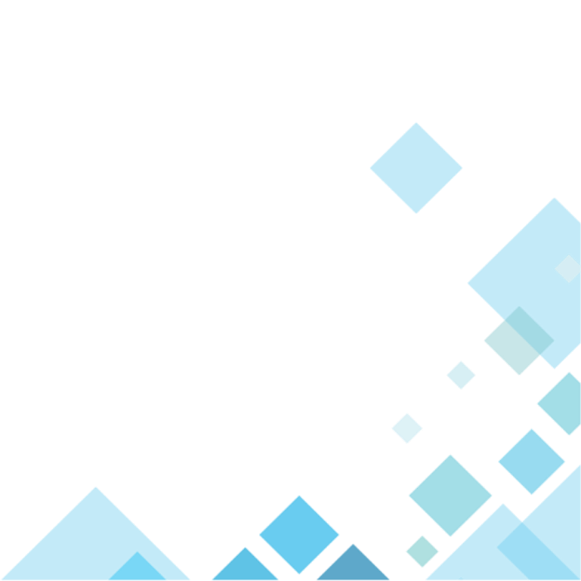
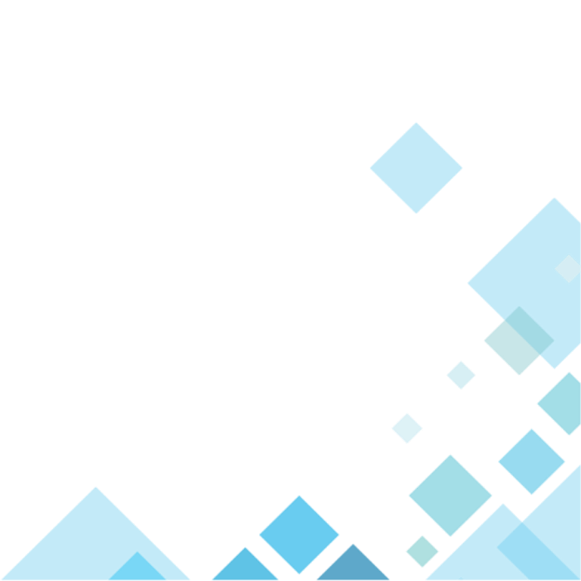
TÀI LIỆU

**POSTGRESQL CƠ BẢN**

Người tạo tài liệu: PHẠM NGỌC QUỐC

Ngày tạo tài liệu: 16/04/2022

Ngày sửa cuối cùng: 16/04/2022

Version: 1.0

Mục lục

[***1***](#_heading=h.30j0zll) ***SQL CHUNG 4***

[**1.1**](#_heading=h.1fob9te) **Giới thiệu về SQL 4**

[**1.2**](#_heading=h.3znysh7) **Câu lệnh SELECT 5**

[**1.3**](#_heading=h.2et92p0) **Từ khóa DISTINCT 5**

[**1.4**](#_heading=h.tyjcwt) **Mệnh đề WHERE 6**

[**1.5**](#_heading=h.3dy6vkm) **Mệnh đề GROUP BY 7**

[**1.6**](#_heading=h.1t3h5sf) **Mệnh đề HAVING 8**

[**1.7**](#_heading=h.4d34og8) **Mệnh đề LIMIT 8**

[**1.8**](#_heading=h.2s8eyo1) **Toán tử logic AND và OR 8**

[**1.9**](#_heading=h.17dp8vu) **Từ khóa ORDER BY 8**

[**1.10**](#_heading=h.3rdcrjn) **Câu lệnh INSERT INTO 9**

[**1.11**](#_heading=h.26in1rg) **Câu lệnh UPDATE 9**

[**1.12**](#_heading=h.lnxbz9) **Câu lệnh DELETE 9**

[**1.13**](#_heading=h.35nkun2) **Toán tử LIKE 10**

[**1.14**](#_heading=h.1ksv4uv) **Toán tử IN 10**

[**1.15**](#_heading=h.44sinio) **Toán tử BETWEEN 10**

[**1.16**](#_heading=h.2jxsxqh) **Toán tử UNION 10**

[**1.17**](#_heading=h.z337ya) **SQL ALIASES 11**

[**1.18**](#_heading=h.3j2qqm3) **SQL JOINS 11**

[**1.19**](#_heading=h.1y810tw) **Từ khóa INNER JOIN 11**

[**1.20**](#_heading=h.4i7ojhp) **Từ khóa LEFT JOIN 12**

[**1.21**](#_heading=h.2xcytpi) **Từ khóa RIGHT JOIN 13**

[**1.22**](#_heading=h.1ci93xb) **Từ khóa FULL JOIN 13**

[**1.23**](#_heading=h.3whwml4) **Câu lệnh INSERT INTO SELECT 14**

[**1.24**](#_heading=h.2bn6wsx) **Câu lệnh CREATE DATABASE 14**

[**1.25**](#_heading=h.qsh70q) **Câu lệnh CREATE TABLE 14**

[**1.26**](#_heading=h.3as4poj) **SQL Constraints 15**

[**1.27**](#_heading=h.1pxezwc) **SQL DROP 16**

[**1.28**](#_heading=h.49x2ik5) **SQL ALTER 17**

[***2***](#_heading=h.2p2csry) ***Cài đặt PostgreSQL và cấu hình các thông số quan trọng 17***

