## Database (PSQL)

PostgreSQL là một hệ thống quản trị cơ sở dữ liệu quan hệ-đối tượng (object-relational database management system).  
PostgreSQL được thiết kế để chạy trên các nền tảng tương tự UNIX.  
PostgreSQL là một phần mềm mã nguồn mở miễn phí

<https://www.tutorialspoint.com/postgresql/>

docker exec -it 05b3a3471f6f bash

~~root@05b3a3471f6f:/# psql -U postgres~~

# psql -U postgres -d tpi\_next\_development

psql (14.2 (Debian 14.2-1.pgdg110+1))

Type "help" for help.

postgres=t \*.\*

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at

----+-------------+------------+--------------------+-------+------------+------------

(0 rows)

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at

----+-------------+------------+--------------------+-------+------------+------------

(0 rows)

tpi\_next\_development=# ALTER TABLE key\_area\_results

tpi\_next\_development-# ADD author VARCHAR(50);

Cachs 2:

tpi\_next\_development-# ALTER TABLE key\_area\_results ADD memostotal jsonb;

tpi\_next\_development-# ALTER TABLE key\_area\_results ADD memostotal2 jsonb NOT NULL;

ok

 t.jsonb "memostotal2", null: false

sh prep\_docker.sh sẽ migrate db đã tạo từ docker vào schema.db

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at | author

----+-------------+------------+--------------------+-------+------------+------------+--------

(0 rows)

<https://stackoverflow.com/questions/37694987/connecting-to-postgresql-in-a-docker-container-from-outside>

-U username

--username=username Connect to the database as the user username instead of the default. (You must have permission to do so, of course.)

PostgreSQL

Là dạng RDBMS  
Dùng GUI: Download tool pgAdmin GUI tool

|  |  |
| --- | --- |
| service postgresql status | Kiểm tra xem postgresql có chạy không  Active là OK |
| psql pms\_db | Kết nối DB |
| \l command in psql | Show ra các list DB |
| \c some\_database | the command \c followed by the database name. This will allow you to enter the database and start performing queries |
| \dt; | Display table list |
| \d table\_name; | Display of table structure |
| select \* from table\_name; | List the data in the table |
| select \* from admin\_value order by column\_name; | Display the contents of the specified column in ascending order |
| select \* from admin\_value order by column\_name desc; | Display the contents of the specified column in descending order |
| select \* from table\_name limit NUMBER; | Indication number designation |
| select \* from table\_name offset NUMBER; | Specify the display start position |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tham khảo thêm (Không hữu ích thật sự)

<https://kb.objectrocket.com/postgresql/how-to-show-databases-in-postgresql-848>

<https://openplanning.net/11325/cai-dat-co-so-du-lieu-postgresql-tren-ubuntu>

Hay chứa nhiều lệnh cần: <https://qiita.com/H-A-L/items/fe8cb0e0ee0041ff3ceb>

Tiếng Việt : <https://vinasupport.com/database/postgresql/>

### Truy cập Database

$ psql pms\_db #to connect pms\_db

$ select \* from table\_name; // chú ý ;

VD: select \* from job\_p\_usb; #job\_p\_usb là 1 table trong cơ sở dữ liệu // chú ý ;

Then, you can access the PostgreSQL using the psql by typing the following command:

$ psql

You’ll access the postgres prompt like this:

postgres=#

To quit the PostgreSQL prompt, you run the following command:

postgres=# \q

<https://www.postgresqltutorial.com/install-postgresql-linux/>

Drop database

To destroy the database demo on the default database server:

$ dropdb demo

<https://www.postgresql.org/docs/9.3/app-dropdb.html>

Clean Database

- dropdb pms\_db

- rm /custom/postgres/db\_flag

### OK Postgre

docker exec -it 05b3a3471f6f bash

~~root@05b3a3471f6f:/# psql -U postgres~~

# psql -U postgres -d tpi\_next\_development

psql (14.2 (Debian 14.2-1.pgdg110+1))

Type "help" for help.

postgres=t \*.\*

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at

----+-------------+------------+--------------------+-------+------------+------------

(0 rows)

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at

----+-------------+------------+--------------------+-------+------------+------------

(0 rows)

tpi\_next\_development=# ALTER TABLE key\_area\_results

tpi\_next\_development-# ADD author VARCHAR(50);

Cachs 2:

tpi\_next\_development-# ALTER TABLE key\_area\_results ADD memostotal jsonb;

tpi\_next\_development-# ALTER TABLE key\_area\_results ADD memostotal jsonb NOT NULL;

ok

 t.jsonb "memostotal2", null: false

sh prep\_docker.sh sẽ migrate db đã tạo từ docker vào schema.db

tpi\_next\_development=# select \* from key\_area\_results;

id | key\_area\_id | project\_id | checkpoint\_results | memos | created\_at | updated\_at | author

----+-------------+------------+--------------------+-------+------------+------------+--------

(0 rows)

<https://stackoverflow.com/questions/37694987/connecting-to-postgresql-in-a-docker-container-from-outside>

-U username

--username=username Connect to the database as the user username instead of the default. (You must have permission to do so, of course.)

Cài postgresql Server trên Window

rake db:create

connection to server at "localhost" (::1), port 5432 failed: Connection refused (0x0000274D/10061)

Is the server running on that host and accepting TCP/IP connections?

connection to server at "localhost" (127.0.0.1), port 5432 failed: Connection refused (0x0000274D/10061)

Is the server running on that host and accepting TCP/IP connections?

Couldn't create 'projectName1\_development' database. Please check your configuration.

Pass : 1

Port 5432

<https://openplanning.net/10713/cai-dat-co-so-du-lieu-postgresql-tren-windows#a114891>

C:\Program Files\PostgreSQL\14\pgAdmin 4\bin

Open postgresql Server in linux, Docker

docker run -itd -e POSTGRES\_USER=baeldung -e POSTGRES\_PASSWORD=baeldung -p 5432:5432 -v /data:/var/lib/postgresql/data --name postgresql postgres

Verify if it is running:

ps auxwww | grep postgres

<https://itviec.com/it-jobs/junior-fresher-java-dev-from-800-techcom-securities-2250>

<https://www.tcbs.com.vn/home>

Công nghệ chúng tôi đang sử dụng:

- Kiến trúc: Microservices

- Ngôn ngữ lập trình: Go, Java, Python, Typescript

- Message broker: Kafka, RabbitMQ

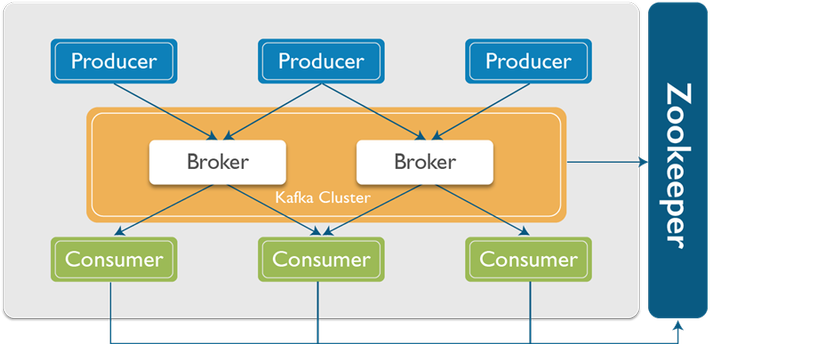
- Cơ sở dữ liệu: Oracle, Redis, MongoDB

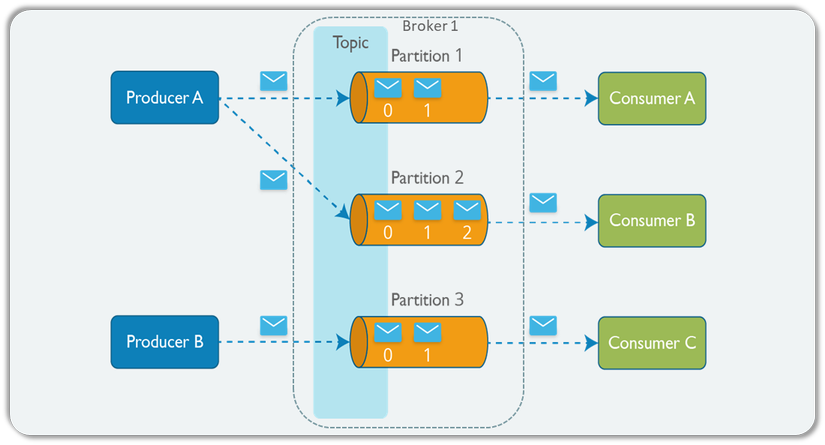
- Công nghệ xử lý realtime: Apache Flink, Kafka

- Công nghệ triển khai và vận hành: IBM Datapower (API Gateway), Docker, OpenShift, Jenkins, Ansible, ElasticSearch, Grafana, Prometheus, AWS/Azure.

Kafka:

Kafka là gì? – Đó là hệ thống message pub/sub phân tán (distributed messaging system). Bên pulbic dữ liệu được gọi là producer, bên subscribe nhận dữ liệu theo topic được gọi là consumer. Kafka có khả năng truyền một lượng lớn message theo thời gian thực, trong trường hợp bên nhận chưa nhận message vẫn được lưu trữ sao lưu trên một hàng đợi và cả trên ổ đĩa bảo đảm an toàn





<https://topdev.vn/blog/kafka-la-gi/>

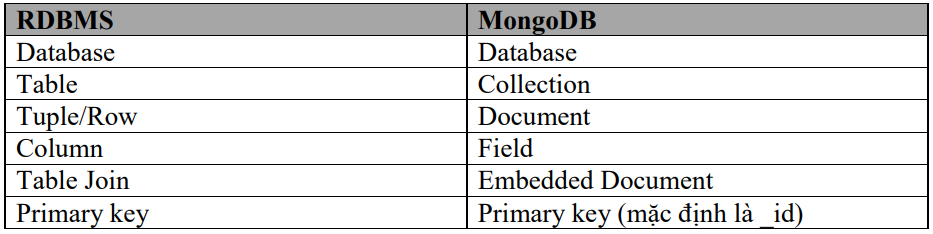
cơ sở dữ liệu quan hệ :

So sánh Orance, Redis, MongoDB

|  |  |  |  |
| --- | --- | --- | --- |
|  | Orance | Redis | MongoDB |
| Giống |  | cơ sở dữ liệu không quan hệ (non-relational database) | |
|  |  |  | MongoDB là một cơ sở dữ liệu tài liệu NoSQL lưu trữ thông tin trong cấu trúc tài liệu giống JSON |
|  |  |  |  |
| Khi nào sử dụng |  |  | MongoDB có thể phù hợp nhất với kiến trúc ít giản đồ để tạo mẫu, các ứng dụng trò chơi, di động và nội dung phong phú hiện đại.  Website bạn có nhiều dữ liệu quá |
|  |  |  | Mã nguồn mở, viết bằng C++ |
|  |  |  | MongoDB là document database ( dữ liệu trong MongoDB lưu dưới định dạng JSON ) |
| Ưu điểm |  |  | Khả năng mở rộng cực lớn: việc mở rộng dữ liệu mà không phải lo đến các vấn đề như khóa ngoại, khóa chính, kiểm tra ràng buộc, |
| Nhược điểm |  |  | Dữ liệu được caching, lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.  Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao |
|  |  |  |  |

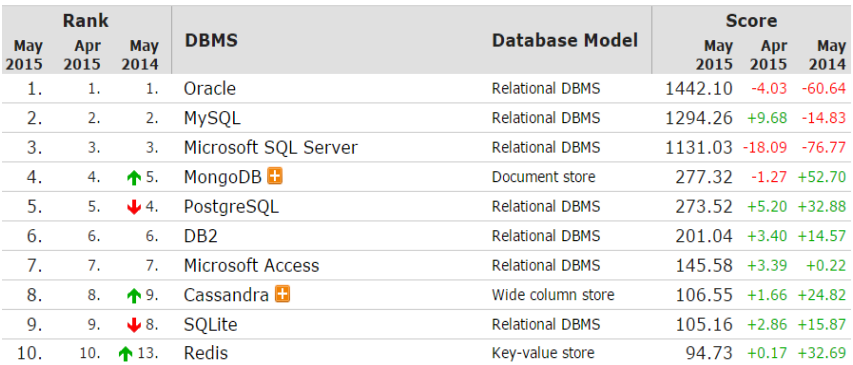
<https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3>

|  |  |  |
| --- | --- | --- |
| So sánh | RDBMs | Non-SQL |
|  | Cũ | Mới hơn |
| Mô hình | Table | Collection Key/Value, Columnbased – Tabular,  Document-based, Graphbased data-stores |
| Yêu cầu phần cứng cho dữ liệu lớn | Yêu cầu phần cứng máy chủ cao. | Không nhất thiết phần cứng phải cao, có thể mở rộng máy chủ ở nhiều nơi với tài nguyên phần cứng thấp. |
| Cách lấy dữ liệu từ client | Thông qua chuỗi kết nối (tên máy chủ, tên cơ sở dữ liệu, tên tài khoản và mật khẩu). | Thông qua API (Application Programming Interface) |
| Hiệu xuất | Kém hơn  SQL  Relational giữa các bảng | Cực tốt  Bỏ qua SQL  Bỏ qua các ràng buộc dữ  liệu |
| Thay đổi số node trong  hệ thống | Phải shutdown cả hệ thống. Việc thay đổi số node phức tạp. | Không cần phải shutdown cả hệ thống.  Việc thay đổi số node đơn giản, không ảnh hưởng đến hệ thống |
| Hỗ trợ lập trình hướng  đối tượng | Kém | Tốt |
|  |  |  |
|  |  |  |



