

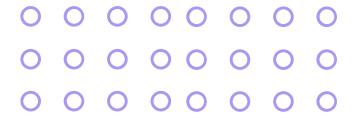


SQL & NOSQL

22127180 - Nguyễn Phúc Khang

22127419 - Nguyễn Minh Toàn



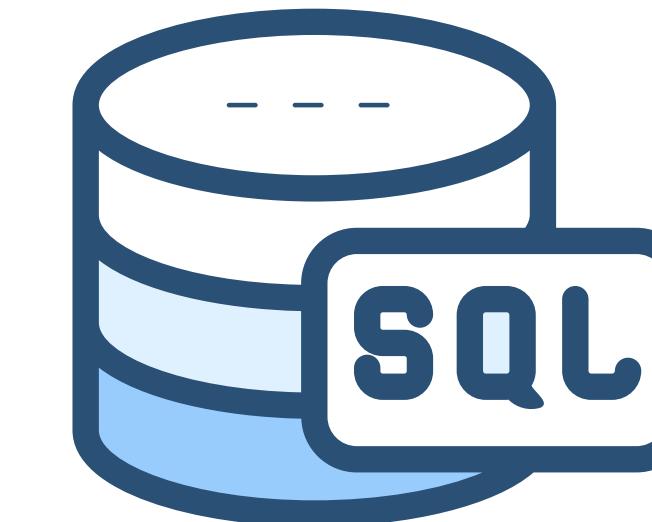


NỘI DUNG CHÍNH

- 01 GIỚI THIỆU**
- 02 SO SÁNH ĐẶC ĐIỂM CHÍNH GIỮA SQL VÀ NOSQL**
- 03 KHI NÀO NÊN SỬ DỤNG SQL VÀ NOSQL?**
- 04 VÍ DỤ THỰC TẾ MINH HỌA**
- 05 CÁC HỆ THỐNG CƠ SỞ DỮ LIỆU PHỔ BIẾN**
- 06 CÂU CHUYỆN VỀ SQL VÀ NOSQL**



LỊCH SỬ PHÁT TRIỂN



1970

EDGAR F. CODD ĐỀ XUẤT MÔ HÌNH RDBM

1974

IBM PHÁT TRIỂN SEQUEL (SAU THÀNH SQL)

1986

SQL TRỞ THÀNH TIÊU CHUẨN ANSI



NoSQL

2000

XUẤT HIỆN ĐỂ XỬ LÝ DỮ LIỆU LỚN

2006

GOOGLE RA MẮT BIGTABLE

2007

AMAZON PHÁT TRIỂN DYNAMODB

2009

THUẬT NGỮ "NOSQL" ĐƯỢC PHỔ BIẾN

SQL

- Mô hình quan hệ: Dữ liệu được tổ chức thành bảng (hàng và cột)
- Schema cố định: Cấu trúc phải được định nghĩa trước
- Quy chuẩn hóa: Giảm thiểu dư thừa



NOSQL

- Mô hình phi quan hệ với nhiều dạng khác nhau
- Schema linh hoạt: Cho phép thay đổi cấu trúc dễ dàng



Nosql



```
        "_id": "5cf0029caff5056591b0ce7d",
        "firstname": "Jane",
        "lastname": "Wu",
        "address": {
            "street": "1 Circle Rd",
            "city": "Los Angeles",
            "state": "CA",
            "zip": "90404"
        }
        "hobbies": ["surfing", "coding"]
    }
```