[**2.1**](#_heading=h.147n2zr) **Chọn file cài đặt 17**

[**2.2**](#_heading=h.3o7alnk) **Cấu hình file pg\_hba.conf 17**

[**2.3**](#_heading=h.23ckvvd) **Cấu hình file postgresql.conf 18**

[***3***](#_heading=h.ihv636) ***Kiểm tra khi PostgreSQL lỗi 20***

# SQL CHUNG

## Giới thiệu về SQL

* Bảng(Table) : Một CSDL bao gồm một hoặc nhiều bảng. Mỗi bảng được xác định thông qua một tên(ví dụ : dmkh, dmnv, …). Bảng chứa các dòng là dữ liệu của bảng.
* Ví dụ dưới đây là bảng « dmkh ».



* Với SQL ta có thể truy vấn CSDL và nhận lấy kết quả trả về thông qua các câu truy vấn.
* Ví dụ, một câu truy vấn như sau :

SELECT ma\_kh, ten\_kh FROM dmkh ;

* Sẽ trả về kết quả như sau :



* Một số lệnh quan trọng nhất trong SQL :
* **SELECT** : Lấy dữ liệu từ bảng trong CSDL.
* **UPDATE** : Cập nhật/sửa đổi dữ liệu trong bảng.
* **DELETE** : Xóa dữ liệu trong bảng.
* **INSERT INTO** : Thêm mới dữ liệu vào bảng.
* **CREATE DATABASE** : Tạo CSDL mới.
* **ALTER DATABSE** : Sửa CSDL.
* **CREATE TABLE** : Tạo bảng mới.
* **ALTER TABLE** : Sửa thông tin bảng.
* **DROP TABLE** : Xóa một bảng.
* **CREATE INDEX** : Tạo một chỉ mục(dùng để tăng tốc độ khi tìm kiếm).
* **DROP INDEX** : Xóa bỏ chỉ mục trong bảng.



* Với SQL ta có thể truy vấn CSDL và nhận lấy kết quả trả về thông qua các câu truy vấn.

## Câu lệnh SELECT

* Câu lệnh SELECT dùng để lấy dữ liệu từ một CSDL.
* Cú pháp của lệnh SELECT :

SELECT tên\_cột1, tên\_cột2, …   
FROM tên\_bảng;

* Hoặc

SELECT \* FROM tên\_bảng;

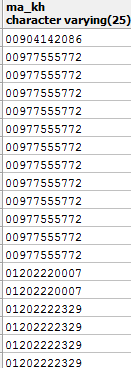
* Ví dụ : Lấy Mã khách hàng, Tên khách hàng, Địa chỉ trong Danh mục khách hàng

SELECT ma\_kh, ten\_kh, dia\_chi FROM dmkh

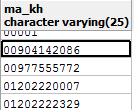
## Từ khóa DISTINCT

* Dùng để xóa những bản ghi trùng nhau khi SELECT.
* Cú pháp của lệnh DISTINCT :

SELECT DISTINCT tên\_cột1, tên\_cột2, …   
FROM tên\_bảng

* Ví dụ : bảng trước khi DISTINCT
* 
* Sau khi DISTINCT:

SELECT DISTINCT ma\_kh FROM ct00;



## Mệnh đề WHERE

* Để truy xuất dữ liệu trong bảng theo các điều kiện nào đó, một mệnh đề WHERE có thể thêm vào câu lệnh SELECT.
* Cú pháp :

SELECT tên\_cột FROM tên\_bảng  
WHERE tên\_cột phép\_toán giá\_trị

* Ví dụ :

SELECT ma\_kh, ten\_kh FROM dmkh WHERE ma\_kh = 'KH001' ;

* Quy tắc sử dụng dấy nháy đơn (‘) :
* SQL sử dụng dấu nháy đơn bao quanh các giá trị ở dạng chuỗi văn bản (text).
* Các giá trị dạng kiểu số không dùng dấu nháy bao quanh.
* Ví dụ với dạng chuỗi văn bản :

Câu lệnh đúng :

SELECT \* FROM dmkh WHERE ma\_kh = 'KH001';

Câu lệnh sai :

SELECT \* FROM dmkh WHERE ma\_kh = KH001 ;

* Ví dụ với dạng Số :

Câu lệnh đúng :