Một số câu lệnh dùng trong MongoDB

| **Câu lệnh** | **SQL** | **MongoDB** |
| --- | --- | --- |
| Create table | CREATE TABLE people (id MEDIUMINT NOT NULL AUTO\_INCREMENT, user\_id Varchar(30), age Number, status char(1), PRIMARY KEY (id)) | db.people.insertOne({User\_id: “abc123”, Age: 55, Status: “A”}) |
| Drop table | DROP TABLE people | db.people.drop() |
| Insert records into tables | INSERT INTO people(user\_id, age, status) VALUES ("bcd001", 45, "A") | db.people.insertOne( { user\_id: "bcd001", age: 45, status: "A" }) |
| Select | SELECT \*FROM people | db.people.find() |
|  | SELECT id,user\_id, status FROM people | db.people.find( { }, { user\_id: 1, status: 1 } ) |
|  | SELECT \* FROM people WHERE status = "A" | db.people.find( { status: "A" } ) |
|  | SELECT \* FROM people WHERE status = "A" AND age = 50 | db.people.find( { status: "A", age: 50 } ) |
|  | SELECT \* FROM people WHERE status = "A" OR age = 50 | db.people.find( { $or: [ { status: "A" } , { age: 50 } ] } ) |
|  | SELECT \* FROM people WHERE user\_id like "%bc%" | db.people.find( { user\_id: /bc/ } ) db.people.find( { user\_id: { $regex: /bc/ } } ) |
|  | SELECT COUNT(user\_id) FROM people | db.people.count( { user\_id: { $exists: true } } ) db.people.find( { user\_id: { $exists: true } } ).count() |
| Update records | UPDATE people SET status = "C" WHERE age > 25 | db.people.updateMany( { age: { $gt: 25 } }, { $set: { status: "C" } } ) |
|  | UPDATE people SET age = age + 3 WHERE status = "A" | db.people.updateMany( { status: "A" } , { $inc: { age: 3 } } ) |
| Delete Records | DELETE FROM people WHERE status = "D" | db.people.deleteMany( { status: "D" } ) |
|  | DELETE FROM people | db.people.deleteMany({}) |



**Hệ thống MicroService**

Đi đường dài và có định hướng scale up

Nghiệp vụ nhiều và phức tạp

Lượng người dùng lớn

Team phải có độ chất nhất định (lèo tèo đú trend coi như tự kỷ)

Bổ sung các nghiệp vụ mới dễ dàng và khi đưa lên hệ thống phải đảm an toàn không làm ảnh hưởng nhiều tới hệ thống

SQL Lite trong Portable Project

SQLite dữ liệu nhỏ, mọi thứ đã có sẵn trong file CDSL (kèm SQL database engine)

Sau khi tạo Database rồi -> lưu file => các câu lệnh truy cấn bằng SQL Lite

File DatabaseContext.cs

SQLiteCommand sqlQuery = new SQLiteCommand();

public DatabaseContext(string dbFilePath = "", string dbName = "") :

base(new SQLiteConnection()

{

ConnectionString = new SQLiteConnectionStringBuilder()

{

DataSource = dbFilePath + dbName, ForeignKeys = true

}.ConnectionString

}, true)

{

\_dbFilePath = dbFilePath;

}

public void CreateDatabase(string dbPath)

{

Directory.CreateDirectory(Path.GetDirectoryName(dbPath));

if (!File.Exists(dbPath))

{

SQLiteConnection.CreateFile(dbPath);

}

}

using (SQLiteConnection m\_dbConnection = new SQLiteConnection("Data Source=" + dbPath))

{

m\_dbConnection.Open();

SQLiteCommand sqlQuery = new SQLiteCommand();

sqlQuery.CommandText = createQuery;

sqlQuery.Connection = m\_dbConnection;

sqlQuery.ExecuteNonQuery();

}

createQuery = “CREATE TABLE CompanyInfo(CUSTOMER\_ID VARCHAR(255) PRIMARY KEY, COMPANY VARCHAR(255) NOT NULL CHECK (length(COMPANY) > 0 ) )"

dbPath = "../userInfo/PortableDatabase.db"

SQLite error (1555): abort at 56 in

[INSERT INTO [Video]([VIDEO\_ID],

[VIDEO\_TIME],

[VIDEO\_PATH],

[IS\_GASLEAK],

[VISIBLE\_SIZE\_W],

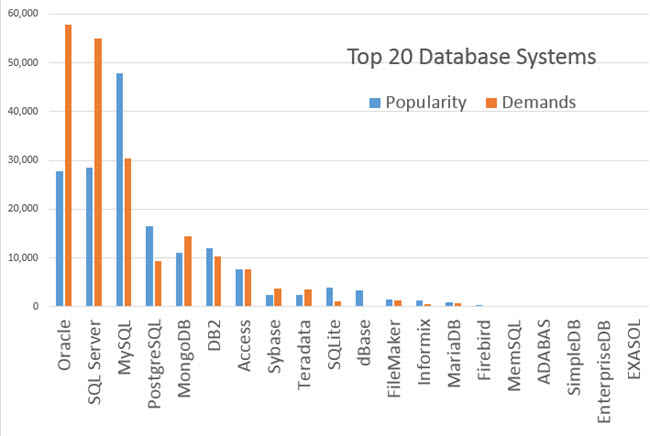
[VISIBLE\_SIZE\_H],

[DETECTABILITY\_JUDGEMENT],

[VIDEO\_DETECTION\_CONDITION\_ID],

[CAMERA\_ENVIRONMENT\_ID],

Mức độ phổ biến của các CSDL, DataBase, DB:



Các cơ sở dữ liệu được quản lý, vận hành bởi các chương trình về cơ sở dữ liệu (hệ quản trị CSDL), có rất nhiều loại chương trình mà bạn lựa chọn cho phù hợp với mục đích ứng dụng của bạn, có thể kể ra như:

Oracle RDBMS thiết kế cho doanh nghiệp, dữ liệu phức tạp và lớn

Microsoft SQL Server hệ quản trị rất tốt, nhanh của Microsoft

IBM DB2

Teradata

MySQL phù hợp với các ứng dụng vừa và nhỏ, có phiên bản miễn phí

PostgreSQL

Microsoft Access hệ CSDL đi cùng bộ MS Office

MariaDB giống MySQL, nguồn mở, miễn phí

Server

Quan trọng nhất khi làm DBA

- Kinh nghiệm

- Cẩn thận

- Chịu khó đọc tl về DB

- Chịu khó backup Server

- Sao lưu backup định kì

Chứng chỉ:

* CSA – Certificate Solutions Architect
* AWS Certified Developer
* Oracle DataBase 10G Administrator Certified.

