**BÀI 1**

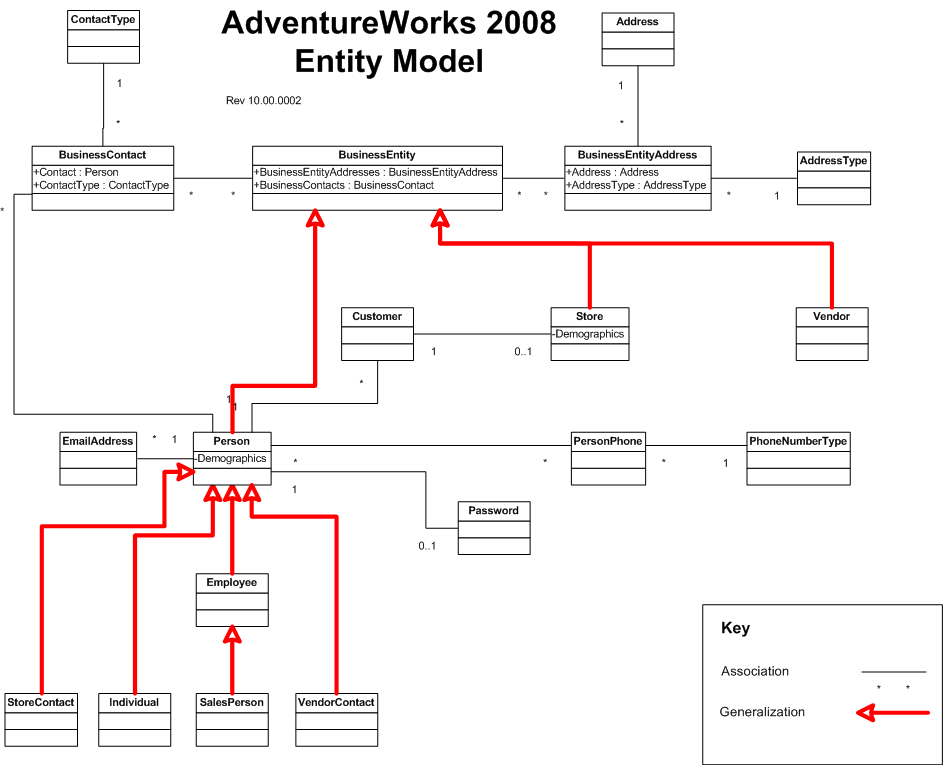
# Mục đích

Làm quen với công cụ hệ quản trị CSDL, tìm hiểu CSDL AdventureWork, nhắc lại về SQL, tinh chỉnh lược đồ CSDL