SELECT \* FROM dmkh WHERE kh\_yn = 1 ;

Câu lệnh sai :

SELECT \* FROM dmkh WHERE kh\_yn = '1' ;

* Các toán tử trong mệnh đề WHERE :

|  |  |
| --- | --- |
| Toán tử | Mô tả |
| = | So sánh bằng |
| <> | So sánh khác nhau. Có thể dùng toán tử != |
| > | Lớn hơn |
| < | Nhỏ hơn |
| >= | Lơn hơn hoặc bằng |
| <= | Nhỏ hơn hoặc bằng |
| BETWEEN | Nằm trong khoảng |
| LIKE | So sánh mẫu chuỗi |
| IN | Xác định nhiều giá trị cho một cột  (   * Toán tử IN cho phép bạn chỉ định nhiều giá trị trong mệnh đề WHERE. * Toán tử IN là viết tắt của nhiều điều kiện OR. * **SELECT** (tên) cột1, cột2, cột3...**FROM** tên\_bảng **WHERE** tên\_cột **IN** (giá\_trị1, giá\_trị2, ...); * Ex: **SELECT** \* **FROM** khach\_hang **WHERE** dia\_chi **NOT** **IN** ('Gia Lâm', 'Hà Đông', 'Hoàng Mai');   ) |

## Mệnh đề GROUP BY

* Mệnh đề GROUP BY trong SQL được sử dụng kết hợp với lệnh SELECT để sắp xếp dữ liệu đồng nhất vào trong các nhóm.
* Trong SQL, mệnh đề GROUP BY theo sau mệnh đề WHERE trong một lệnh SELECT và ở trước mệnh đề ORDER BY.

SELECT cot1, cot2, … aggregate\_function(Tên\_cột)

FROM ten\_bang

WHERE [ dieu\_kien ]

GROUP BY cot1, cot2

ORDER BY cot1, cot2

* Ví dụ :

SELECT ma\_vt, SUM(sl\_xuat) as sl\_xuat

FROM ct70

GROUP BY ma\_vt

ORDER BY ma\_vt

## Mệnh đề HAVING

* Dùng để thêm vào điều kiện sau khi GROUP BY. Vì từ khóa WHERE không thể sử dụng cho các hàm tập hợp.
* Ví dụ :

SELECT ma\_vt, SUM(sl\_xuat) as sl\_xuat

FROM ct70

GROUP BY ma\_vt

HAVING sum(sl\_xuat) >100

ORDER BY ma\_vt

* Ví dụ trên lấy tổng Số lượng xuất của mỗi mã hàng với điều kiện là > 100.

**// Note**

## Mệnh đề LIMIT

* Dùng để xác định cố định số lượng bản ghi trả về.
* Mệnh đề này rất hữu ích khi muốn lấy ra số lượng bản ghi trong 1 bảng chứa nhiều dữ liệu.
* Cú pháp :

SELECT Tên\_cot1, Tên\_cot2, … FROM tên\_bảng Limit Số\_lượng\_bản\_ghi ;

* Ví dụ :

SELECT \* FROM dmkh LIMIT 100

## Toán tử logic AND và OR

* Hai toán tử này nối hai hoặc nhiều điều kiện trong mệnh đề WHERE với nhau.
* Toán tử AND sẽ trả về dữ liệu nếu **tất cả** các điều kiện đều thỏa mãn.
* Toán tử OR sẽ trả về dữ liệu nếu **bất kỳ** điều kiện nào được thỏa mãn.
* Ví dụ 1 : Toán tử AND : Tìm trong danh mục khách hàng có Mã là KH001 và địa chỉ ở Hà Nội.

SELECT \* FROM dmkh WHERE ma\_kh = 'KH001' AND dia\_chi = 'Hà Nội';

* Ví dụ 2 : Toán tử OR : Tìm trong danh mục khách hàng có Mã là KH001 hoặc địa chỉ ở Hà Nội.

SELECT \* FROM dmkh WHERE ma\_kh = 'KH001' OR dia\_chi = 'Hà Nội';

* Ví dụ 3 : Sự kết hợp Toán tử AND và Toán tử OR : Tìm trong danh mục khách hàng có Mã đvcs là ‘01’ và Mã là KH001 hoặc địa chỉ ở Hà Nội.

SELECT \* FROM dmkh WHERE ma\_dvcs = '01' AND (ma\_kh = 'KH001' OR dia\_chi = 'Hà Nội') ;

* Lưu ý : Với những điều kiện thuộc OR phải có đóng mở ngoặc đơn để không bị tìm kiếm sai.

