* **Char(10)** luôn chiếm 10 ô nhớ chỉ chứa kí tự tiếng anh.
* **Varchar(10)** cấp phát động theo dữ liệu được lưu trữ, tối đa 10 ô nhớ
* **Nchar(10)** luôn chiếm 10 ô nhớ và lưu được kí tự unicode
* **Nvarchar(10)** cấp phát động theo dữ liệu được lưu trữ, tối đa 10 ô nhớ và lưu được kí tự unicode
* **Khóa chính:** dùng để định danh bản ghi duy nhất trong bảng, vì là định danh bản ghi nên nó là duy nhất trong bảng, nó không chấp nhận giá trị rỗng.
* **Khóa ngoại:** là 1 trường trong 1 bảng tham chiếu đến 1 khóa chính trong 1 bảng khác. Mục đích là để tạo ra cỡ sở dữ liệu quan hệ và đảm bảo tính toàn vẹn của dữ liệu. Khóa ngoại có thể là rỗng nhưng mà nếu khác rỗng thì dữ liệu phải là khóa chính của bảng mà nó tham chiếu tới, có thể có nhiều khóa ngoại trong 1 bảng.
* **Toàn vẹn dữ liệu:** là việc đặt ra các quy tắc trong 1 cơ sở dữ liệu nhằm kiểm tra các giá trị của dữ liệu trước khi được lưu trữ phải đảm bảo tính chính xác hợp lý bên trong 1 cơ sở dữ liệu.
* **Inner Join:** là tổ hợp 2 bảng và lấy ra bản ghi thỏa mãn điều kiện join



* **Full join:** là tổ hợp 2 bảng lấy ra các bản ghi thỏa mãn điều kiện join và các bản ghi không thỏa mãn điều kiện join ở 2 bảng cũng sẽ được lấy ra, những trường không có dữ liệu sẽ là null
* ****
* **Left join:** là tổ hợp 2 bảng lấy ra các bản ghi thỏa mãn điều kiện join và các bản ghi không thỏa mãn điều kiện join ở bảng bên trái
* ****
* **Right join:** là tổ hợp 2 bảng lấy ra các bản ghi thỏa mãn điều kiện join và các bẩn ghi không thỏa mãn điều kiện join ở bảng bên phải
* ****
* **Union:** là select gộp, gộp kết quả của 2 or nhiều câu select thành 1 và loại bỏ đi các bản ghi duplicate chỉ để lại 1 bản ghi duy nhất (muốn lấy tất cả duplicate thì dùng **union all**) -> số trường truy vấn ở mỗi câu phải bằng nhau, kiểu dữ liệu của chúng phải giống nhau, và cách sắp xếp các cột phải tương ứng nhau
* **Select into:** dùng để tạo ra bảng mới từ các dữ liệu vừa select. Mục đích để backup dữ liệu
* **Insert into select:** insert dữ liệu từ câu select vào bảng đã tồn tại.
* **Group by:** nhóm các bản ghi có giá trị giống nhau theo những trường nào đó và dùng các hàm count(), max(), min(), avg() để tính toán giá trị trên 1 trường khác. **Chú ý**: cột hiển thị phải là trường nằm trong group by or Agreegate function, không được dùng select\*
* **Ví du:**
* -- Xuất ra Danh sách tên bộ môn và số lượng giáo viên của bộ môn đó
* select BOMON.TENBM, count(GIAOVIEN.MAGV) from GIAOVIEN right join BOMON
* on GIAOVIEN.MABM = BOMON.MABM
* group by TENBM
* order by TENBM asc
* **having:** tương tự như where trong select nhưng dùng trong group by. **Chú ý:** trường dùng kiểm tra điều kiện trong having phải là trường trong group by or Agreegate function(tính toán trên trường group by
* -- Xuất ra Danh sách tên bộ môn và số lượng giáo viên của bộ môn mà số lượng > 2
* select BOMON.TENBM, count(\*) as tsl from GIAOVIEN, BOMON
* where GIAOVIEN.MABM = BOMON.MABM
* group by BOMON.TENBM
* having count(\*) > 1
* --Xuất ra Lương và số lượng người nhận mức lương đó với điều kiện Lương > 2000 và số lượng người nhận mức lương đó lớn hơn 1
* select LUONG, count(\*) from GIAOVIEN
* group by LUONG
* having LUONG > 2000 and count(\*) > 1
* **View:** là bảng ảo được tạo từ dữ liệu của câu truy vấn, view sẽ cập nhập dữ liệu mới nhất theo bảng mà view truy vấn tới mỗi khi lấy view ra sử dụng

create view TestView as

select \* from GIAOVIEN

select \* from TestView

-- Update view

Alter view TestView as

select HOTEN from GIAOVIEN

-- Tạo view có dấu

create view [học sql server] as

select \* from KHOA

select \* from [học sql server]

* **Index:** là 1 cấu trúc dữ liệu giúp ta định vị và truy cập nhanh nhất vào dữ liệu trong các bảng của database -> index giúp ta truy vấn một cách nhanh hơn nhưng lại giảm tốc độ của việc insert, update, delete dữ liệu vì mỗi lần insert, update, delete nó lại phải update lại cái index (cấu trúc dữ liệu) -> nên hạn chế tạo index trên những cột mà dữ liệu của nó thường xuyên bị thay đổi
* CREATE INDEX index\_name  
  ON table\_name (column1, column2, ...);
* **Cách tối ưu câu truy vấn SQL:**

**+ Đánh index 1 cách hợp lý**: vì index giúp tăng perfomance của truy vấn nhưng lại giảm tốc độ của việc insert, update vì thế nên đánh index vào những trường ít có thay đổi

**+ Chỉ nên lấy ra các trường cần thiết tránh select \***

**+ Nếu trong câu select join 2 bảng và có điều kiện where thì ta nên select theo where trước rồi join sau**

**+ Nên tránh việc dùng Group by having nếu có thể viết truy vấn bằng where vì group by phải nhóm rồi tính toán so sánh trên các trường đó thì tốc độ sẽ chậm hơn**

* **Full text search:** dùng để search gần đúng và giúp seach nhanh hơn.

**+** thay vì đánh index trên từng row thì nó sẽ đánh trên từng từ (Inverted Index)

* **Hạn chế của câu lệnh like:**

**+** chỉ search được row đã định sẵn

**+** performance không tốt

**+ .**

12. JPA là 1 chuẩn, 1 cái interface mô tả các phương thức để java có thể tương tác với các cơ sở dữ liệu quan hệ.

13. Hibernate là 1 ORM framework(mã nguồn mở), nó sẽ implement JPA, nó giúp chúng ta có thể dễ dàng thao tác với DB bằng cách hỗi trợ mapping giữa đối tượng trong java với các table.

Hibernate sử dụng HQL để query dữ liệu (nó khác sql ở chỗ nó tương tác với Entity chứ không phải các table)

14. Spring data jpa là 1 phần trong hệ sinh thái spring Data giúp chúng ta thao tác với db 1 cách dễ dàng hơn. Nó wrapper hibernate lại và tạo ra 1 interface mạnh mẽ.

Spring data jpa hỗi trợ tạo ra các câu Query từ tên method mà không cần phải code

Spring data jpa còn hỗi trợ viết các câu truy vấn thông thường bằng cách thêm @Query ở trên tên method