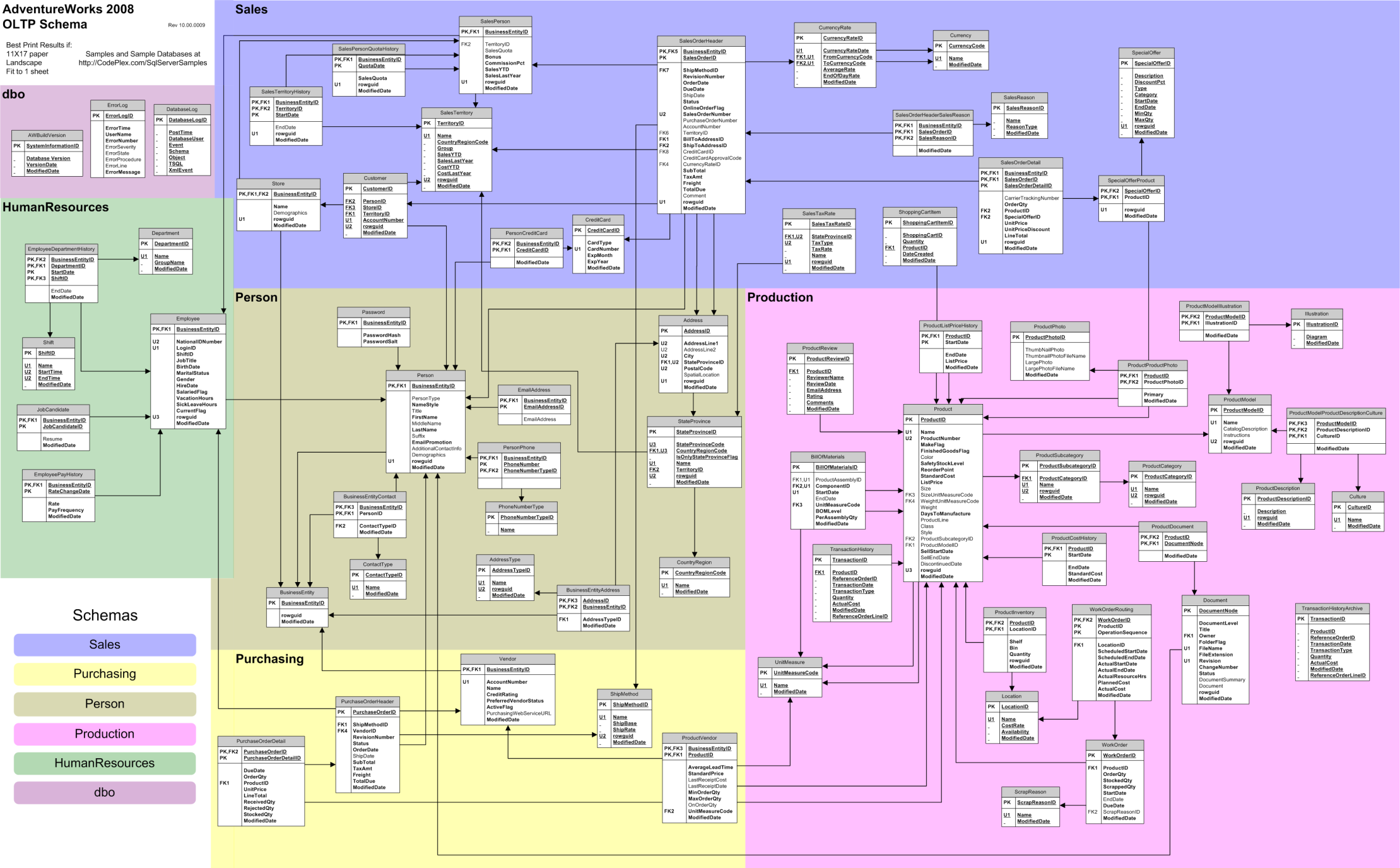
# Giới thiệu chung

# Bài tập thực hành

* 1. Tìm hiểu cơ sở dữ liệu AdventureWork
* Các bảng dữ liệu : http://technet.microsoft.com/en-us/library/ms124438(v=sql.100).aspx