## Từ khóa ORDER BY

* Được dùng để sắp xếp kết quả trả về theo một hoặc nhiều cột.
* Trong từ khóa **ORDER BY** việc sắp xếp các bản ghi theo chiều tăng dần được mặc định. Để sắp xếp các bản ghi theo chiều giảm dần, ta có thể dùng thêm từ khóa **DESC**.
* Cú pháp :

SELECT Tên\_cột1, Tên\_cột2,…  
FROM tên\_bảng  
ORDER BY tên\_cột1 ASC|DESC, Tên\_cột2 ASC|DESC, …;

* Ví dụ 1: Lấy Mã khách, Tên khách từ danh mục khách hàng theo mã khách tăng dần theo thứ tự ABC

SELECT ma\_kh, ten\_kh FROM dmkh ORDER BY ma\_kh;

SELECT ma\_kh, ten\_kh FROM dmkh ORDER BY ma\_kh ;

* Ví dụ 2: Lấy Mã khách, Tên khách, Địa chỉ từ danh mục khách hàng theo mã khách tăng dần theo thứ tự ABC và địa chỉ giảm dần theo thứ tự ABC

SELECT ma\_kh, ten\_kh, dia\_chi FROM dmkh ORDER BY ma\_kh ASC, dia\_chi DESC;

SELECT ma\_kh, ten\_kh,dia\_chi FROM dmkh ORDER BY ma\_kh ASC, dia\_chi DESC ;

## Câu lệnh INSERT INTO

* Câu lệnh này dùng để thêm mới dữ liệu vào bảng.
* Cú pháp :

INSERT INTO tên\_bảng(Tên\_côt1, Tên\_côt2, Tên\_côt3, ….)

VALUES(Gia\_trị1, Gia\_tri2, Giá\_trị3, ….) ;

* Ví dụ : Thêm bản ghi mới vào bảng dmkh :

INSERT INTO dmkh(ma\_kh, ten\_kh, dia\_chi, kh\_yn) VALUES (‘KH002’, ‘nguyen manh hung’, ‘Ha\_Noi’, 1);

**//INSERT nhiều bản ghi một lúc:**

INSERT INTO table\_name (column\_list)

VALUES

(value\_list\_1),

(value\_list\_2),

...

(value\_list\_n);

INSERT INTO dmkh(ma\_kh, ten\_kh, dia\_chi, kh\_yn)

VALUES(' KH002', 'Công ty KH001', 'Hà Nội', 1) ;

## Câu lệnh UPDATE

* Dùng để Cập nhật/sửa đổi dữ liệu trong bảng.
* Cú pháp :

UPDATE tên\_bảng  
SET Tên\_cột1 = Giá\_trị1, Tên\_cột2 = Giá\_trị2,...  
WHERE Các\_cột = các\_giá\_trị;

* **Lưu ý : Mệnh đề WHERE dùng để lựa chọn các bản ghi để update, nếu Mệnh đề WHERE trống thì toàn bộ bản ghi trong bảng sẽ được cập nhật.**
* Ví dụ :

UPDATE dmkh SET ten\_kh = 'Công ty KH001', dia\_chi = 'Dương Đình Nghệ - Hà Nội' WHERE ma\_kh = 'KH001';

## Câu lệnh DELETE

* Dùng để xóa dữ liệu trong bảng.
* Cú pháp :

DELETE FROM tên\_bảng  
WHERE Các\_cột = các\_giá\_trị;

* **Lưu ý : Mệnh đề WHERE dùng để lựa chọn các bản ghi để DELETE, nếu Mệnh đề WHERE trống thì toàn bộ bản ghi trong bảng sẽ bị xóa.**
* Ví dụ :

DELETE FROM dmkh WHERE ma\_kh = 'KH001';

## Toán tử LIKE

* Toán tử này được dùng trong mệnh đề WHERE để tìm kiếm theo mẫu trong 1 cột của bảng.
* Ví dụ 1 : Lấy danh sách khách hàng có tên khách bắt đầu bằng chữ C

SELECT \* FROM dmkh WHERE ten\_kh LIKE 'C%';

* Ví dụ 2 : Lấy danh sách khách hàng có tên khách kết thúc bằng chữ C

SELECT \* FROM dmkh WHERE ten\_kh LIKE '%C';

* Ví dụ 3 : Lấy danh sách khách hàng có tên khách có chứa chữ C

SELECT \* FROM dmkh WHERE ten\_kh LIKE '%C%';

## Toán tử IN

* Toán tử này được dùng trong mệnh đề WHERE để xác định nhiều giá trị.
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng  
WHERE tên\_cột IN (giá\_trị1,giá\_trị2,...);

* Ví dụ  : Lấy danh sách khách hàng có địa chỉ là ‘Hà Nội ‘ hoặc ‘HCM’

SELECT \* FROM dmkh WHERE dia\_chi IN(‘Hà Nội’, ‘HCM’);

SELECT \* FROM dmkh WHERE dia\_chi IN('Hà Nội', 'HCM');