DOCUMENT

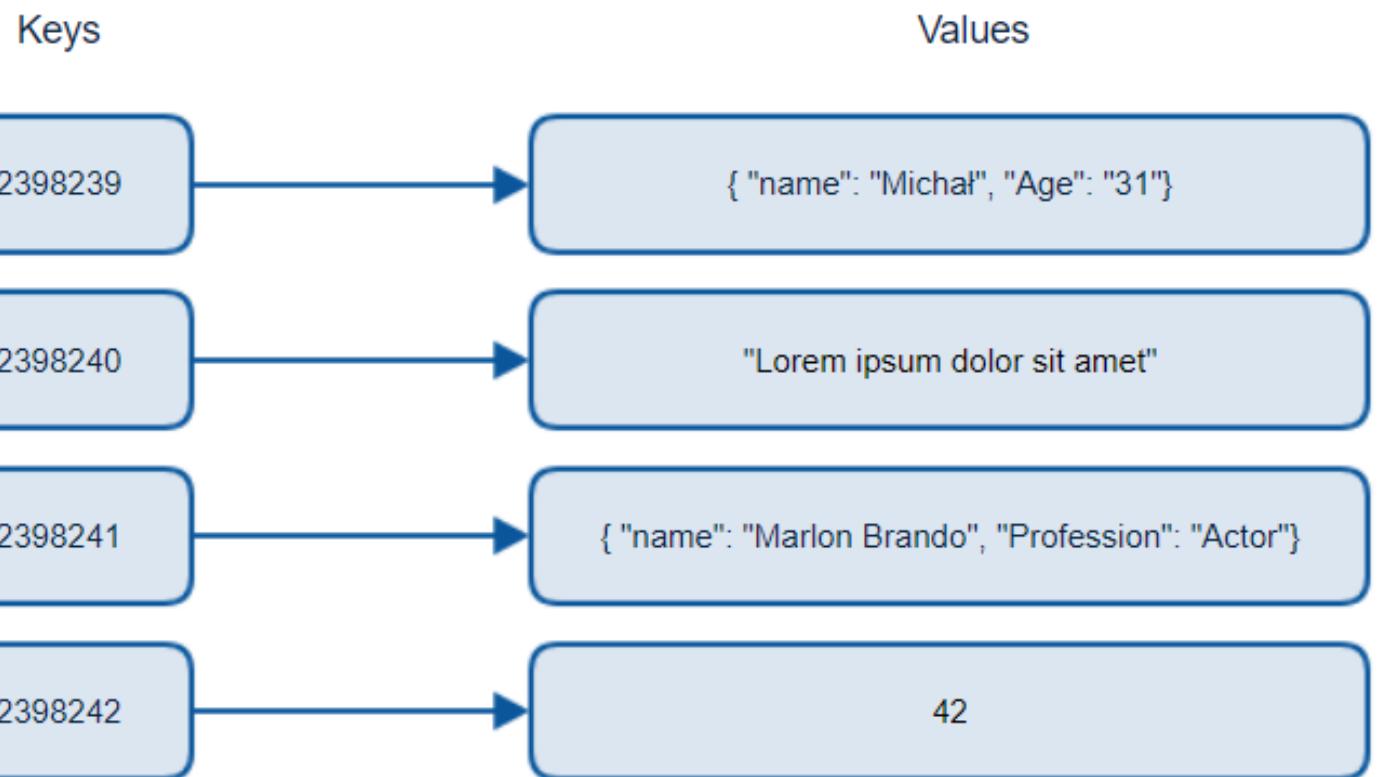
Column Family : Users

Keys	Columns						
Peter	<table border="1"><tr><td>Name</td><td>Number</td><td>Mobile Phone</td></tr><tr><td>Peter...</td><td>234786459</td><td>994398909</td></tr></table>	Name	Number	Mobile Phone	Peter...	234786459	994398909
Name	Number	Mobile Phone					