|  |  |
| --- | --- |
| [Address Table](http://technet.microsoft.com/en-us/library/ms124900(v=sql.100).aspx) | [ProductModelIllustration Table](http://technet.microsoft.com/en-us/library/ms124707(v=sql.100).aspx) |
| [AddressType Table](http://technet.microsoft.com/en-us/library/ms124753(v=sql.100).aspx) | [ProductModelProductDescriptionCulture Table](http://technet.microsoft.com/en-us/library/ms124801(v=sql.100).aspx) |
| [AWBuildVersion Table](http://technet.microsoft.com/en-us/library/ms124736(v=sql.100).aspx) | [ProductPhoto Table](http://technet.microsoft.com/en-us/library/ms124678(v=sql.100).aspx) |
| [BillOfMaterials Table](http://technet.microsoft.com/en-us/library/ms124597(v=sql.100).aspx) | [ProductProductPhoto Table](http://technet.microsoft.com/en-us/library/ms124665(v=sql.100).aspx) |
| [Contact Table](http://technet.microsoft.com/en-us/library/ms124714(v=sql.100).aspx) | [ProductReview Table](http://technet.microsoft.com/en-us/library/ms124590(v=sql.100).aspx) |
| [ContactCreditCard Table](http://technet.microsoft.com/en-us/library/ms124463(v=sql.100).aspx) | [ProductSubcategory Table](http://technet.microsoft.com/en-us/library/ms124630(v=sql.100).aspx) |
| [ContactType Table](http://technet.microsoft.com/en-us/library/ms124759(v=sql.100).aspx) | [ProductVendor Table](http://technet.microsoft.com/en-us/library/ms124576(v=sql.100).aspx) |
| [CountryRegion Table](http://technet.microsoft.com/en-us/library/ms124730(v=sql.100).aspx) | [PurchaseOrderDetail Table](http://technet.microsoft.com/en-us/library/ms124638(v=sql.100).aspx) |
| [CountryRegionCurrency Table](http://technet.microsoft.com/en-us/library/ms124778(v=sql.100).aspx) | [PurchaseOrderHeader Table](http://technet.microsoft.com/en-us/library/ms124469(v=sql.100).aspx) |
| [CreditCard Table](http://technet.microsoft.com/en-us/library/ms124632(v=sql.100).aspx) | [SalesOrderDetail Table](http://technet.microsoft.com/en-us/library/ms124498(v=sql.100).aspx) |
| [Culture Table](http://technet.microsoft.com/en-us/library/ms124602(v=sql.100).aspx) | [SalesOrderHeader Table](http://technet.microsoft.com/en-us/library/ms124879(v=sql.100).aspx) |
| [Currency Table](http://technet.microsoft.com/en-us/library/ms124861(v=sql.100).aspx) | [SalesOrderHeaderSalesReason Table](http://technet.microsoft.com/en-us/library/ms124510(v=sql.100).aspx) |
| [CurrencyRate Table](http://technet.microsoft.com/en-us/library/ms124784(v=sql.100).aspx) | [SalesPerson Table](http://technet.microsoft.com/en-us/library/ms124726(v=sql.100).aspx) |
| [Customer Table](http://technet.microsoft.com/en-us/library/ms124893(v=sql.100).aspx) | [SalesPersonQuotaHistory Table](http://technet.microsoft.com/en-us/library/ms124771(v=sql.100).aspx) |
| [CustomerAddress Table](http://technet.microsoft.com/en-us/library/ms124906(v=sql.100).aspx) | [SalesReason Table](http://technet.microsoft.com/en-us/library/ms124560(v=sql.100).aspx) |
| [DatabaseLog Table](http://technet.microsoft.com/en-us/library/ms124872(v=sql.100).aspx) | [SalesTaxRate Table](http://technet.microsoft.com/en-us/library/ms124544(v=sql.100).aspx) |
| [Department Table](http://technet.microsoft.com/en-us/library/ms124543(v=sql.100).aspx) | [SalesTerritory Table](http://technet.microsoft.com/en-us/library/ms124658(v=sql.100).aspx) |
| [Document Table](http://technet.microsoft.com/en-us/library/ms124492(v=sql.100).aspx) | [SalesTerritoryHistory Table](http://technet.microsoft.com/en-us/library/ms124577(v=sql.100).aspx) |
| [Employee Table](http://technet.microsoft.com/en-us/library/ms124432(v=sql.100).aspx) | [ScrapReason Table](http://technet.microsoft.com/en-us/library/ms124845(v=sql.100).aspx) |
| [EmployeeAddress Table](http://technet.microsoft.com/en-us/library/ms124671(v=sql.100).aspx) | [Shift Table](http://technet.microsoft.com/en-us/library/ms124800(v=sql.100).aspx) |
| [EmployeeDepartmentHistory Table](http://technet.microsoft.com/en-us/library/ms124551(v=sql.100).aspx) | [ShipMethod Table](http://technet.microsoft.com/en-us/library/ms124591(v=sql.100).aspx) |
| [EmployeePayHistory Table](http://technet.microsoft.com/en-us/library/ms124708(v=sql.100).aspx) | [ShoppingCartItem Table](http://technet.microsoft.com/en-us/library/ms124720(v=sql.100).aspx) |
| [ErrorLog Table](http://technet.microsoft.com/en-us/library/ms124689(v=sql.100).aspx) | [SpecialOffer Table](http://technet.microsoft.com/en-us/library/ms124485(v=sql.100).aspx) |
| [Illustration Table](http://technet.microsoft.com/en-us/library/ms124603(v=sql.100).aspx) | [SpecialOfferProduct Table](http://technet.microsoft.com/en-us/library/ms124918(v=sql.100).aspx) |
| [Individual Table](http://technet.microsoft.com/en-us/library/ms124643(v=sql.100).aspx) | [StateProvince Table](http://technet.microsoft.com/en-us/library/ms124550(v=sql.100).aspx) |
| [JobCandidate Table](http://technet.microsoft.com/en-us/library/ms124917(v=sql.100).aspx) | [Store Table](http://technet.microsoft.com/en-us/library/ms124477(v=sql.100).aspx) |
| [Location Table](http://technet.microsoft.com/en-us/library/ms124746(v=sql.100).aspx) | [StoreContact Table](http://technet.microsoft.com/en-us/library/ms124615(v=sql.100).aspx) |
| [Product Table](http://technet.microsoft.com/en-us/library/ms124719(v=sql.100).aspx) | [TransactionHistory Table](http://technet.microsoft.com/en-us/library/ms124844(v=sql.100).aspx) |
| [ProductCategory Table](http://technet.microsoft.com/en-us/library/ms124760(v=sql.100).aspx) | [TransactionHistoryArchive Table](http://technet.microsoft.com/en-us/library/ms124470(v=sql.100).aspx) |
| [ProductCostHistory Table](http://technet.microsoft.com/en-us/library/ms124770(v=sql.100).aspx) | [UnitMeasure Table](http://technet.microsoft.com/en-us/library/ms124907(v=sql.100).aspx) |
| [ProductDescription Table](http://technet.microsoft.com/en-us/library/ms124766(v=sql.100).aspx) | [Vendor Table](http://technet.microsoft.com/en-us/library/ms124653(v=sql.100).aspx) |
| [ProductDocument Table](http://technet.microsoft.com/en-us/library/ms124729(v=sql.100).aspx) | [VendorAddress Table](http://technet.microsoft.com/en-us/library/ms124913(v=sql.100).aspx) |
| [ProductInventory Table](http://technet.microsoft.com/en-us/library/ms124455(v=sql.100).aspx) | [VendorContact Table](http://technet.microsoft.com/en-us/library/ms124484(v=sql.100).aspx) |
| [ProductListPriceHistory Table](http://technet.microsoft.com/en-us/library/ms124609(v=sql.100).aspx) | [WorkOrder Table](http://technet.microsoft.com/en-us/library/ms124622(v=sql.100).aspx) |
| [ProductModel Table](http://technet.microsoft.com/en-us/library/ms124873(v=sql.100).aspx) | [WorkOrderRouting Table](http://technet.microsoft.com/en-us/library/ms124747(v=sql.100).aspx) |