## Toán tử BETWEEN( đi cùng AND)

* Toán tử này được dùng trong mệnh đề WHERE để xác định giá trị nằm trong khoảng.
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng  
WHERE tên\_cột BETWEEN giá\_trị1 AND giá\_trị2;

* Ví dụ  : Lấy danh sách khách hàng có ngày sinh từ 01/01/1980 đến 31/12/1980

SELECT \* FROM dmkh WHERE ngay\_sinh BETWEEN ‘1980-01-01’ AND ‘1980-12-31’;

SELECT \* FROM dmkh WHERE ngay\_sinh BETWEEN '1980-01-01' AND '1980-12-31';

* Lưu ý : Định dạng kiểu ngày là : Năm-Tháng-Ngày

## Toán tử UNION

* Toán tử UNION cho phép bạn nối kết quả của hai hoặc nhiều câu truy vấn lại với nhau để trở thành một danh sách kết quả duy nhất.
* Cú pháp UNION:

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1

UNION

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng2

* **Khi sử dụng UNION phải lưu ý các nguyên tắc sau :**
* ***Số lượng colums trong tất cả các lệnh SELECT phải bằng nhau.***
* ***Mỗi column tương ứng vị trí phải có cùng kiểu dữ liệu và độ dài.***
* Ví dụ :

select ma\_kh, ten\_kh from dmkh

UNION

select ma\_kh, ten\_kh from ph81

* Chú ý : **UNION sẽ loại bỏ các kết quả trùng lặp của mệnh đề SELECT .**
* UNION ALL :
* Tương tự như UNION, sẽ dữ lại kết quả trùng lặp
* Cú pháp UNION ALL:

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1

UNION ALL

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng2

## SQL ALIASES

* Dùng để đổi tên 1 bảng hoặc 1 cột trong câu lệnh SELECT
* Cú pháp đổi tên cột :

SELECT Tên\_cột1 AS Tên\_cột1\_mới,Tên\_cột2 AS Tên\_cột2\_mới,…  
FROM Tên\_bảng;

* Cú pháp đổi tên bảng:

SELECT Tên\_cột1, Tên\_cột2,…  
FROM Tên\_bảng as Tên\_bảng\_mới;

* Ví dụ :

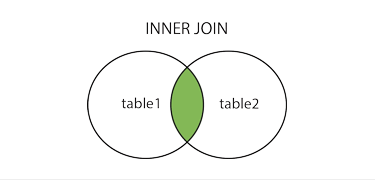
SELECT ma\_kh AS ma\_kh\_new, ten\_kh AS ten\_kh\_new FROM dmkh ;

## SQL JOINS

* Một mệnh đề JOIN dùng để kết hợp nhiều dòng từ 2 hoặc nhiều bảng dựa trên các cột giữa chúng.
* Có các loại JOIN sau :
* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

## Từ khóa INNER JOIN

* Ý nghĩa như hình ảnh sau :



* Chỗ màu xanh là kết quả của INNER JOIN
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 INNER JOIN Tên\_bảng2

ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

Hoặc

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 JOIN Tên\_bảng2

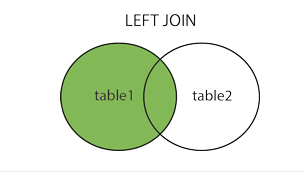
ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

* Ví dụ :

SELECT dmkh.ma\_kh, dmkh.ten\_kh from dmkh inner JOIN ct00 ON dmkh.ma\_kh = ct00.ma\_kh

## Từ khóa LEFT JOIN()

* Ý nghĩa như hình ảnh sau :



* Chỗ màu xanh là kết quả của LEFT JOIN
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 LEFT JOIN Tên\_bảng2

ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

Hoặc

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 LEFT OUTER JOIN Tên\_bảng2

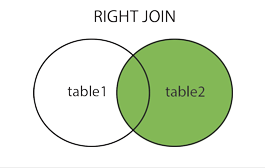
ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

* Ví dụ :

SELECT ct00.ma\_kh, dmkh.ten\_kh from ct00 LEFT JOIN dmkh ON ct00.ma\_kh = dmkh.ma\_kh

## Từ khóa RIGHT JOIN()

* Ý nghĩa như hình ảnh sau :



* Chỗ màu xanh là kết quả của RIGHT JOIN
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 RIGHT JOIN Tên\_bảng2

ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

Hoặc

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 RIGHT OUTER JOIN Tên\_bảng2

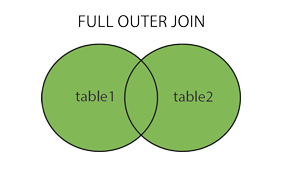
ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

* Ví dụ :

SELECT ct00.ma\_kh, dmkh.ten\_kh from ct00 RIGHT JOIN dmkh ON ct00.ma\_kh = dmkh.ma\_kh