Peter...	234786459	994398909					
Joseph	<table border="1"><tr><td>Name</td><td>Number</td></tr><tr><td>Joseph</td><td>234786459</td></tr></table>	Name	Number	Joseph	234786459		
Name	Number						
Joseph	234786459						

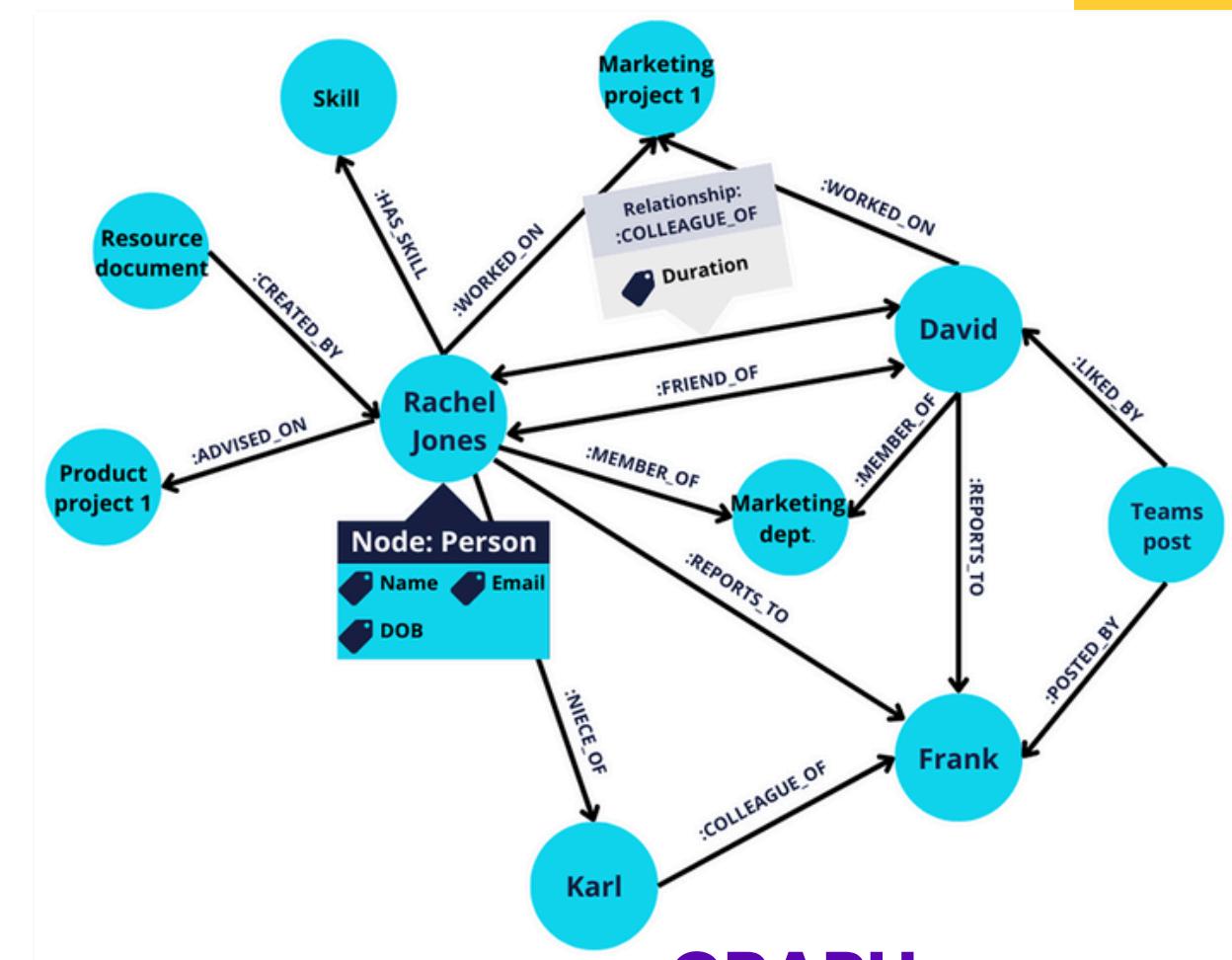
A 2x8 grid of circles, arranged in two rows of four circles each.