* Cấu trúc các bảng dữ liệu : thuộc tính, kiểu dữ liệu, ràng buộc
* Kích thước các bảng dữ liệu : kích thước bản ghi, số lượng các bản ghi,…
  1. Viết câu SQL
* Sử dụng csdl AdventureWorks

USE AdventureWorks

Viết các câu lệnh phù hợp để thực hiện các công việc sau đây:

* Hiển thị chi tiết của tất cả mọi người từ bảng Person.Contact
* Hiển thị Title, FirstName, MiddleName, LastName và EmailAddress từ bảng Person.Contact
* Hiển thị Title, FirstName, LastName như là một chuỗi nối nhằm dễ đọc và cung cấp tiêu đề cho cột tên (PersonName).
* Hiển thị chi tiết địa chỉ của tất cả các nhân viên trong bảng Person.Address
* Liệt kê tên của các thành phố từ bảng Person.Address và bỏ đi phần lặp lại.
* Hiển thị chi tiết của 10 bảng ghi đầu tiên của bảng Person.Address.
* Hiển thị trung bình của tỷ giá (Rate) từ bảng HumanResources.EmployeePayHistory.
* Hiển thị tổng số nhân viên từ bảng HumanResources.Employee
* Đưa ra danh sách các khách hàng có trên 10 đơn hàng
* Đưa ra danh sách các mặt hàng chưa từng được đặt hàng
* Đưa ra
  1. Tinh chỉnh lược đồ CSDL

## 4. Phụ lục

- **PHẦN 1: KIỂU DỮ LIỆU**

Trong phần này chúng ta sẽ xem xét sự khác biệt về kiểu dữ liệu mà các hệ quản trị cơ sở dữ liệu khác nhau hỗ trợ như thế nào:

**1. Kiểu CHUỖI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | **DBMS** | **Cú pháp** | **Diễn giải** | **Ghi chú** |
| 1 | **SQL-99** | CHAR(n) | n số lượng ký tự tối đa cần lưu trữ. Tự lắp đầy kích thước bằng khoản trắng | VARCHAR2(n) - tương tự CHAR(n) nhưng không tự lắp đầy bằng khoản trắng |
| 2 | **SQL Server** | CHAR(n) | tối đa 8000 byte | VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT - kiểu chỉ định VARCHAR(8000) được 8000 byte hoặc VARCHAR(max) được 2^31 byte (2GB) |
| 3 | **Oracle** | CHAR(n) | tối đa 2000 byte | NCHAR, CHAR2, NVARCHAR2 - kiểu CHAR2 và NVARCHAR2 được tối đa 4000 byte |
| 4 | **IBM DB2** | CHARACTER(n) | tối đa 255 ký tự | VARCHAR, CLOB, DBCLOB - kiểu VARCHAR được tối đa 32704 ký tự, CLOB hơn 2 tỉ ký tự, double byte DBCLOB hơn 1 tỉ ký tự |
| 5 | **MySQL** | CHAR(n) | tối đa 255 ký tự | VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT - Kiểu VARCHAR được tối đa 65535 bytes |
| 6 | **PostgreSQL** | CHAR(n) |  | VARCHAR, TEXT |
| 7 | **Sybase** | CHAR(n) |  | VARCHAR, UNICHAR, UNIVARCHAR, NCHAR, NVARCHAR, TEXT, UNITEXT |
| 8 | **Firebird** | CHAR(n) |  | VARCHAR, BLOB SUB\_TYPE TEXT - kiểu VARCHAR tối đa 4000 ký tự Unicode, BLOB 2 tỉ |

**2. Kiểu BOOLEAN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | **DBMS** | **Cú pháp** | **Diễn giải** | **Ghi chú** |
| 1 | **SQL-99** | BOOLEAN | Tùy chọn | Nhận các giá trị TRUE, FALSE ở trạng thái NULL |
| 2 | **SQL Server** |  | Không hỗ trợ | Kiểu dữ liệu có thể thay thế là kiểu BIT (chấp nhận FALSE:0, TRUE:còn lại) |
| 3 | **Oracle** |  | Không hỗ trợ |  |
| 4 | **IBM DB2** |  | Không hỗ trợ |  |
| 5 | **MySQL** |  | Không hỗ trợ | Kiểu dữ liệu thay thế là kiểu BIT(1) |
| 6 | **PostgreSQL** | BOOLEAN | TRUE, FALSE hoặc NULL | Chấp nhận các trực hằng ('true', 't', 'y', 'yes', '1', 'on') và ('false', 'f, 'n', 'no', '0', 'off') |
| 7 | **Sybase** |  | Không hỗ trợ | Kiểu dữ liệu thay thế là kiểu BIT (chấp nhận FALES:0, TRUE: các số nguyên khác 0) |
| 8 | **Firebird** |  | Không hỗ trợ |  |