Đọc tài liệu trang Acloud.guru

novice/ˈnɑː.vɪs/

inception: trong sự khởi đầu => it is inception

RDS (Dịch vụ đám mây) – cung cấp giải pháp cài đặt, vận hành, mở rộng dành cho relational database.  
Các hệ quản trị CSDL mà RDS hỗ trợ:

* SQL Server
* Oracle
* My SQL, Maria DB
* Aurora

Network : VPC, VPN, mạng con

Storage : S3 (1 trong những dịch vụ quan trọng nhất), CloudFont, EBS

Mô hình CS hạ tầng đám mây dưới dạng dịch vụ (IaaS)

1980 -2000 : DBMS quan hệ

2000 - nay : NoSQL (nhanh)

Học về Oracle:

<https://oracle-base.com/>

<https://docs.oracle.com/database/121/index.html>

Học chung chung:

Medium.com

Pluralsight.com

Tìm hiểu:

- SQL Server Reporting Services (SSRS)

-

Services:

SQS

SNS

SWF

CloudWatch

Tier /tɪr/

SaaS : Software as a Service (Email, ERP …)

PaaS: Platform as a Service (Web, streaming …)

IaaS : Infrastructure as a Service (Networking, File…. – ví dụ: EC2 , MS Azure Platform , VMWare)

SQL Server Services

Cài đặt SQL Server 2005 Express

SQL Server Management Studio Express/Developer (Free)

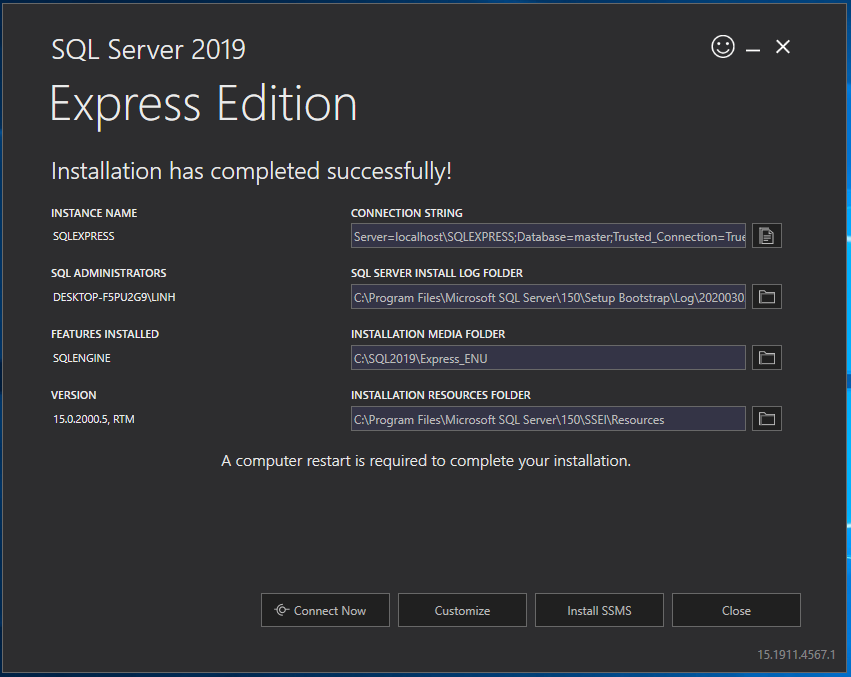
Enterprice/ Standard/ Web (Phí đắt)

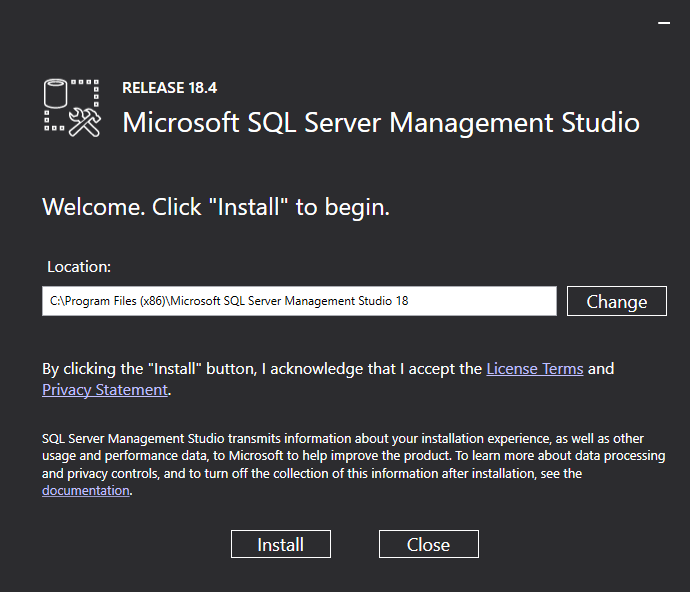
Chọn SQL Server Management Studio Express (SSMS) (free)

* Quản lí SQL Server thay vì dùng dòng lệnh

Link: <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>

……………………Cài khá lâu





Câu lệnh:

SELECT

INSERT

UPDATE

DELETE

Web:

|  |  |  |
| --- | --- | --- |
|  | Web tĩnh | Web động |
| Ngôn ngữ | Viết bằng HTML | Viết bằng ASP.NET, JS, PHP |
| Nội dung | Trang .html chỉ cung cấp thông tin | Tương tác với người dùng, sự kiện nhập liệu, nhấn chuột |
| CSDL | Không có CSDL | Lưu trữ dữ liệu trong CSDL |

ASP. NET là 1 framework được phát triển để xây dựng các trang web động kết hợp HTML, CSS, JS

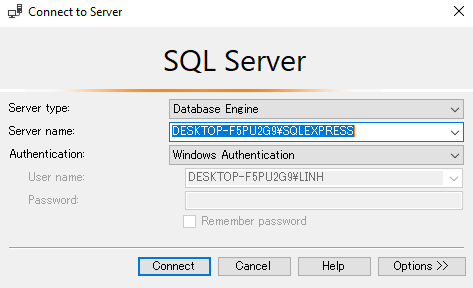
2 mô hình cho 1 Web form :

+ Trang đơn

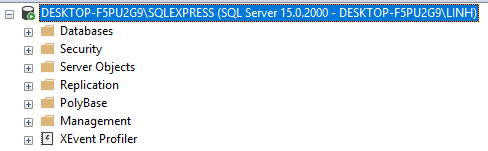
+ Trang code-behind (dễ đọc, dễ viết)

Entity /ˈen.t̬ə.t̬i/ : thực thể đối tượng

Connect to Server:



Sau khi connect thành công:

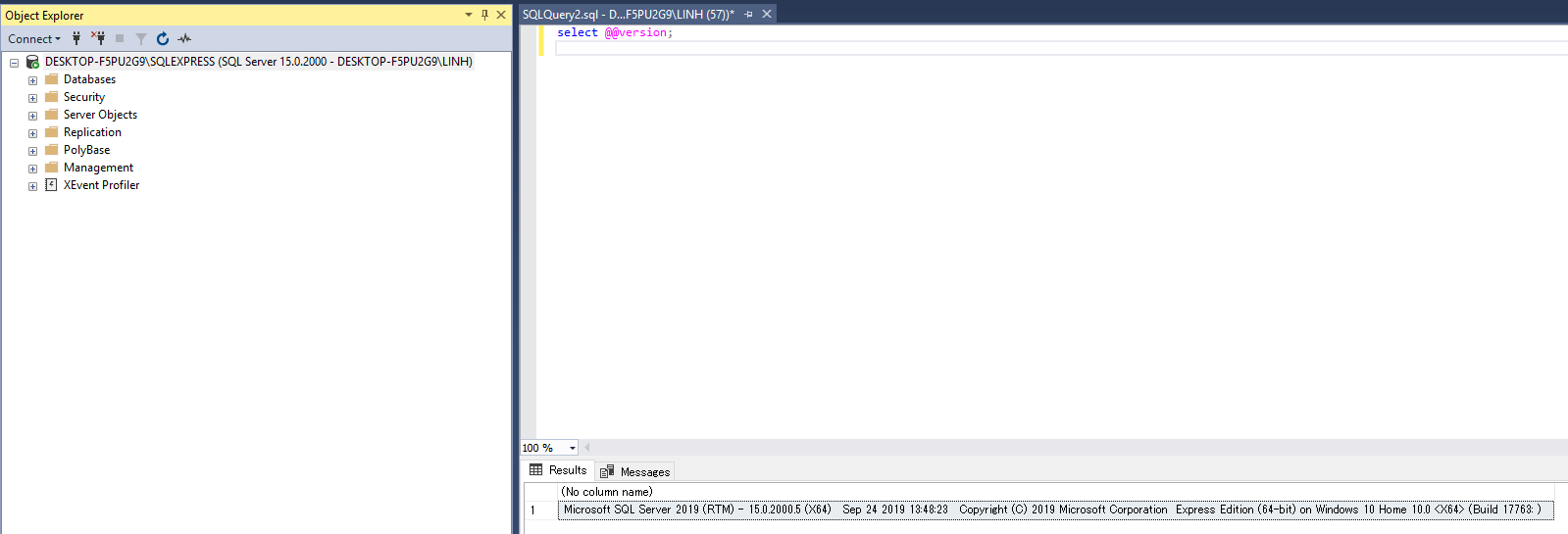


Query/ˈkwɪr.i/

Kích chuột phải vào tên DataBase => New Query

Select @@version;

Execute (F5)

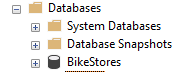


Trang hay học về làm DataBase:

<https://www.sqlservertutorial.net/sql-server-sample-database/>

Chưa biết tạo Table???

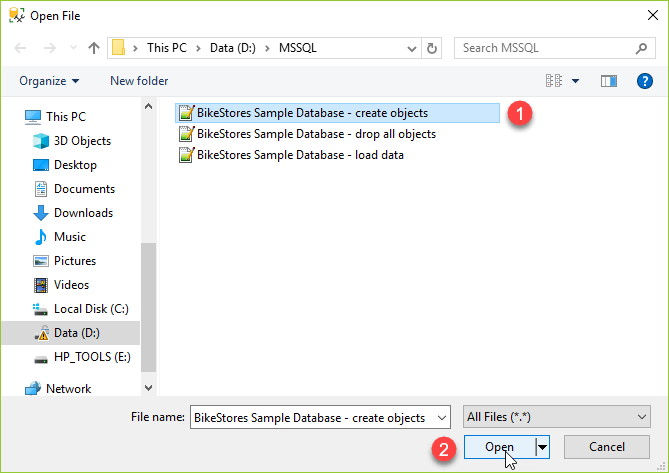
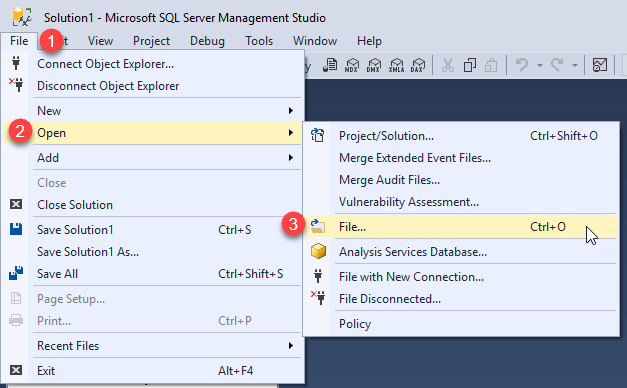
Tạo 1 DataBase mới:



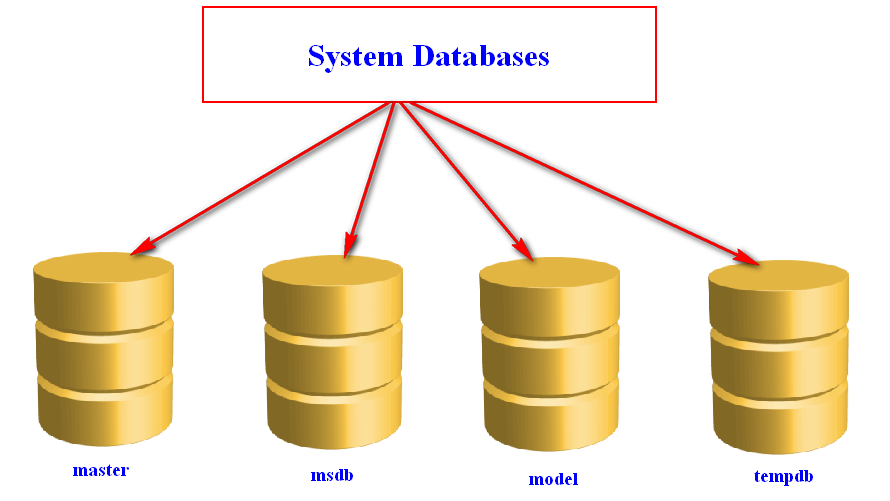
Mở script file đã được tạo:

Để DataBase – script file ở đây:

D:\STUDY\New Technology\DataBase\SQL-Server-Sample-Database



System DataBase:



**Cơ sở dữ liệu Master:**

* Chứa tất cả những thông tin cấp hệ thống (system-level information) : data files, các login account và các thiết đặt cấu hình hệ thống của SQL Server (system configuration settings).

**Cơ sở dữ liệu Tempdb:**

* Chứa tất cả những table hay stored procedure được tạm thời tạo ra trong quá trình làm việc bởi user hay do bản thân SQL Server engine.
* Các table hay stored procedure này sẽ biến mất khi khởi động lại SQL Server hay khi ta disconnect.

**Cơ sở dữ liệu Model:**

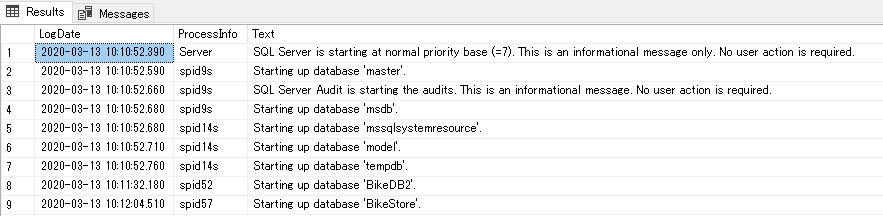
* Database này đóng vai trò như một bảng mẫu (template) cho các database khác. Nghĩa là khi một user database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures…) từ Model database sang database mới vừa tạo.

**Cơ sở dữ liệu Msdb:**

* Được sử dụng cho SQL Server Agent để lập lịch các công việc và các cảnh báo (schedule alerts and jobs).

Chạy lệnh:

sp\_readerrorlog 0,1,'starting'



Kinh nghiệm:

Chịu khó đọc log server.