Super Column Family : Services

Keys	Super Columns		
Peter	Voice	Type Default	Balance 20
	SMS	Type Default	Amount 10
Joseph	Voice	Type Enterprise	Balance 60

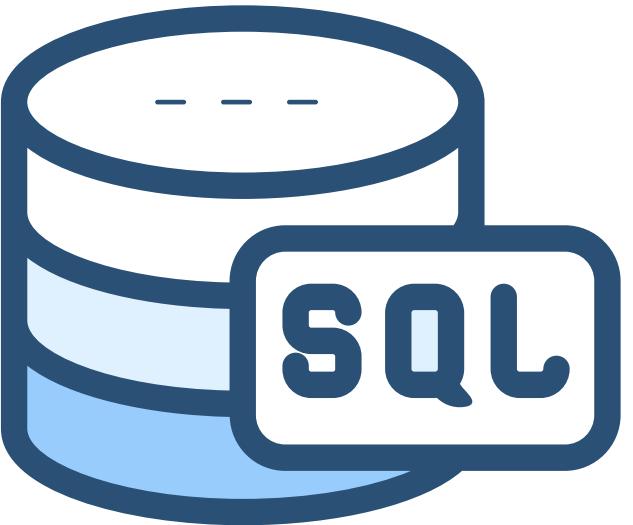


KEY - VALUE

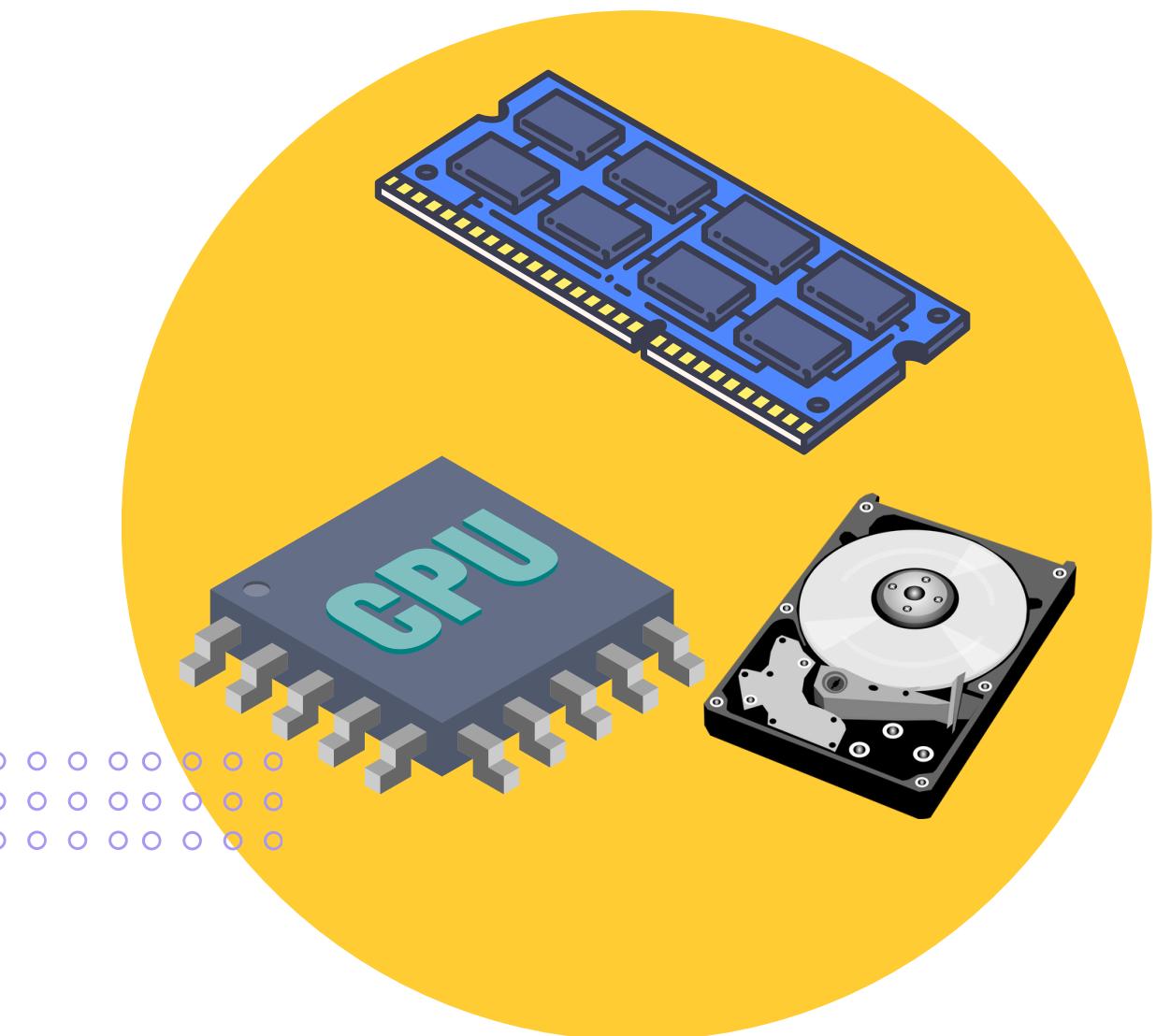


COLUMN-FAMILY

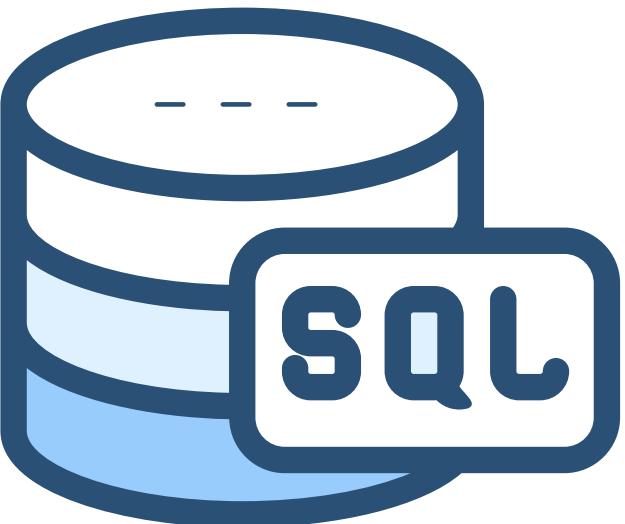
KHẢ NĂNG MỞ RỘNG



NoSQL



TÍNH NHẤT QUÁN



ACID

01 ATOMICITY

03 ISOLATION

02 CONSISTENCY

04 DURABILITY



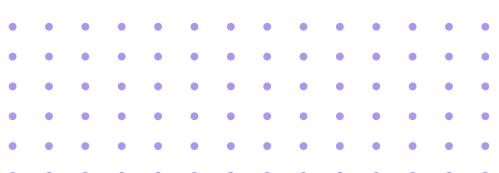
NoSQL

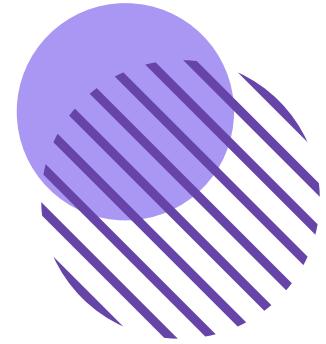
BASE

01 BASICALLY AVAILABLE

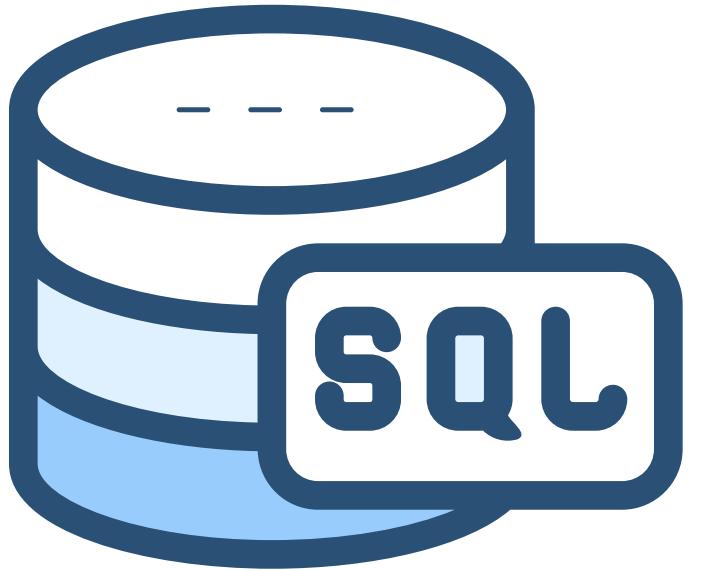
03 EVENTUALLY CONSISTENT

02 SOFT STATE



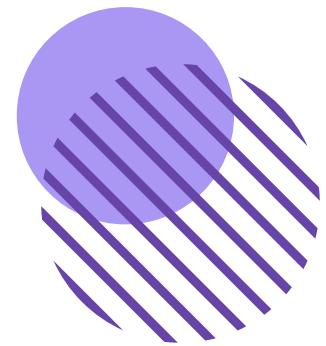


NGÔN NGỮ TRUY VẤN VÀ HIỆU SUẤT

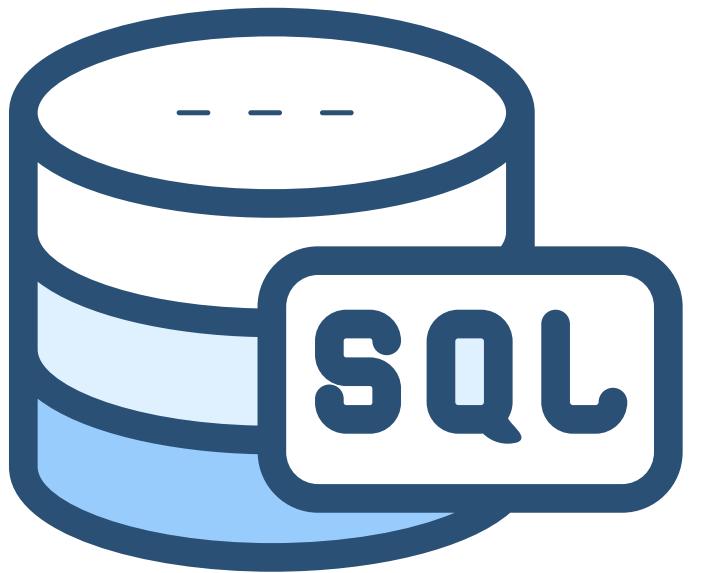


- Sử dụng ngôn ngữ SQL tiêu chuẩn
- Hiệu quả cho truy vấn phức tạp (JOIN)
- Hiệu suất giảm khi dữ liệu tăng