## Từ khóa FULL JOIN()

* Ý nghĩa như hình ảnh sau :



* Chỗ màu xanh là kết quả của FULL JOIN
* Cú pháp :

SELECT Tên\_cột1,Tên\_cột2,…  
FROM Tên\_bảng1 FULL OUTER JOIN Tên\_bảng2

ON Tên\_bảng1.Tên\_cột = Tên\_bảng2.Tên\_cột

* Ví dụ :

SELECT ct00.ma\_kh, dmkh.ten\_kh from ct00 FULL OUTER JOIN dmkh ON ct00.ma\_kh = dmkh.ma\_kh

## SQL View

* Bạn có thể hiểu một cách cực kỳ đơn giản, việc tạo 1 View có thể hiểu nôm na là tạo nên 1 "bảng tạm" .

Khi bạn gọi đến view, đồng nghĩa với việc bạn gọi câu query mà bạn gán khi tạo VIEW. “*VIEW thì không được lưu trữ vật lý, nó giống như một bảng ảo, Thay vào đó, câu query sẽ được chạy mỗi lần gọi tới VIEW*."

* Cú pháp tạo view:

CREATE [OR REPLACE] [TEMP|TEMPORARY] VIEW name [(column\_name [...])]

[WITH (view\_option\_name[= view\_option\_value] [, ... ])]

AS query

* Cú pháp xóa view

|  |
| --- |
| DROP VIEW Tên\_View |

## Câu lệnh INSERT INTO SELECT

* Dùng để copy thông tin từ 1 bảng sang bảng khác
* Copy toàn bộ dữ liệu từ bảng này qua bảng khác :

INSERT INTO table2

SELECT \* FROM table1;

* Trường hợp này, cấu trúc 2 bảng phải giống hệt nhau.
* Copy một số cột cần qua bảng khác

INSERT INTO Tên\_bảng2(Tên\_cột1,Tên\_cột2,…)  
SELECT Tên\_cột1,Tên\_cột2,…

FROM Tên\_bảng1

## Câu lệnh CREATE DATABASE

* Dùng để tạo ra 1 CSDL.

CREATE DATABASE kh\_data;

## Câu lệnh CREATE TABLE

* Lệnh này dùng để tạo ra 1 bảng trong CSDL.
* Cú pháp :

CREATE TABLE table\_name

(

column\_name1 data\_type(size),

column\_name2 data\_type(size),

column\_name3 data\_type(size),

....

);

* Trong đó :
* column\_name là tên cột trong bảng.
* Data\_type là kiểu dữ liệu của cột có thể lưu trữ(Ví dụ : integer, date,…)
* Size : Xác định độ dài tối đa của cột trong bảng
* Ví dụ :

CREATE TABLE dmkh1

(

ma\_kh ud\_ma,

ten\_kh ud\_ten,

CONSTRAINT dmkh1\_pkey PRIMARY KEY (ma\_kh)

)

WITH (

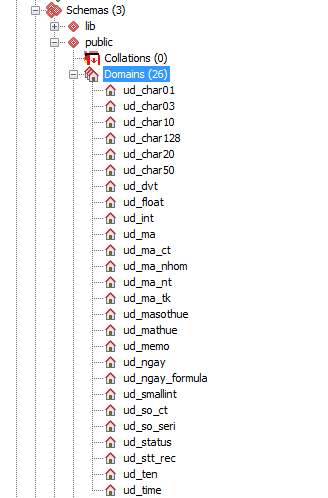
OIDS=TRUE

);

ALTER TABLE dmkh1

OWNER TO postgres;

* Trong đó data\_type ud\_ma, ud\_ten,… là các kiểu dữ liệu tự định nghĩa trong CSDL.



## SQL Constraints

* Dùng để xác định nguyên tắc cho dữ liệu của bảng
* Constraints có thể được xác định khi tạo bảng(CREATE TABLE) hoặc sửa bảng(ALTER TABLE).
* Cú pháp :

CREATE TABLE table\_name

(

column\_name1 data\_type(size) constraint\_name,

column\_name2 data\_type(size) constraint\_name,

column\_name3 data\_type(size) constraint\_name,

....

);

* Các loại Constraints hay dùng sau :
* NOT NULL : Cột không thể lưu giá trị NULL.
* PRIMARY KEY : Khóa chính - mỗi table thông thường chúng ta sẽ có một khóa chính nhằm mục đích phân biệt giữa các dòng (records) dữ liệu. Khóa chính để đảm bảo bảng không được lưu trùng số liệu.
* DEFAULT : Xác định giá trị mặc định cho cột.
* **PRIMARY KEY**
* Ví dụ :

CREATE TABLE dmkh1

(

ma\_kh ud\_ma,

loai\_kh ud\_smallint,

ten\_kh ud\_ten,

CONSTRAINT dmkh1\_pkey PRIMARY KEY (ma\_kh, loai\_kh)

)

WITH (

OIDS=TRUE

);

ALTER TABLE dmkh1

OWNER TO postgres;

* Trong danh sách khóa chính có thể là 1 hoặc nhiều cột.
* Cú pháp thêm Constraints **PRIMARY KEY** vào bảng :

ALTER TABLE Tên\_bảng

ADD CONSTRAINT Tên\_Constraints PRIMARY KEY (Tên\_cột1, Tên\_cột2, ..)