Sách SQL for Dummy.

Link:

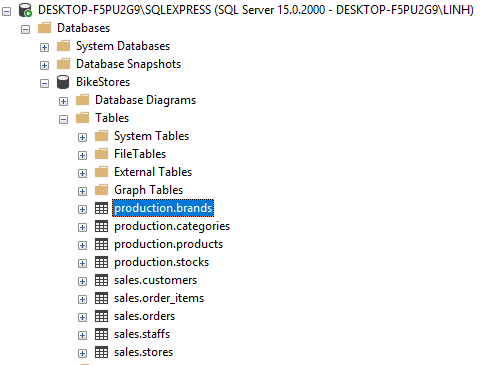
<http://index-of.co.uk/PHP/SQL%20For%20Dummies.pdf>

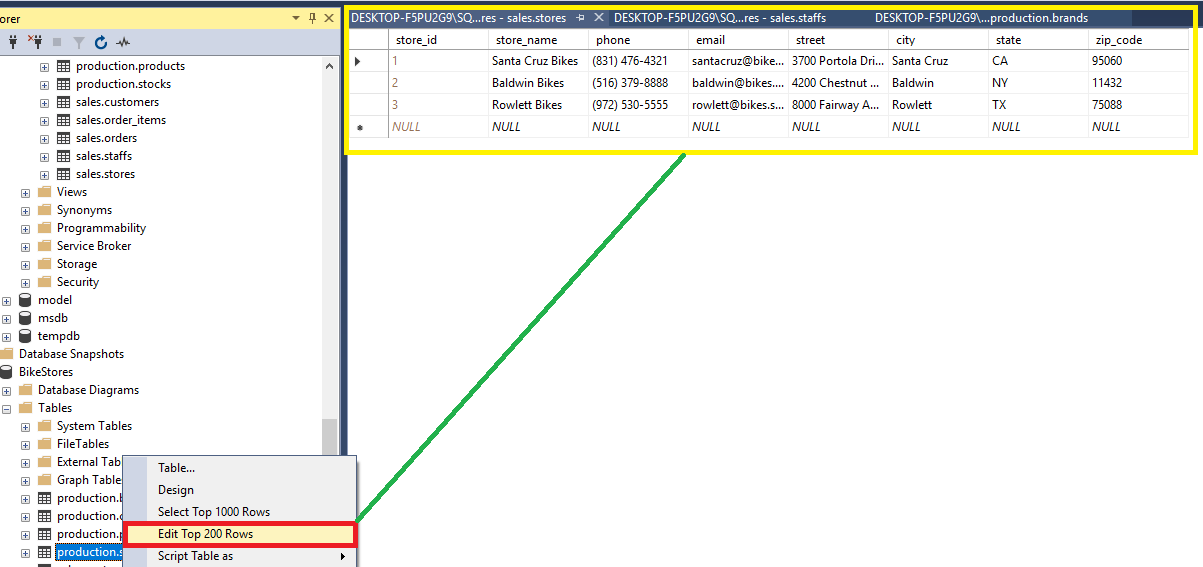
Tham khảo thêm:

<https://www.c-sharpcorner.com/UploadFile/ff0d0f/exploring-system-databases-in-sql-server/>

Load DataBase vào SQL Server.

Đã Load được:





Cấu trúc vật lý của CSDL SQL Server:

* 1 file data chính (.mdf): chứa data , system tables
* 1 hoặc nhiều file data phụ (.ndf):
* 1 transaction log file (.ldf): ghi log của database, chứa thông tin để roll back hoặc roll forward

Tạo dữ liệu cho thằng bảng ở trên:

CREATE TABLE sales.stores (

store\_id INT IDENTITY (1, 1) PRIMARY KEY,

store\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255),

street VARCHAR (255),

city VARCHAR (255),

state VARCHAR (10),

zip\_code VARCHAR (5)

);

**Ví dụ đầy đủ về tạo:**

-- create schemas

CREATE SCHEMA production;

go

CREATE SCHEMA sales;

go

-- create tables

CREATE TABLE production.categories (

category\_id INT IDENTITY (1, 1) PRIMARY KEY,

category\_name VARCHAR (255) NOT NULL

);

CREATE TABLE production.brands (

brand\_id INT IDENTITY (1, 1) PRIMARY KEY,

brand\_name VARCHAR (255) NOT NULL

);

CREATE TABLE production.products (

product\_id INT IDENTITY (1, 1) PRIMARY KEY,

product\_name VARCHAR (255) NOT NULL,

brand\_id INT NOT NULL,

category\_id INT NOT NULL,

model\_year SMALLINT NOT NULL,

list\_price DECIMAL (10, 2) NOT NULL,

FOREIGN KEY (category\_id) REFERENCES production.categories (category\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (brand\_id) REFERENCES production.brands (brand\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE sales.customers (

customer\_id INT IDENTITY (1, 1) PRIMARY KEY,

first\_name VARCHAR (255) NOT NULL,

last\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255) NOT NULL,

street VARCHAR (255),

city VARCHAR (50),

state VARCHAR (25),

zip\_code VARCHAR (5)

);

CREATE TABLE sales.stores (

store\_id INT IDENTITY (1, 1) PRIMARY KEY,

store\_name VARCHAR (255) NOT NULL,

phone VARCHAR (25),

email VARCHAR (255),

street VARCHAR (255),

city VARCHAR (255),

state VARCHAR (10),

zip\_code VARCHAR (5)

);

CREATE TABLE sales.staffs (

staff\_id INT IDENTITY (1, 1) PRIMARY KEY,

first\_name VARCHAR (50) NOT NULL,

last\_name VARCHAR (50) NOT NULL,

email VARCHAR (255) NOT NULL UNIQUE,

phone VARCHAR (25),

active tinyint NOT NULL,

store\_id INT NOT NULL,

manager\_id INT,

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (manager\_id) REFERENCES sales.staffs (staff\_id) ON DELETE NO ACTION ON UPDATE NO ACTION

);

CREATE TABLE sales.orders (

order\_id INT IDENTITY (1, 1) PRIMARY KEY,

customer\_id INT,

order\_status tinyint NOT NULL,

-- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed

order\_date DATE NOT NULL,

required\_date DATE NOT NULL,

shipped\_date DATE,

store\_id INT NOT NULL,

staff\_id INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES sales.customers (customer\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (staff\_id) REFERENCES sales.staffs (staff\_id) ON DELETE NO ACTION ON UPDATE NO ACTION

);

CREATE TABLE sales.order\_items (

order\_id INT,

item\_id INT,

product\_id INT NOT NULL,

quantity INT NOT NULL,

list\_price DECIMAL (10, 2) NOT NULL,

discount DECIMAL (4, 2) NOT NULL DEFAULT 0,

PRIMARY KEY (order\_id, item\_id),

FOREIGN KEY (order\_id) REFERENCES sales.orders (order\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (product\_id) REFERENCES production.products (product\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE production.stocks (

store\_id INT,

product\_id INT,

quantity INT,

PRIMARY KEY (store\_id, product\_id),

FOREIGN KEY (store\_id) REFERENCES sales.stores (store\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (product\_id) REFERENCES production.products (product\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