- Mỗi loại có ngôn ngữ/API riêng
- Tối ưu cho đọc/ghi nhanh
- Hiệu suất cao với dữ liệu lớn



TRƯỜNG HỢP SỬ DỤNG



- Hệ thống giao dịch (ngân hàng, thương mại điện tử)
- Ứng dụng cần truy vấn phức tạp
- Dữ liệu có cấu trúc rõ ràng và ít thay đổi
- Hệ thống yêu cầu tuân thủ ACID

- Dữ liệu lớn và phân tích
- Ứng dụng thời gian thực
- Phát triển nhanh và linh hoạt
- Mô hình dữ liệu đa dạng
- Khả năng mở rộng cao
- Lưu trữ và truy xuất dữ liệu đơn giản



VÍ DỤ MINH HỌA



SQL



```
1 -- Bắt đầu một giao dịch
2 BEGIN TRANSACTION;
3 -- Trừ 5 triệu từ tài khoản A
4 UPDATE accounts SET balance = balance - 5000000 WHERE account_id = 'A';
5 -- Cộng 5 triệu vào tài khoản B
6 UPDATE accounts SET balance = balance + 5000000 WHERE account_id = 'B';
7 -- Kết thúc giao dịch
8 -- Nếu có lỗi thì quay lại trạng thái ban đầu
9 IF @@ERROR <> 0 ROLLBACK; ELSE COMMIT;
10
```