**3. Kiểu NGÀY, GIỜ & NHÃN THỜI GIAN**

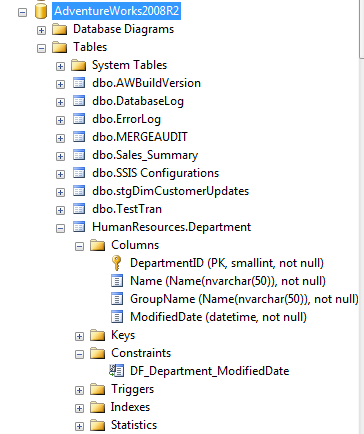
|  |  |  |  |
| --- | --- | --- | --- |
| # | **DBMS** | **Cú pháp** | **Diễn giải** |
| 1 | **SQL-99** | DATE  TIME  TIMESTAMP | TIME WITH TIME ZONE TIMETAMP WITH TIME ZONE YEAR-MONTH INTERVAL DAY-TIME INTERVAL |
| 2 | **SQL Server** | DATE DATETIME DATETIME2 DATETIMEOFFSET SMALLDATETIME TIME | DATE: từ 0001-01-01 tới 9999-12-31 DATETIME: ngày từ 1753-01-01, tới 9999-12-31; giờ từ 00:00:00 tới 23:59:59.997 DATETIME2: ngày từ 0001-01-01 tới 9999-12-31; giờ từ 00:00:00 tới 23:59:59.997 DATETIMEOFFSET: thêm múi giờ từ -14:00 tới +14:00 SMALLDATETIME: ngày chỉ từ 1900-01-01 tới 2079-06-06 TIME: từ 00:00:00.0000000 tới 23:59:59.9999999 |
| 3 | **Oracle** | DATE TIMESTAMP TIMESTAMP WITH TIME ZONE TIMESTAMP WITH LOCAL TIME ZONE INTERVAL YEAR TO MONTH INTERVAL DAY TO SECOND | DATE: từ 4712-01-01 TCN tới 9999-12-31 SCN TIMESTAMP(n): n là số ký số phía sau dấu chấm của thành phần giây có giá trị từ 0 đến 9 TIMESTAMP[(n)] WITH TIME ZONE: bổ sung thêm thành phần múi giờ chuẩn TIMESTAMP[(n)] WITH LOCAL TIME ZONE: bổ sung thêm thành phần múi giờ chuẩn INTERVAL YEAR[(n)] TO MONTH: định nghĩa quãng thời gian gồm bao nhiêu năm bao nhiêu tháng INTERVAL DAY[(n)] TO SECOND[(m)]: định nghĩa quãng thời gian gồm bao nhiêu ngày, giờ, phút, giây |
| 4 | **IBM DB2** | DATE  TIME  TIMESTAMP | DATE: từ 0001-01-01 tới 9999-12-31 TIME: từ 00.00.00 tới 24.00.00 TIMESTAMP: từ 0001-01-01-00.00.00.000000000 tới 9999-12-31-24.00.00.000000000 chính xác tới mức nano second |
| 5 | **MySQL** | DATE DATETIME TIMESTAMP TIME YEAR | DATE: từ '1000-01-01' to '9999-12-31' DATETIME: từ '1000-01-01 00:00:00' to '9999-12-31 23:59:59' TIMESTAMP: từ '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC TIME: từ '-838:59:59' to '838:59:59' YEAR(2), YEAR(4): từ 1901 to 2155 |
| 6 | **PostgreSQL** | DATE TIME TIMESTAMP TIMESTAMP WITH TIME ZONE INTERVAL |  |
| 7 | **Sybase** | DATE TIME SMALLDATETIME DATETIME TIMESTAMP | DATE: từ 0001-01-01 đến 9999-12-31 TIME: từ 00:00:00:000 đến 23:59:59:999 SMALLDATETIME: phần ngày từ 1900-01-01 đến 2079-06-06 DATETIME: phần ngày từ 1753-01-01 đến 9999-12-31 |
| 8 | **Firebird** | DATE TIME TIMESTAMP |  |

**4. Kiểu SỐ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | **DBMS** | **Cú pháp** | **Diễn giải** | **Ghi chú** |
| 1 | **SQL-99** |  |  |  |
| 2 | **SQL Server** |  |  |  |
| 3 | **Oracle** |  |  |  |
| 4 | **IBM DB2** |  |  |  |
| 5 | **MySQL** |  |  |  |
| 6 | **PostgreSQL** |  |  |  |
| 7 | **Sybase** |  |  |  |
| 8 | **Firebird** |  |  |  |

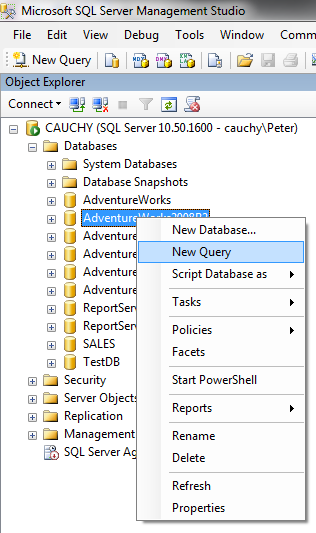
PHẦN 2 Query In a [previous post](http://coldlogics.wordpress.com/2010/11/11/installing-sql-server-2008/), we walked through installing SQL Server 2008 and the Adventure Works databases. Upon successful install, we were able to connect to our database server and expand the installed databases.