Sort a result set by one column in ASCENDING ORDER:

Cách tạo 1 Query mới:



Sau đó gõ các dòng lệnh sau:

SELECT

first\_name,

last\_name

FROM

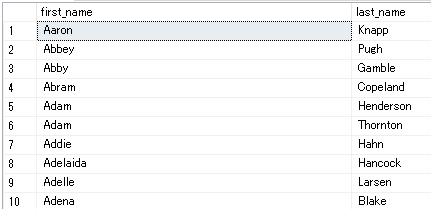
sales.customers (đây là tên bảng)

ORDER BY

first\_name;

(Nếu ORDER BY mà không nói rõ là ASC hay DESC => mặc định là ASC)

Kết quả:



ORDER BY

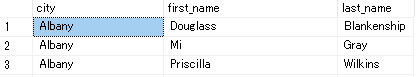
first\_name DESC;



ORDER BY

city,

first\_name DESC;



Note: có thể ORDER BY 1 thằng không có trong lệnh SELECT ở trên (Nhưng phải được defined trong bảng đã tạo rồi)

Sort by the length of the first name:

ORDER BY

LEN (first\_name) DESC;

ORDER BY

list\_price,

product\_name

Sử dụng Offset để bỏ qua 10 dữ liệu đầu và trả về phần còn lại:

ORDER BY

list\_price,

product\_name

OFFSET 10 ROWS;



Sử dụng Offset để bỏ qua 10 dữ liệu đầu và trả về 10 dữ liệu tiếp theo:

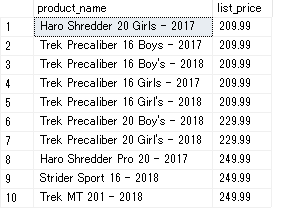
ORDER BY

list\_price,

product\_name

OFFSET 10 ROWS

FETCH NEXT 10 ROWS ONLY;



Lấy 10 most expensive products:

ORDER BY

list\_price DESC, (tên cột)

product\_name (tên cột)

OFFSET 0 ROWS

FETCH NEXT 10 ROWS ONLY;

Note: OFFSET và FETCH được dùng sau ORDER BY để lọc dữ liệu

SELECT TOP: giới hạn số hàng trả về, luôn dùng với ORDER BY

Ví dụ lấy 10 sản phẩm đắt nhất:

SELECT TOP 10

product\_name,

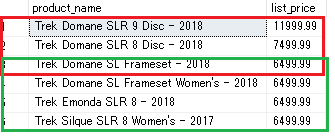
list\_price

TOP WITH TIES : trả về kết quả bao gồm những cái giống với thằng cuối

SELECT TOP 7 WITH TIES

product\_name,

list\_price

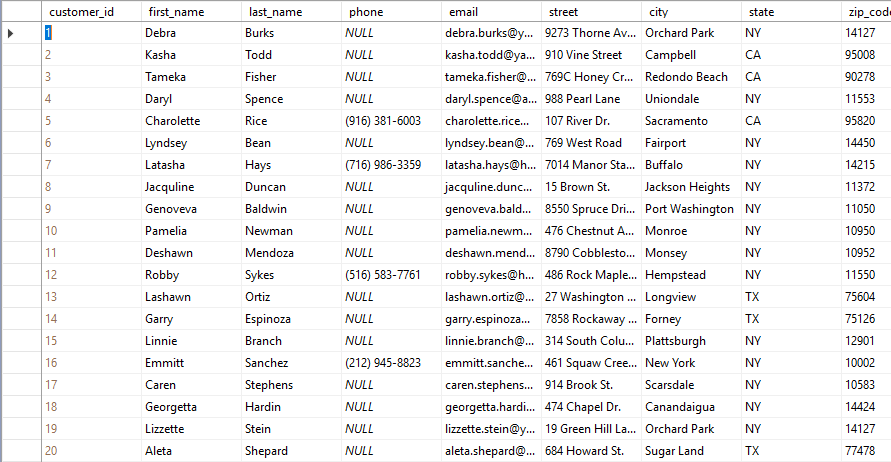


Có 4 thằng giống nhau nên gọi là ties/tai/

Trả về top 1%

SELECT TOP 1 PERCENT

Bảng Sales.customers



SELECT DISTINCT

city

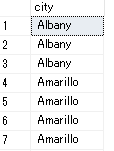
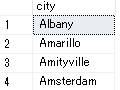
FROM

sales.customers

ORDER BY

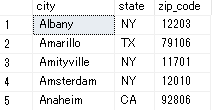
city

Thay vì SELECT city thông thường => dùng SELECT DISTINCT để lọc ra các thành phố (không trùng tên)

 => 

GROUP BY sẽ gần giống với DISTINCT , nhưng lọc ra được cả city, state, zip\_code khác nhau:

SELECT

 city,

state,

zip\_code

FROM

sales.customers

GROUP BY

city, state, zip\_code

ORDER BY

city, state, zip\_code

Cách 2:

SELECT

DISTINCT

city,

state,

zip\_code

FROM

sales.customers;

13 câu lệnh SQL thường dùng (Các câu lệnh nên viết hoa cho dễ đọc):

|  |  |  |
| --- | --- | --- |
| Tên câu lệnh | Giải thích | Ví dụ |
| SELECT  (Basic of Basic) | Đơn giản nhất | SELECT  first\_name,  last\_name  FROM  sales.customers (đây là tên bảng) |
| WHERE | Lấy dữ liệu với lựa chọn cụ thể |  |
| ORDER  (Basic of Basic) | Sắp xếp kết quả trả về | ORDER BY  city,  first\_name DESC; |
| JOIN | Kết hợp các dữ liệu liên quan |  |
| ALIAS | Tạm thời thay đổi tên 1 bảng |  |
| UNION | Lệnh tuyệt vời  Cho phép nối các hàng với nhau |  |
| INSERT | Chèn thêm dữ liệu vào database |  |
| UPDATE | Sau khi chèn thêm dữ liệu, cần phải thay đổi các hàng cụ thể => DÙng UPDATE |  |
| UPSERT | Cho phép cập nhật bản ghi nếu đã tồn tại |  |
| DELETE | Xóa toàn bộ các bản ghi (khá nguy hiểm)  Nếu muốn xóa những bản ghi nhất định thì thêm WHERE |  |
| CREATE TABLE  (Basic of Basic) | Tạo 1 bảng | CREATE TABLE production.categories (  category\_id INT IDENTITY (1, 1) PRIMARY KEY,  category\_name VARCHAR (255) NOT NULL  ); |
| ALTER TABLE | Sửa đổi cấu trúc 1 bảng, ví dụ sửa chuỗi thành số nguyên |  |
| DROP TABLE | Gần giống lệnh DELETE  Xóa mọi bản ghi trong bảng (không xóa 1 bản ghi) => Nguy hiểm, nên thực hiện bằng tay thay vì lệnh |  |

<https://www.sqlservertutorial.net/>

Tham khảo:

<https://quantrimang.com/13-cau-lenh-sql-quan-trong-programmer-nao-cung-can-biet-136595>

Khó khăn còn lại:

Data Connection , chưa kết nối được với Server Local => do không có thằng này nên connection strings không có

Tại sao cần dùng database thay vì text:

* Cấu trúc dữ liệu được chuẩn hóa (dễ quản lí, đọc, viết)
* Nhiều nền tảng hỗ trợ xử lí (support gần từ A đến Z)
* Gắn liền với MVC model

Thực hành:

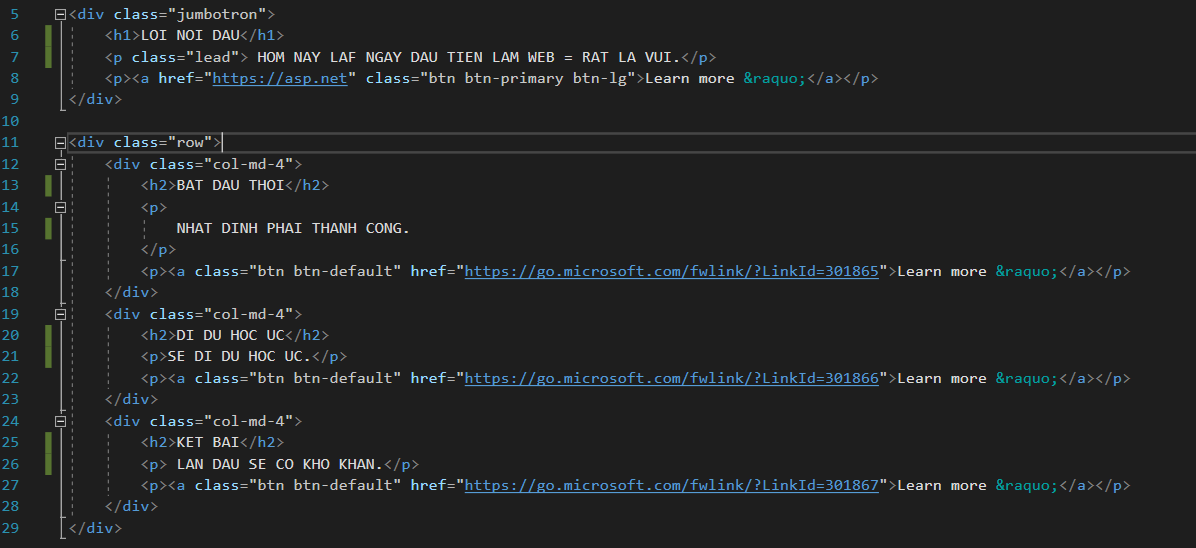
- Xử lí TXT

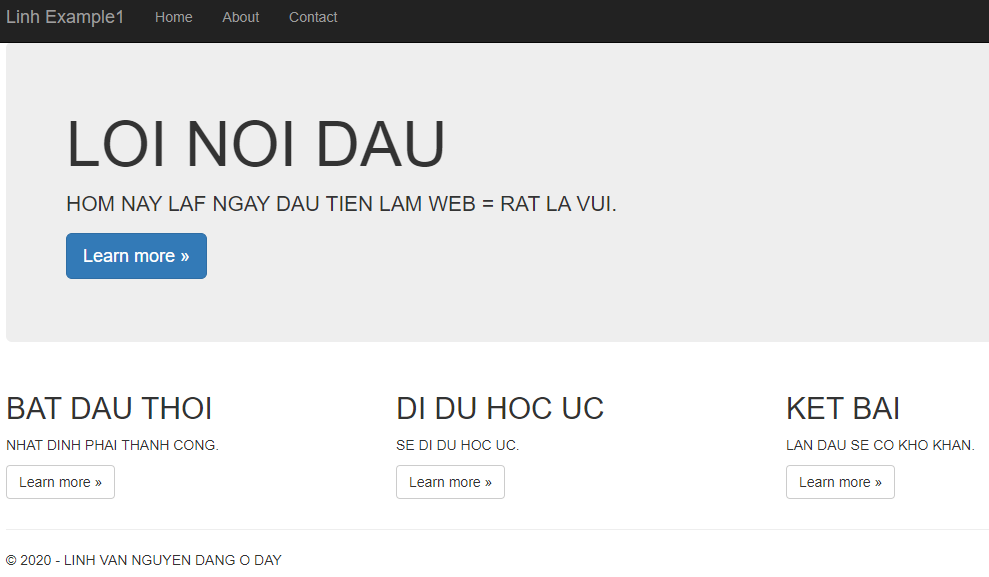
- Xử lí xml

- Xử lí JSON

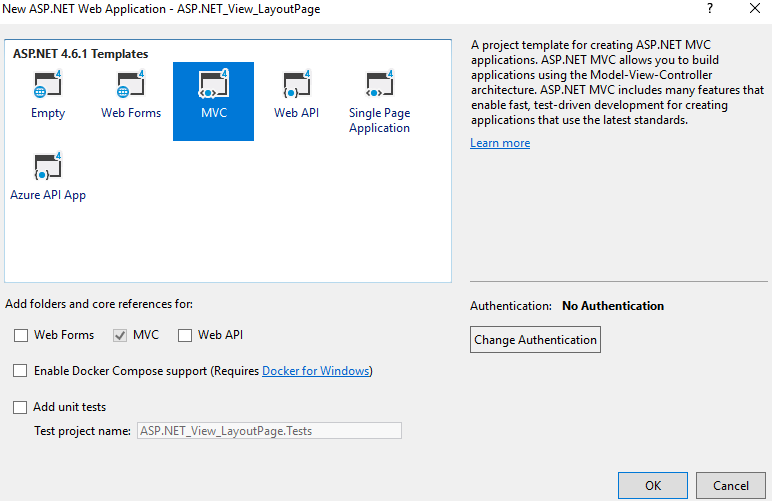
- Xử lí Database

Hiển thị:





Thiết kế giao diện – View – Layout Page



Nội dung trong thẻ \_layout.cshtml:

Header:

<div class="navbar navbar-inverse navbar-fixed-top">

Footer:

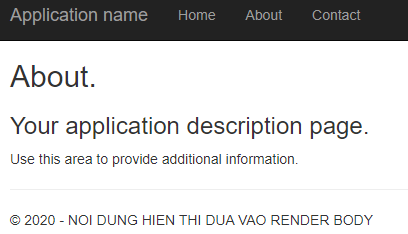
<footer>

<p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>

</footer>

Thuộc tính @RenderBody() để show nội dung khác nhau giữa các Page

@Html.ActionLink: tạo kết nối tới các action trong 1 controller và thực thi action đó



Ví dụ: Kích vào about ra nội dung của About, kích vào Contact ra nội dung của Contact.

ASP.NET - 2002

Aspx : thành phần mở rộng của ASPX

Apache

Apache Là Web Server

Apache trong Linux  
Sau khi cài đặt Apache,ứng dụng sẽ được thêm vào danh sách init.d của hệ thống, do đó có thể tự khởi động cùng với hệ điều hành.