**Chuẩn nf**

**1nf:**

Các row, bảng, dữ liệu dữ liệu là nguyên tử

**2nf:**

Tất cả các trường không khoá phải phụ thuộc hoàn toàn vào toàn bộ khoá chính.

Vd: nếu bảng có 2 khoá chính thì trường không khoá khác trong bảng phải phụ thuộc vào cả 2 khoá chính, không thể chỉ phụ thuộc vào 1 khoá.

**3nf:**

không tồn tại trường không khoá phụ thuộc và 1 trường không khoá khác trong bảng.

ví dụ: tên nhà sản xuất và id nhà sx không được nằm trong bảng sản phầm. mà chỉ có id nhà sx nằm trong bảng sp thôi. Và có thêm 1 bảng để lưu tên nhà sx riêng

Thứ tự thực hiện câu lệnh sql:

FROM -> JOIN -> WHERE -> GROUP -> HAVING -> SELECT -> ORDER

Khác biệt subquery và join:

Sub query thực hiện nhanh hơn chỉ lấy dữ liệu của bảng 1

Join sử dụng khi muốn lấy thêm trường của bảng thứ 2

**CTE**

Vd: with tmp as (select \* from customer)

Câu lệnh bắt đầu bằng with:

Giúp câu query cần join không cần map toàn bộ bảng mà chỉ map dữ liệu trong Cte thôi.

**Ranking:**

**Rank:**

Sử dụng để đánh thứ hạng bản ghi nhưng nó bỏ qua thứ tự rank

Nếu có 2 bản ghi cùng rank thì bản ghi thứ 3 sẽ có rank là 3

Lọc giá trị lớn nhất của từng đối tượng

Thường sử dụng trong trường hợp lấy ra dữ liệu lớn nhất hoặc nhỏ nhất của 1 đối tượng trong nhiều bản ghi.

Ví dụ: mỗi khách hàng có nhiều lần mua hàng mỗi lần mua có tổng giá trị đơn hàng khác nhau. Rank dùng để tìm đơn hàng lớn nhất hoặc nhỏ nhất của mỗi khách hàng.

**DENSE\_RANK**

Cũng sử dụng để đánh thứ hạng như rank nhưng nó không bỏ qua thứ tự rank

Nếu có 2 bản ghi cùng rank thì bản ghi thứ 3 sẽ có rank là 2

**Row\_number**

Sử dụng để đánh số thứ tự cho bản ghi. Nếu 2 bản ghi giống nhau vẫn được đánh số thứ tự khác nhau.

**View:**

Sử dụng để che giấu dữ liệu bảng gốc,

View thường chỉ sử dụng để xem dữ liệu không nên chỉnh sửa thông tin trong view

View thực sự không lưu dữ liệu mà thực chất nó lưu trữ câu lệnh truy vấn, mỗi khi mở view nó sẽ chạy lại câu lệnh truy vấn đó. Nên khi bản gốc thay đổi view cũng sẽ thay đổi dữ liệu cho lần mở sau đó.

**Store:**

Sử dụng: hiệu năng cải thiện.

Dữ liệu đầu vào không đổi:

Tránh deploy lại server khi sửa câu lệnh sql.

Để tối ưu hiệu suất của câu sql:

Index,

Partition,

**Index:**

Tìm kiếm nhanh hơn vì nó được sắp xếp và tìm kiếm theo nhị phân

Cluster:

Được sắp xếp theo cluster index và gắn với dữ liệu

Khi thay đổi dữ liệu chỉ thay đổi trực tiếp dữ liêu;

Sắp xếp trực tiếp trên bảng

Non-cluster:

Được sắp xếp theo field và trỏ tới cluster tương ứng.

Nó tạo ra vùng nhớ để chứ dữ liệu sắp xếp này

Khi thay đổi dữ liệu của Non-cluster nó sẽ phải sắp xếp lại

Sử dụng cho trường ít bị thay đổi

cluester index chỉ được đánh 1 khoá còn  non-cluster index đánh bao nhiêu cũng được nhưng không nên đánh tuỳ tiện.

**Partition:**

Nó sẽ phân đoạn dữ liệu theo gì đó:

Kết hợp giữa partition và index giúp cải thiện hiệu suất vì nó sẽ phân đoạn nên index nhỏ hơn -> tìm kiếm nhanh hơn

Partition phải được đánh thủ công và cập nhật liên tục.

Nếu dữ liệu thêm vào mà chưa tạo Partition sẽ lỗi và không thể thêm.

Để biết câu query có sử dụng index hay không thì sử dụng từ khoá EXPLAIN đầu câu sql. Lệnh EXPLAIN sẽ truy vấn ra toàn bộ thông tin, các bước thực hiện của câu lệnh sql. Câu truy vấn sử dụng index sẽ có field access type là INDEX hoặc INDEX RANGE SCAN

**Store:**

Sử dụng: hiệu năng cải thiện.

Dữ liệu đầu vào không đổi:

Tránh deploy lại server khi sửa câu lệnh sql.

Khi tạo store nó sẽ tạo 1 file chứa tập lệnh sql bên trong và có thể truyền tham số.

Tạo store bằng từ khoá Crete procedure <tên store> < tham số> as begin <Các tập lệnh sql> end;

Khi có tham số đầu ra ta thêm từ khoá Output ở tham số.