Let’s take a moment to explore our surroundings a little bit. Click the plus sign next to AdventureWorks2008R2. You’ll see a number of tables within the database. I have to assume some basic level of acquaintance with databases, but succinctly put, tables are to databases what tracks are to CDs (I wonder how long that analogy will last). If you expand a table, say HumanResources.Department, you find features that are appropriate to tables contained within. The two features you’re likely to be most interested to start with are columns and constraints.

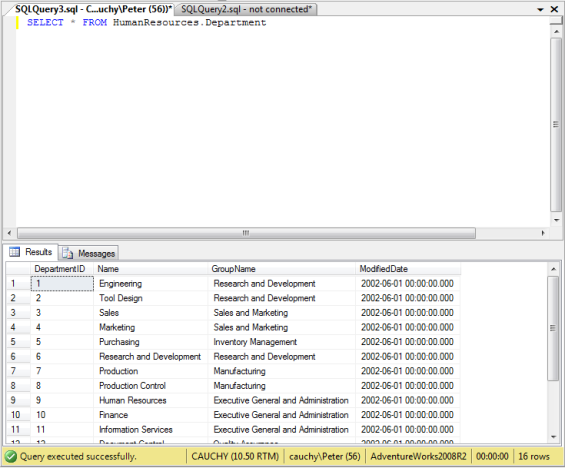


In the above picture, we can observe that we have four columns titled, “DepartmentID”, “Name”, “GroupName”, and, “ModifiedDate”. In addition, we have a default constraint on the ModifiedDate column. Constraints are quite important in databases, you can learn about them more [here](http://blog.sqlauthority.com/2008/09/08/sql-server-%E2%80%93-2008-creating-primary-key-foreign-key-and-default-constraint/) as well as in Books Online (BOL), the help built into SQL Server Management Studio (SSMS). You may find the BOL entry by clicking, “Help” inside of SSMS and clicking, “search”. At that point, just type in “Constraints”, and you’ll find an entry constraints in the section on understanding tables. It’s a good read.

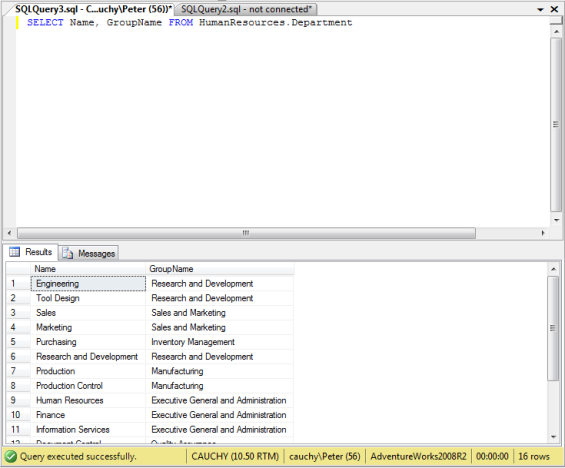
We can assume that this table models departments and divisions inside of the AdventureWorks organization. So let’s begin to query the database. Right click on AdventureWorks2008RS and select, “New Query”.



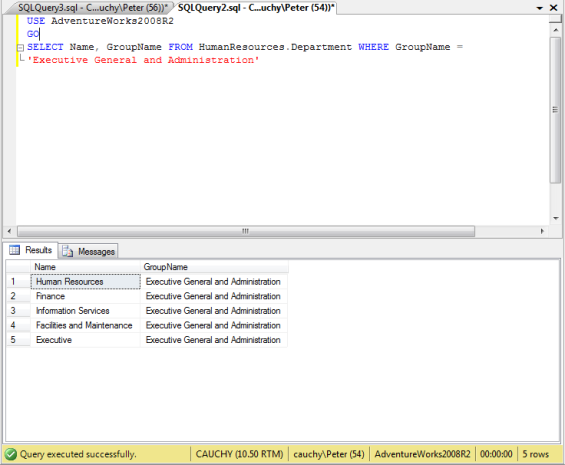
Let’s start by just returning all rows and all columns from with HumanResources.Department. We can do this by typing out in the new query window, “SELECT \* FROM HumanResources.Department”.