NOSQL

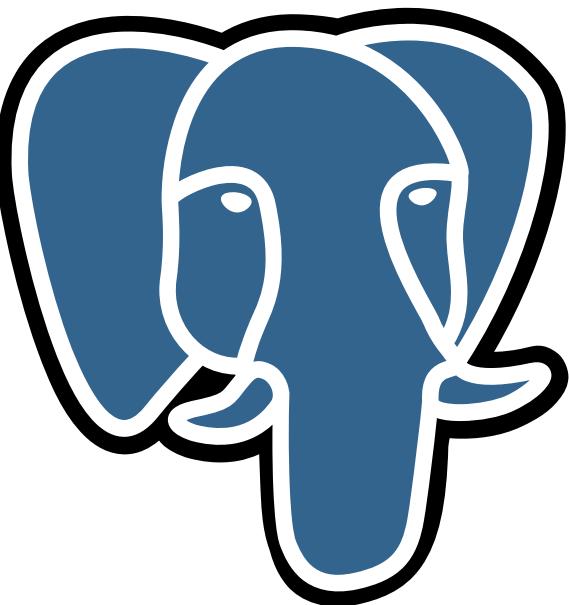


```
● ● ●  
1  {  
2    "post_id": "12345",  
3    "user": "John Doe",  
4    "content": "Hôm nay là một ngày đẹp trời!",  
5    "likes": 1000,  
6    "comments": [  
7      {"user": "Alice", "comment": "Tuyệt quá!", "timestamp": "2025-03-08T12:00:00"},  
8      {"user": "Bob", "comment": "Chúc bạn một ngày vui vẻ!", "timestamp": "2025-03-08T12:05:00"}  
9    ]  
10 }  
11
```



CÁC HỆ THỐNG CƠ SƠ DỮ LIỆU PHỔ BIẾN





POSTGRESQL



Ưu điểm

- Mã nguồn mở, nhiều tính năng nâng cao
- Hỗ trợ cả NoSQL (JSON, JSONB, HSTORE) vẫn đảm bảo tính toàn vẹn dữ liệu SQL.
- Xử lý dữ liệu phức tạp, nhiều truy vấn nặng (JOIN, GIS, AI/ML, Big Data).
- Quản lý đồng thời tốt, tránh deadlock.



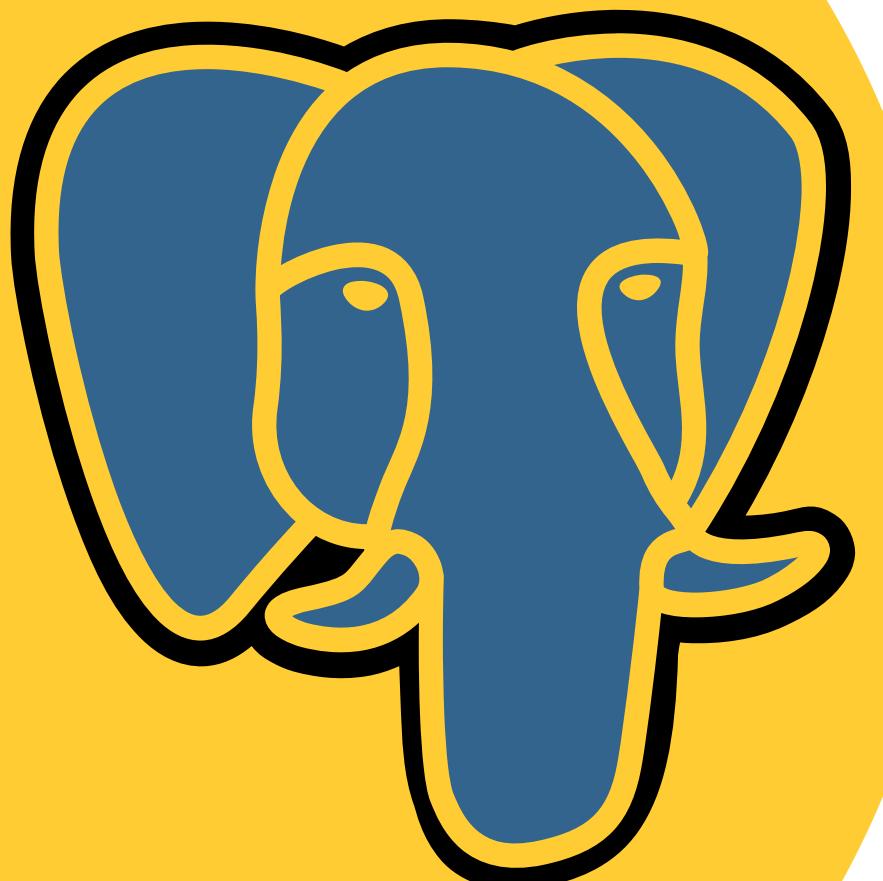
title	key	value
▶ PostgreSQL Tutorial	weight	11.2 ounces
PostgreSQL Tutorial	ISBN-13	978-1449370000
PostgreSQL Tutorial	languag	English
PostgreSQL Tutorial	paperba	243
PostgreSQL Tutorial	publishe	postgresqltutoria
PostgreSQL Cheat Sheet	weight	1 ounces
PostgreSQL Cheat Sheet	ISBN-13	978-1449370001
PostgreSQL Cheat Sheet	languag	English
PostgreSQL Cheat Sheet	paperba	5
PostgreSQL Cheat Sheet	publishe	postgresqltutoria

