1. **Partitioning là gì?**

* Partitioning là một chức năng cho phép bạn phân tán các phần của các tables riêng biệt qua một hệ thống file và dựa theo những quy định bạn đặt ra. Với mục đích này thì các phần của một bảng sẽ được lưu trữ tại các bảng riêng biệt và ở những nơi khác nhau.
* Có thể tạo Partitions bằng cách sử dụng câu lệnh CREATE TABLE, hoặc ALTER TABLE.
* Khi thực thi Partitioning tức là bạn phân chia dữ liệu thành các Partition theo một quy luật để phục vụ cho một mục đích nào đó.vd: phân chia theo năm, phân chia theo khu vực địa lý,…
* Sau khi đã thực hiện Partitioning thì khi mà có yêu cầu truy vấn dữ liệu thì MySQL sẽ dựa vào điều kiện truy vấn dữ liệu xem nó sẽ nằm ở những Partitions nào và chỉ tìm kiếm trong những Partitions đó mà bỏ qua các Partitions không match với điều kiện tìm kiếm.

1. **Các loại Partitioning:**

Trong MySQL bao gồm các loại Partitioning như sau: Range Partitioning, List Partitioning, Hash Partitioning, Key Partitioning, Sub Partitioning.

* **Range Partitioning:** 
  + Range Partitioning là việc phân chia bảng thành các phân vùng trong đó các Partition chứa các rows mà giá trị của một trường dùng để phân vùng nằm trong một khoảng nhất địnhn nào đó.
  + Phạm vi phân vùng nên liên tiếp nhau, không chồng chéo nên nhau.
  + Range Partitioning thường được sử dụng để phân vùng những giá trị liên quan đến ngày tháng.
  + Để tạo Partition, trong câu lệnh CREATE TABLE ta sử dụng mệnh đề PARTITION BY RANGE ().Các khoảng phân vùng được định nghĩa bằng cách sử dụng mệnh đề VALUES LESS THAN .

VD1: Ví dụ về tạo Range Partitioning:

Tạo một bảng Employeess lưu thông tin của các nhân viên:

CREATE TABLE employees (

id INT NOT NULL,

fname VARCHAR(30),

lname VARCHAR(30),

hired DATE NOT NULL DEFAULT '1970-01-01',

separated DATE NOT NULL DEFAULT '9999-12-31',

job\_code INT NOT NULL,

store\_id INT NOT NULL

);

Ta có thể có nhiều lựa chọn để thực hiện Partition.Ở đây ta chọn Partitioning theo store\_id. Câu lệnh thực hiện Partitioning như sau:

CREATE TABLE employees (

id INT NOT NULL,

fname VARCHAR(30),

lname VARCHAR(30),

hired DATE NOT NULL DEFAULT '1970-01-01',

separated DATE NOT NULL DEFAULT '9999-12-31',

job\_code INT NOT NULL,

store\_id INT NOT NULL

)

PARTITION BY RANGE (store\_id) (

PARTITION p0 VALUES LESS THAN (6),

PARTITION p1 VALUES LESS THAN (11),

PARTITION p2 VALUES LESS THAN (16),

PARTITION p3 VALUES LESS THAN (21)

);

Sau khi thự hiện partitioning thì những nhân viên có store\_id < 6 thì sẽ lưu ở partition p0, store\_id từ 6->10 thì lưu ở p1, store\_id từ 11->15 thì lưu ở p2 và store\_id từ 16->20 thì lưu ở p3.

* **List Partitioning:**
  + List Partitioning cũng tương tự với Range Partitioning.
  + Điểm khác của List Partitioning so với Range Partitioning là các row trong mỗi phân vùng của List Partitioning được lựa chọn chính xác từ tập giá trị của column mà thực hiện Partioning trên nó.
  + Để thực hiện List Partition, trong câu lệnh CREATE TABLE ta sử dụng mềnh đề PARTITION BY LIST (expr). Expr là column hoặc là một phép toán thực thi trên giá trị của column và trả về một số nguyên. Và để xác định các Partition ta sử dụng mệnh đề VALUES IN (value\_list,.).value\_list là danh sách các số nguyên phân cách nhau bởi dấu “,”.

VD2: tạo List Partitioning: Vẫn sử dụng bảng Employee được tạo ở ví dụ 1. Tại VD 2 ta chỉ tạo các partitions.

