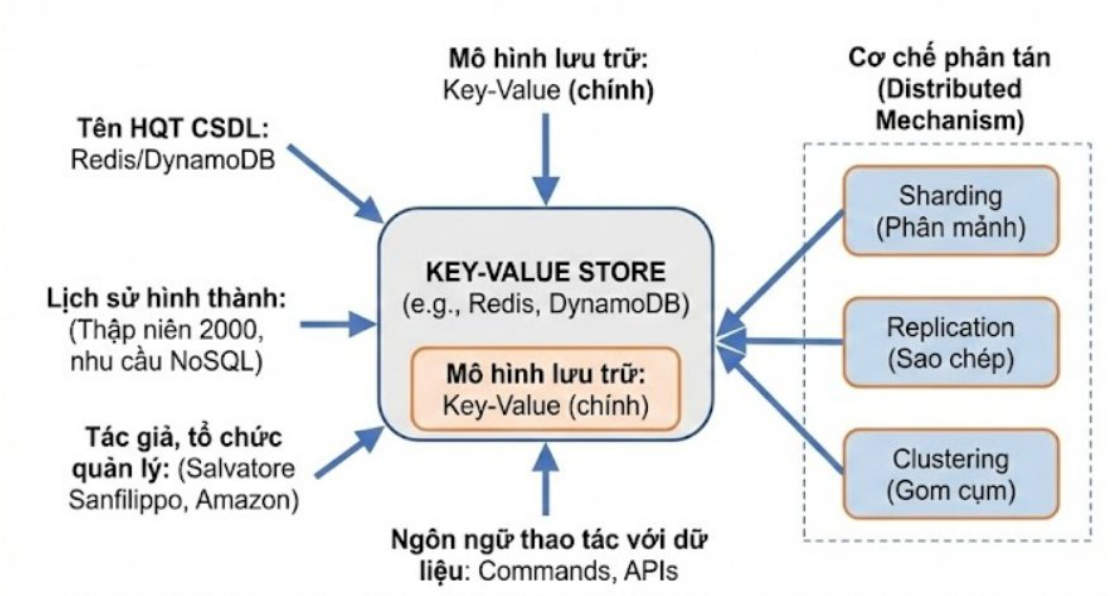
NGUYỄN THỊ DIỄM KIỀU
LỚP: 12_ĐH_CNTT5
MSSV: 1250080255
KIẾN TRÚC



Nguyễn Thị Diễm Kiều 12_ĐH_CNTT5
MSSV: 1250080255

1. Tên HQT CSDL: Key-Value Stores
- Là kiểu CSDL sử dụng mảng kết hợp. Trong đó mỗi key sẽ tồn tại duy nhất một value trong collection. Mối quan hệ này được gọi là một cặp key-value.
- Trong Hệ cơ sở dữ liệu phân tán (Distributed Database System), dữ liệu có thể được chia thành hàng triệu khối (blocks) và phân tán trên nhiều node để đảm bảo khả năng mở rộng, tính sẵn sàng và chịu lỗi.
- Với key-value Stores, mỗi bản ghi có key (vd: video id) và value là 1 object JSON chứa toàn bộ thông tin (title, url, description, ...)

2. Lịch sử hình thành
- Key-value Stores ra đời đầu những năm 2000 để đáp ứng nhu cầu đơn giản, nhanh và mở rộng cao cho các ứng dụng web lớn, khắc phục hạn chế của RDBMS khi xử lí dữ liệu lớn và phân tán. Key-value Stores là một mô hình ~~NoSQL~~ NoSQL, lưu trữ dưới dạng cặp key-value, ưu tiên hiệu năng và tính sẵn sàng, chấp nhận đánh đổi về nhất quán

- Các cột mốc quan trọng gồm Memcached (2003), Amazon Dynamo (2007) - nền tảng cho nhiều key-value Stores hiện tại, và Redis (2009) - key-value Stores in-memory phổ biến cho cache và session. Ngày nay, key-value Stores như Redis, Riak, Couchbase, Cassandra là thành phần quan trọng trong hệ thống phân tán và microservices.

3. Tác giả, tổ chức quản lí

- Key-value Stores là một mô hình dữ liệu NoSQL được phát triển bởi nhiều tổ chức và cá nhân; ví dụ nổi bật: Redis (do Salvatore Sanfilippo tạo ra), Memcached (Brad Fitzpatrick, được phát triển bởi nhiều công ty), và Riak (Basho Technologies). Chúng được quản lí và phát triển bởi các công ty công nghệ lớn (Amazon, Google) hoặc cộng đồng mã nguồn mở (Redis, Memcached).