* Ví dụ :

ALTER TABLE dmkh ADD CONSTRAINT dmkh\_pkey PRIMARY KEY (ma\_kh)

* Cú pháp xóa Constraints **PRIMARY KEY** ra khỏi bảng :

ALTER TABLE Tên\_bảng

DROP CONSTRAINT Tên\_Constraints ;

## SQL DROP

* Câu lệnh này dùng để xóa : INDEX, TABLE, DATABSE
* DROP INDEX
* Cú pháp :

ALTER TABLE table\_name DROP INDEX index\_name

* DROP TABLE
* Được sử dụng để xóa một bảng và tất cả dữ liệu, chỉ mục (index), trigger, ràng buộc và quyền được trao cho bảng đó.
* GHI CHÚ: Phải thật cẩn thận trong khi sử dụng lệnh này, bởi vì một khi một bảng bị xóa vĩnh viễn
* Cú pháp :

DROP TABLE Tên\_bảng ;

* DROP DATABASE : Dùng để xóa 1 CSDL
* Cú pháp :

DROP DATABASE Tên\_database

* TRUNCATE TABLE : Dùng để xóa dữ liệu trong bảng cần xóa mà không thực thi Trigger.
* Ví dụ :

TRUNCATE TABLE dmgiaban

* Lệnh này xóa toàn bộ dữ liệu từ bảng dmgiaban mà không tự xóa trong dmgiabanct.

## SQL ALTER

1. Lệnh này dùng để thêm, sửa, xóa cột trong bảng.

* Thêm mới cột vào 1 bảng :

ALTER TABLE table\_name  
ADD column\_name datatype

* Xóa 1 cột trong bảng :

ALTER TABLE table\_name  
DROP COLUMN column\_name

Đổi kiểu dữ liệu của cột :

ALTER TABLE table\_name  
ALTER COLUMN column\_name TYPE data\_type

Ví dụ :

ALTER TABLE dmvv

ALTER COLUMN ten\_vv type ud\_char128;

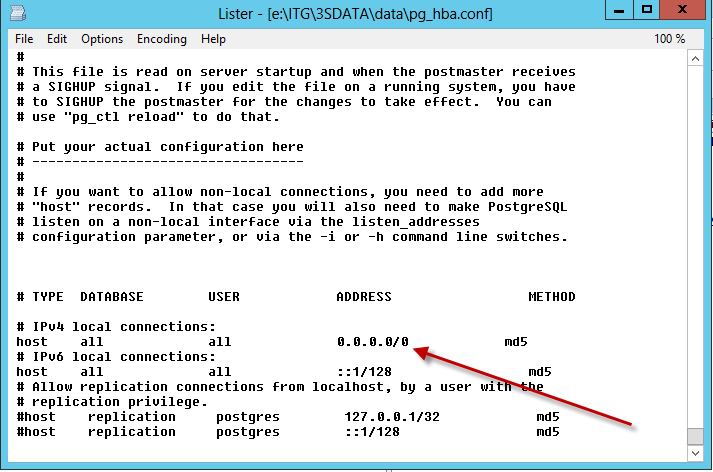
# Cài đặt PostgreSQL và cấu hình các thông số quan trọng

## Chọn file cài đặt

* Máy tính Win 32 bit thì cài Version cho Win 32 bít. Phiên bản phổ biến và ổn định nhất hiện tại là : 9.3.5-1(postgresql-9.3.5-1-windows.exe).
* Máy tính Win 64 bit thì cài Version cho Win 64 bít. Phiên bản phổ biến và ổn định nhất hiện tại là : 9.3.5-1(postgresql-9.3.5-1-windows-x64.exe).
* Cấu hình :

## Cấu hình file pg\_hba.conf

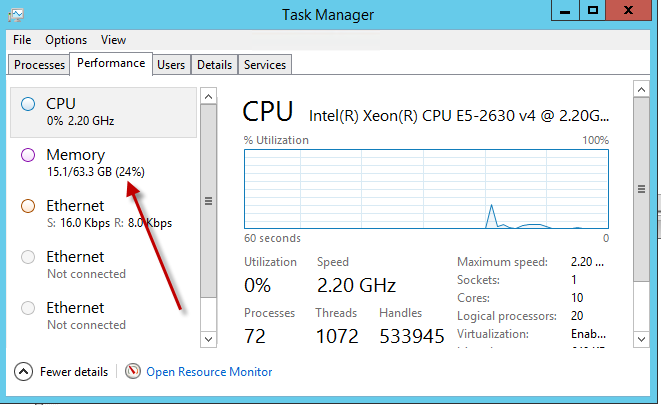
* Cấu hình file này thì các máy tính con mới kết nối được máy chủ chứa CSDL.
* Cấu hình **ADDRESS** như hình mũi tên đỏ.