CREATE TABLE employees (

id INT NOT NULL,

fname VARCHAR(30),

lname VARCHAR(30),

hired DATE NOT NULL DEFAULT '1970-01-01',

separated DATE NOT NULL DEFAULT '9999-12-31',

job\_code INT,

store\_id INT

)

PARTITION BY LIST(store\_id) (

PARTITION p0 VALUES IN (3,5,6,9,17),

PARTITION p1 VALUES IN (1,2,10,11,19,20),

PARTITION p2 VALUES IN (4,12,13,14,18),

PARTITION p3 VALUES IN (7,8,15,16)

);

* Trong trường hợp mà ta thực hiện INSERT dữ liệu vào bảng đã được phân vùng bằng List Partitioning. Có thể xảy ra trường hợp mà giá trị của trường thêm vào không march với bất kì giá trị nào trong các phân vùng. Khi đó sẽ xảy ra lỗi và không INSERT được data. Để giải quyết ta có thể dử dụng thêm toán tử IGNORE trong câu INSERT để bỏ qua những giá trị mà không match còn những giá trị khác vẫn được thêm vào bảng.
* **Hash Partitioning:**
  + Với Range Partitioning và List Partitioning thì bạn phải chỉ rõ phạm vi hoặc giá trị của column sẽ được lưu vào các phân vùng. Còn đối với Hash Partitioning thì MySQL sẽ thực hiện việc này, bạn chỉ cần phải chỉ ra cột hoặc là một biểu thức tính toán dựa trên giá trị của cột và số lượng các Partitions cần tạo.
  + Để Partitioning một table sử dụng Hash Partitioning thì trong câu lệnh CREATE TABLE ta sử dụng mệnh đề PARTYTION BY HASH(expr). Trong đó [expr] là tên cột nếu giá trị của cột đó là Interger hoặc là một biểu thức tính toán trên các giá trị của cột mà trả về một số nguyên. Và mệnh đề PARTITION num, với num là số PARTITION mà bạn muốn tạo. Nếu không có mệnh đề PARTITION num thì mặc định số Partition tạo ra là 1.
  + Trong mệnh đề PARTYTION BY HASH(expr)thì giá trị mà expr trả về không được là hằng số, không phải là một số ngẫu nhiên(nói cách khác là nó xác định và thay đổi).
  + Khi sử dụng Hash Partitioning thì MySQL xác định Partition dựa trên 2 giá trị là expr của PARTYTION BY HASH(expr) và num của PARTITION num. Cụ thể thì sẽ xác định Partition sẽ lưu recode bằng cách sau: Giả sử record sẽ được lưu vào Partition N thì N sẽ được xác định bằng : N = MOD(expr, num).

VD3: tạo Hash Partition

CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATE)

PARTITION BY HASH( YEAR(col3) )

PARTITIONS 4;

Giả sử có bản ghi mới được thêm vào với giá trị col3 là ‘2005-09-01’ thì ta sẽ tính được vị trí Partition mà record đó sẽ lưu là : MOD(YEAR('2005-09-01'),4) = 1.

* **Key Partitioning:**
  + Key Partitioning cũng tương tự như Hash Partitioning ngoại trừ việc Hash Partitioning sử dụng biểu thức do người dùng định nghĩa còn Key Partitioning thì sử dụng hàm băm do MySQL Server cung cấp. MySQL Cluster thì sử dụng MD5(), còn đối với các bảng mà sử dụng các storage engine khác thì sẽ sử dụng hàm băm nội bộ.
  + Để tạo Key Partitioning cho table, trong mệnh đề CREATE TABLE ta sử dụng mệnh đề PARTITION BY KEY và PARTITION num. Trong đó num là số Partition muốn tạo.

VD4: tạo Key Partitioning cho table:

CREATE TABLE k1 (

id INT NOT NULL PRIMARY KEY,

name VARCHAR(20)

)

PARTITION BY KEY()

PARTITIONS 2;

* + Nếu table không có primary key mà có unique key thì Key Partitioning sẽ sử dụng unique key.

1. **Quản lý Partitioning.**

* MySQL cung câp nhiều chức năng để quản lý Partitioning như là : add, drop, redefine, merge hoặc split các Partition.
* Tất cả các chức năng trên đều có thể thực hiện bằng câu lệnh ALTER TABLE.
* Chú ý khi DROP Partition thì tất cả dữ liệu mà được lưu ở Partition đó sẽ cũng sẽ bị xóa hết.