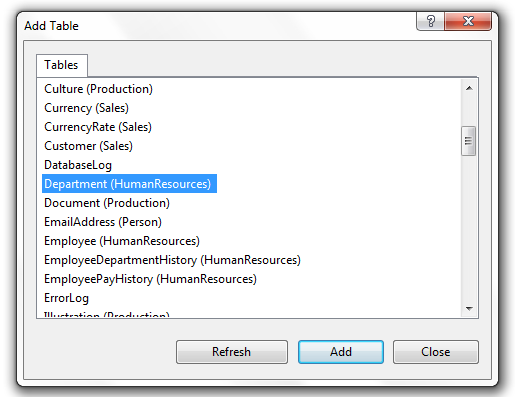
It’s best practice to return only the columns you need. For example, let’s just return the Name and GroupName. To do so let’s type, “SELECT Name, GroupName FROM HumanResources.Department”.



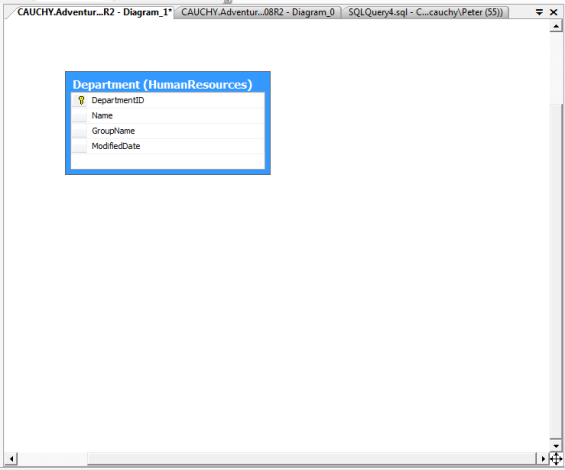
Of course, it’s also best practice to only return the rows you need as well. There’s a lot of reasons for this, sometimes from a managing data perspective, other times from a server load perspective. In general, it just follows from a good principle of living, take only what you need and no more. To return precisely the number of rows you need, we will need use the WHERE clause in our query. Let’s suppose we need the Departments under Executive General and Administration. To do so, let’s use the following query, “SELECT Name, GroupName FROM HumanResources.Department WHERE GroupName = ‘Executive General and Administration”.



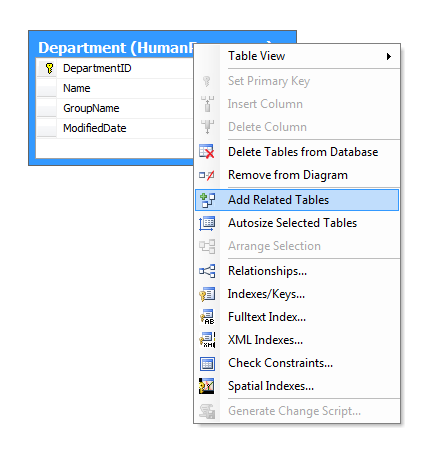
These queries work fine when we only need data out of a single table. Querying data out of a single table is a rare event. SQL Server is a relational database. Intuitively, a [relational database](http://en.wikipedia.org/wiki/Relational_database) is a technology that describes relationships between objects via one or more tables. A common example of this would be employees have departments and vice versa. What if we wanted to know what employees worked in what departments? Well, we need to figure out where the relationships lie within the database. There’s a couple of ways to do this. My favorite is to draw a picture. Many of my mathematics professors in college would say, when in doubt, draw a picture. So let’s draw a picture. Let’s right click on Database Diagrams right above the tables. It’ll ask to configure the database for diagrams. Click okay and once that’s done, you’ll get a window asking what table you want to diagram. Scroll through and find the Departments table.



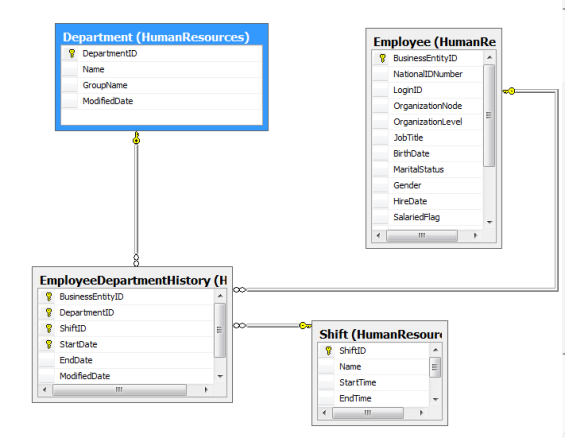
Click, “Add” and then click, “Close. You’ll have a window that should look like the picture below.



Let’s right click on the Departments table and select, “Add Related Tables”.

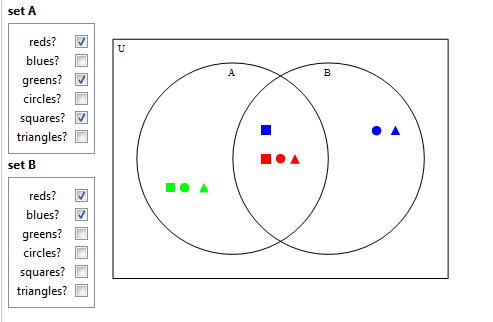


The related table will drop right on top of the first table. Click and drag it off, right click it and select, “Add Related Tables”.



