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11        | Luan UPDATED      | Phan             | luanphan@example.com | 0388654331   | 1980-11-28           | 123 Duong Van Qua, Quan 1 |


1 row (0.000 s) Edit, Explain, Export

COMMIT

Query executed OK, 0 rows affected. (0.001 s) Edit

SELECT * FROM person WHERE email = 'luanphan@example.com'



| <b>id</b> | <b>first_name</b> | <b>last_name</b> | <b>email</b>         | <b>phone</b> | <b>date_of_birth</b> | <b>address</b>            |
|-----------|-------------------|------------------|----------------------|--------------|----------------------|---------------------------|
| 11        | Luan UPDATED      | Phan             | luanphan@example.com | 0388654331   | 1980-11-28           | 123 Duong Van Qua, Quan 1 |


```



Transaction -> Consistency

- Sau khi một transaction thành công hay thất bại, database cần phải giữ được tính nhất quán của dữ liệu theo các điều kiện nhất định
 - Tính nhất quán của dữ liệu trong transaction thực ra là ở tầng application và do lập trình viên chịu trách nhiệm
- Primary Key, Foreign Key, UNIQUE KEY không thuộc phạm vi transaction, mà do database quản lý



Transaction -> Isolation

Khi các transaction được thực hiện

- Phải bảo đảm chúng đụng thực hiện song song, nhưng khi nhìn từ ngoài vào thì như được thực hiện tuần tự
- No dirty read, no dirty write
 - Transaction không nên đọc dữ liệu được ghi bởi uncommitted transaction khác (no dirty read)
 - Transaction chỉ có thể thay đổi dữ liệu chưa được thay đổi bởi transaction khác (no dirty write)

Transaction -> Isolation

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan      |
+-----+
1 row in set (0.00 sec)

mysql>
mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 1' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name      |
+-----+
| Luan UPDATED Transaction 1 |
+-----+
1 row in set (0.00 sec)

mysql>

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan      |
+-----+
1 row in set (0.00 sec)

mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 2' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name      |
+-----+
| Luan UPDATED Transaction 2 |
+-----+
1 row in set (0.00 sec)
```

Transaction -> Isolation

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan       |
+-----+
1 row in set (0.00 sec)

mysql>
mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 1' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name          |
+-----+
| Luan UPDATED Transaction 1 |
+-----+
1 row in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```



```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan       |
+-----+
1 row in set (0.00 sec)

mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 2' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (45.65 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql>
mysql> █
```

Transaction -> Isolation

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan      |
+-----+
1 row in set (0.00 sec)

mysql>
mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 1' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name      |
+-----+
| Luan UPDATED Transaction 1 |
+-----+
1 row in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name |
+-----+
| Luan      |
+-----+
1 row in set (0.00 sec)

mysql> UPDATE person SET first_name = 'Luan UPDATED Transaction 2' WHERE email = 'luanphan@example.com';
Query OK, 1 row affected (45.65 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql>
mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name      |
+-----+
| Luan UPDATED Transaction 2 |
+-----+
1 row in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT first_name FROM person WHERE email = 'luanphan@example.com';
+-----+
| first_name      |
+-----+
| Luan UPDATED Transaction 2 |
+-----+
1 row in set (0.00 sec)

mysql> ■
```



Transaction -> Durability

- Sau khi transaction commit hay rollback, nếu có các sự cố ngoài ý muốn như database down, mất nguồn điện, network issues..., Khi database khôi phục lại thì tất cả dữ liệu vẫn giữ nguyên trạng

WAL

INSERT

- > write to WAL (INSERT)
- > append only file -> nhanh
- > index (16KB)
- > page split -> 2 page

CRASH

- > parent page

CRASH

- > store record in file

START TRANSACTION

- > write to WAL

UPDATE

- > write to WAL
- > append only file -> nhanh

ROLLBACK, COMMIT

- > write to WAL

CRASH



Locking

10 concurrent requests

t1, t2, t3, t4, ..., t10

X COUNT

UPDATE x = val

T10 -> updated

Locking

READ x = 3 // t1, ... t10 = 4

Application x = x + 1 = 4

UPDATE x = 4

READ x = 4 // t2

Application x = x + 1 = 5

UPDATE x = 5

13

LOCKING

SELECT x FOR UPDATE;



Sum up

Database design -> Duplicate data
Tìm hiểu về index -> optimize query
ACID -> transaction



Resources

<https://github.com/luanphandinh/intro-database>

[Design data intensive application](#)