4. Mô hình lưu trữ

- Lưu trữ dữ liệu dạng cặp key - value
- Key là duy nhất, value thường là JSON/ binary / object
- Không schema cố định → linh hoạt, truy cập nhanh.

5. Ngôn ngữ thao tác với dữ liệu
- Không dùng SQL
- Thao tác đơn giản: GET, PUT, DELETE, UPDATE,...
- Truy vấn chủ yếu dựa trên key
- Các lệnh nâng cao cho cấu trúc dữ liệu như TINCR (tăng giá trị) cho số, LPUSH/RPUSH (thêm vào list), HSET/HGET (thao tác hash) các key-value Stores như Redis, DynamoDB, Memcached có API riêng, nhưng nguyên tắc ~~số~~ thao tác cơ bản xoay quanh việc sử dụng một khoá duy nhất để truy cập giá trị tương ứng, hỗ trợ hiệu quả cho dữ liệu ~~lớn~~ lớn và có khả năng mở rộng cao.

6. Cơ chế phân tán
- Dữ liệu được partition (sharding) theo key
- Replication để tăng sẵn sàng và chịu lỗi.
- Thường tuân theo BASE / CAP theorem (ưu tiên availability và partition tolerance)

CODE MÔ PHỎNG:

```
DELIMITER //

--  PUT (Tương tự lệnh SET trong Redis)
CREATE PROCEDURE KV_PUT(IN p_key VARCHAR(255), IN p_value JSON)
BEGIN
    -- Tính toán Sharding: Lấy mã CRC32 của Key chia lấy dư cho 2
    DECLARE shard_id INT;
    SET shard_id = CRC32(p_key) % 2;

    -- Định tuyến dữ liệu vào đúng Node (Mô phỏng Router)
    IF shard_id = 0 THEN
        INSERT INTO kv_node_0 (k, v) VALUES (p_key, p_value)
        ON DUPLICATE KEY UPDATE v = p_value; -- Cơ chế Upsert
        SELECT CONCAT('Da luu vao Node_0: ', p_key) as Status;
    ELSE
        INSERT INTO kv_node_1 (k, v) VALUES (p_key, p_value)
        ON DUPLICATE KEY UPDATE v = p_value;
        SELECT CONCAT('Da luu vao Node_1: ', p_key) as Status;
    END IF;
END //

--GET
CREATE PROCEDURE KV_GET(IN p_key VARCHAR(255))
BEGIN
    DECLARE shard_id INT;
    SET shard_id = CRC32(p_key) % 2;

    -- Chỉ tìm kiếm ở đúng Node chứa Key đó (Tối ưu hóa hiệu năng)
    IF shard_id = 0 THEN
        SELECT k as 'Key', v as 'Value', 'Node_0' as 'Source'
        FROM kv_node_0 WHERE k = p_key;
    ELSE
        SELECT k as 'Key', v as 'Value', 'Node_1' as 'Source'
        FROM kv_node_1 WHERE k = p_key;
    END IF;
END //

--DELETE
CREATE PROCEDURE KV_DEL(IN p_key VARCHAR(255))
BEGIN
    DECLARE shard_id INT;
    SET shard_id = CRC32(p_key) % 2;

    IF shard_id = 0 THEN
        DELETE FROM kv_node_0 WHERE k = p_key;
        SELECT CONCAT('Da xoa tu Node_0: ', p_key) as Status;
    ELSE
        DELETE FROM kv_node_1 WHERE k = p_key;
        SELECT CONCAT('Da xoa tu Node_1: ', p_key) as Status;
    END IF;
END //

DELIMITER ;
```

```
--- GHI DỮ LIỆU (PUT) ---

CALL KV_PUT('user:1001', '{"name": "Nguyen Van A", "age": 25, "role": "admin"}');
CALL KV_PUT('user:1002', '{"name": "Tran Thi B", "city": "Hanoi"}');

-- Lưu thông tin Session
CALL KV_PUT('session:xyz', '{"status": "active", "expires": 3600}');

-- Lưu thông tin Product
CALL KV_PUT('product:555', '{"name": "Laptop Dell", "price": 1500}');


--- ĐỌC DỮ LIỆU (GET) ---
CALL KV_GET('user:1001');
CALL KV_GET('product:555');


--- KIỂM TRA PHÂN TÁN ---
SELECT * FROM kv_node_0;
SELECT * FROM kv_node_1;

--- XÓA DỮ LIỆU (DEL) ---
CALL KV_DEL('session:xyz');
```