HSTORE

123 id	JSON tags
648,519,448	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
653,096,238	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
653,691,506	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
654,387,313	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
673,683,429	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "retail", "tags": "cafe, coffee shop, building, retail, Starbucks, amenity, name, cuisine, brand"}
681,679,646	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
683,445,691	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
688,522,615	{"name": "Starbucks", "brand": "Starbucks", "level": "1", "indoor": "room", "amenity": "cafe", "cuisine": "coffe", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
688,522,658	{"name": "Starbucks", "brand": "Starbucks", "level": "1", "indoor": "room", "amenity": "cafe", "cuisine": "coffe", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
700,704,143	{"name": "Starbucks", "brand": "Starbucks", "level": "1", "indoor": "room", "amenity": "cafe", "cuisine": "coffe", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
700,704,146	{"name": "Starbucks", "brand": "Starbucks", "level": "1", "indoor": "room", "amenity": "cafe", "cuisine": "coffe", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
733,353,075	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "smoking": "no", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}
736,989,448	{"name": "Starbucks", "brand": "Starbucks", "amenity": "cafe", "cuisine": "coffee_shop", "building": "yes", "tags": "cafe, coffee shop, building, yes, Starbucks, amenity, name, cuisine, brand"}

JSONB

POSTGRESQL



Nhược điểm

- Cấu hình phức tạp, cần hiểu sâu để tối ưu.
- Quản lý và bảo trì có thể khó hơn MySQL.
- Các truy vấn đơn giản → MySQL có thể nhanh hơn.

MySQL



Ưu điểm

- Dễ sử dụng, cài đặt đơn giản, phù hợp cho dự án nhỏ và vừa.
- Hiệu suất tốt cho truy vấn đơn giản, không quá phức tạp.
- Hỗ trợ replication dễ dàng, thích hợp cho hệ thống cần nhân bản dữ liệu.

MYSQL



Nhược điểm

- Xử lý truy vấn phức tạp kém hơn PostgreSQL.
- Hỗ trợ đồng thời kém hơn PosgresSQL do lock bảng nhiều.
- Ít linh hoạt trong indexing.

SQL SERVER

Ưu điểm

- Tích hợp chặt chẽ với hệ sinh thái Microsoft (Power BI, .NET, Azure).
- Công cụ quản lý mạnh, giao diện trực quan, dễ sử dụng.
- Hỗ trợ phân tích dữ liệu và Business Intelligence (BI) hiệu quả.



SQLSERVER

Nhược điểm

- Chi phí bản quyền cao, bản Enterprise ~\$7,128/core, yêu cầu mua tối thiểu 4 cores → Tổng chi phí tối thiểu khoảng \$28,512.
- Chạy tốt nhất trên hệ điều hành Windows, ít hỗ trợ Linux.



ORACLE



Ưu điểm

- Nhiều tính năng nâng cao
- Hiệu suất cao, tối ưu cho doanh nghiệp lớn.
- Bảo mật mạnh mẽ, phù hợp với hệ thống ngân hàng, tài chính.
- Hỗ trợ xử lý dữ liệu lớn, khả năng scaling tốt

ORACLE



Nhược điểm

- Chi phí rất cao, bản Enterprise Edition ~\$47,500/bộ xử lý và ~22% phí bảo trì hàng năm.
- Học và quản trị khó, yêu cầu người quản trị có trình độ cao.



NoSQL



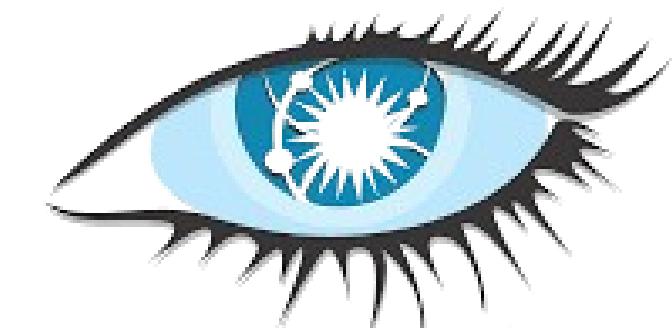
redis



neo4j



mongoDB®



cassandra



Firebase

MONGODB



mongoDB.

Ưu điểm:

- Linh hoạt, lưu trữ dữ liệu JSON-like, dễ mở rộng.
- Hỗ trợ truy vấn mạnh, indexing tốt.
- Phù hợp với dữ liệu không có cấu trúc cố định.
- Lưu trữ dữ liệu linh hoạt.

Nhược điểm:

- Không hỗ trợ giao dịch ACID mạnh như SQL.
- Không phù hợp cho hệ thống truy vấn phức tạp



CASSANDRA

Ưu điểm:

- Cơ sở dữ liệu dạng cột.
- Phân tán tốt, không có điểm hỏng đơn (high availability).
- Xử lý tốt dữ liệu lớn, tốc độ ghi cao.

Nhược điểm:

- Không hỗ trợ JOIN, khó truy vấn phức tạp.
- Ghi dữ liệu nhanh, nhưng đọc có thể chậm hơn so với SQL.