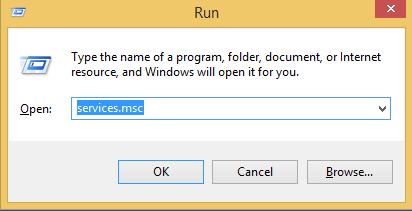
* **METHOD** : để nguyên md5, nếu chuyển thành trust thì Password đăng nhập vào database không có hiệu lực.

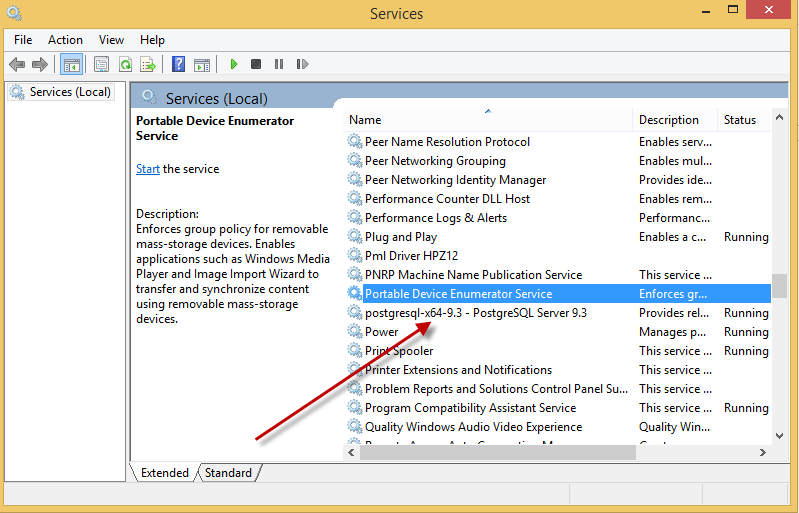
## Cấu hình file postgresql.conf

* File này chứa các thông tin quan trọng như tăng tốc độ, ….
* **max\_connections** : Cho phép số lượng kết nối đến Database tối đa trong 1 thời điểm là bao nhiêu. Bỏ dấu # và restart lại Services Postgres. Tùy thuộc vào lượng người sử dụng hệ thống. Mỗi 1 chức năng của hệ thống là 2 kết nối đến Database(1 \_data, 1 \_sys). Như vậy, ví dụ 1 người dùng mở 10 chức năng là 20 kết nối 🡪 5 người sử dụng là mất 100 kết nối. Khi cài đặt PostgreSQL tham số này mặc định là 100 🡪 chỉ cần 6 người dùng, mỗi người mở 10 chức năng là sẽ bị lỗi hệ thống. Như vậy nếu công ty có 50 người dùng, mỗi người mở trung bình 10 chức năng 1 lúc thì tham số này thiết lập là 1000.
* **shared\_buffers** : Thiết lập = 25% lượng RAM khả dụng của hệ thống(Lượng RAM khả dụng của hệ thống là lượng RAM còn lại chưa sử dụng).
* Cách xem lượng RAM khả dụng : Vào Task Manager



* Ví dụ trong ảnh, Lượng Ram đã sử dụng là 15.1 GB 🡪 Lượng RAM còn lại là 63.3 – 15.1 ~ 48GB 🡪 Thiết lập tham số **shared\_buffers**  là 12GB
* **effective\_cache\_size** : (Tốt nhất để = 50% bộ nhớ khả dụng của hệ thống). Tiếp ví dụ trên. Sau khi thiết lập tham số **shared\_buffers**  là 12GB thì RAM khả dụng = 63.3 – 15.1 – 12 ~ 36GB 🡪 Thiết lập tham số **effective\_cache\_size** là 18 GB. Trên đây là ví dụ với máy chủ RAM 64GB, chúng ta thiết lập tương tự theo tùy thuộc vào dung lượng RAM máy chủ.
* **max\_locks\_per\_transaction** : thiết lập số lượng giao dịch mở đồng thời. (Thiết lập = 6400).
* Cách mở Services PostgreSQL :
* Vào Start\Run. Gõ services.msc





# Kiểm tra khi PostgreSQL lỗi

* Chuột phải vào My Computer🡪Chọn Manage🡪 Chọn Event Viewer 🡪 Window Logs 🡪 Application