CASSANDRA



- 1 Column: id → [1, 2, 3]
- 2 Column: name → ["Alice", "Bob", "Eve"]
- 3 Column: age → [25, 30, 28]
- 4 Column: city → ["London", "New York", "Paris"]

REDIS



redis

Ưu điểm:

- Dạng key-value
- Tốc độ siêu nhanh (lưu trữ trên RAM).
- Hỗ trợ caching, message queue.

Nhược điểm:

- Không phù hợp lưu trữ dữ liệu lâu dài.
- Dữ liệu có thể mất nếu không cấu hình persistence.
- Hỗ trợ chính là tăng tốc hệ thống

NEO4J



Ưu điểm:

- Cơ sở dữ liệu dạng đồ thị.
- Tối ưu cho dữ liệu quan hệ phức tạp (mạng xã hội, đề xuất, phát hiện gian lận).
- Truy vấn quan hệ nhanh hơn so với SQL.

Nhược điểm:

- Không phù hợp với dữ liệu bảng truyền thống.
- Cần học query riêng (Cypher Query).



NEO4J



```
1 MATCH (alice:User {name: "Alice"})-[:FRIENDS_WITH]->(friend)  
2 RETURN friend.name
```



```
1 MATCH (alice:User {name: "Alice"})-[:FRIENDS_WITH]->(:User)-[:FRIENDS_WITH]->(fof)  
2 WHERE fof.name <> "Alice"  
3 RETURN DISTINCT fof.name
```

FIREBASE



Firebase

Ưu điểm:

- Đồng bộ dữ liệu real-time, hỗ trợ offline.
- Phù hợp ứng dụng mobile, web.
- Tốt hơn MongoDB nếu chỉ cần đồng bộ dữ liệu nhanh.

Nhược điểm:

- Khó thực hiện truy vấn phức tạp.
- Phụ thuộc vào hạ tầng Google.



CÂU CHUYỆN VỀ SQL VÀ NOSQL



UBER POSTGRESQL → MYSQL

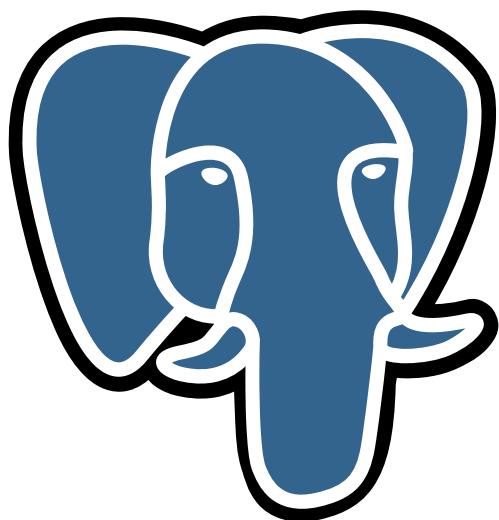


GIAI ĐOẠN ĐẦU

Uber sử dụng PostgreSQL do hỗ trợ ACID tốt, truy vấn linh hoạt và xử lý JSON mạnh mẽ.

Vấn đề khi mở rộng:

- Hiệu suất giảm khi số lượng giao dịch tăng cao.
- Replication chậm.
- Cơ chế xử lý đồng thời chưa tối ưu, gây độ trễ khi tải cao.



GIAI ĐOẠN SAU



Chuyển sang MySQL (MariaDB)

để tối ưu:

- Replication nhanh hơn.
- Sharding linh hoạt hơn, dễ dàng mở rộng quy mô.
- Ghi dữ liệu hiệu suất cao hơn trong hệ thống phân tán.

BÀI HỌC



- PostgreSQL mạnh về tính toàn vẹn dữ liệu và truy vấn phức tạp.
- MySQL (MariaDB) phù hợp hơn khi cần replication, sharding nhanh, tốc độ ghi cao.



DISCORD MONGODB -> SCYLLADB



GIAI ĐOẠN ĐẦU



Vấn đề với MongoDB

- Độ trễ cao khi dữ liệu tăng nhanh.
- Khả năng mở rộng kém.

Giải pháp:

- Chuyển sang Cassandra
- Mở rộng ngang tốt, tốc độ ghi nhanh hơn.

GIAI ĐOẠN SAU



ScyllaDB



Cassandra hoạt động tốt một thời gian

Vấn đề:

- "Hot partitions" gây mất cân bằng tải.
- Hiệu suất không ổn định khi dữ liệu tăng.

Giải pháp:

- Chuyển sang ScyllaDB
- Giảm số node từ 177 Cassandra → 72 ScyllaDB.
- Cải thiện hiệu suất, giảm độ trễ đáng kể.

BÀI HỌC



- Không có database hoàn hảo, cần chọn đúng theo nhu cầu.
- Hiệu suất đọc/ghi quan trọng khi hệ thống lớn, DB có khả năng scaling tốt để tránh độ trễ cao.
- Thiết kế phân tán ngay từ đầu, mở rộng dễ dàng, tránh phải di chuyển dữ liệu về sau.



**Cảm ơn mọi người
đã lắng nghe**