Before we move on, let’s think about what we want. We want to pull information held in two separate tables out. To do this and not have it be a Cartesian product (commonly referred to as a [CROSS JOIN](http://msdn.microsoft.com/en-us/library/ms190690.aspx)), we will need to find the intersecting attribute that unites the data between the tables. This means we must use a [JOIN](http://en.wikipedia.org/wiki/JOIN). In particular, we want the stuff that is only in both tables or sets. To this end, we’re going to employ the INNER JOIN operator.

Let’s look at this screen grab of a handy [Mathematica demonstration](http://demonstrations.wolfram.com/VennDiagramsForTwoSets/) I downloaded:



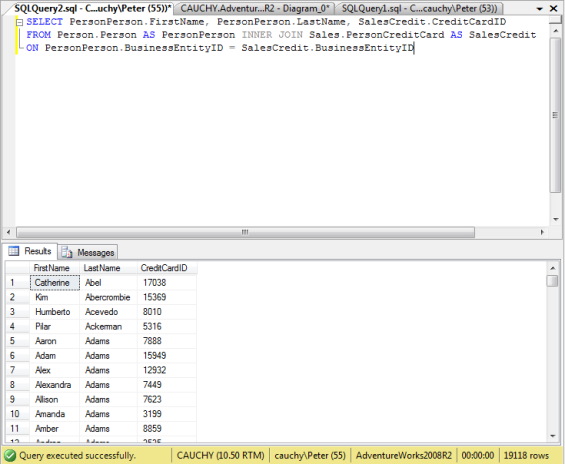
I want the rows from table A that have red circles in table B. The syntax would be as follows: SELECT A.\* FROM A INNER JOIN B ON A.RedCircles = B.RedCircles. That would only give us the elements in table A that have corresponding elements in tables B.

Let’s move back into the world of AdventureWorks. I’m going to want to find out who in the Person table has a company credit card assigned to them and get the ID of that credit card. This is a great opportunity for you to dive back into the Database Diagram tool and see if you can find the column that is common between the two tables.

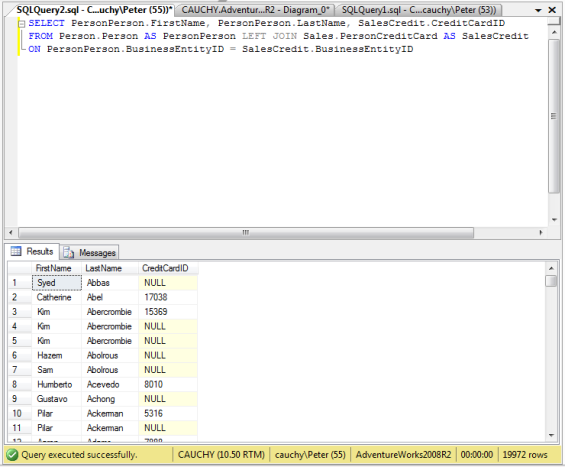
Let’s check the row counts on the two tables by themselves. We want uneven tables so we can see the operator for as it. SELECT \* FROM Person.Person returns 19972 rows. SELECT \* FROM Sales.PersonCreditCard returns 19918 rows. We have our match. To illustrate my point, I will perform an INNER JOIN on those tables.

As an aside, you may be wondering why I use all caps for certain words in my queries. It’s common practice to capitalize T-SQL syntax when writing a query and leaving non-SQL operators and clauses as lowercase. It’s a readability issue but I would suggest strongly in adopting it as a practice.

Returning to the query, let’s get all the data out of the Person.Person table and match it with the Credit card ID of the Sales.PersonCreditCard table.



Note the row count matches the row count of the Sales.PersonCreditCard table. That means there’s rows in the Person.Person table that do not have a matching entry in the Sales.PersonCreditCard table. What if we wanted those rows as well? Enter the LEFT JOIN operator! Let’s make a small adjustment to the query and see what happens.



19972 rows! We’ve seen that number before! Note the duplicate rows and note that rows without a matching BusinessEntityID in the Sales.PersonCreditCard table have NULL in the CreditCardID column. You can also use the RIGHT JOIN operator if you need all rows from the right table in your query. Here’s plenty of documentation on joins [here](http://technet.microsoft.com/en-us/library/ms187518.aspx).

So let’s get back to the original example. We need to get data that spans three tables. Not a problem? Not a problem!

Câu lệnh SQL

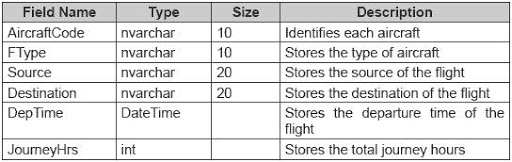
**L2: Làm theo hướng dẫn chi tiết**

**L2: Bài tập Tự làm**

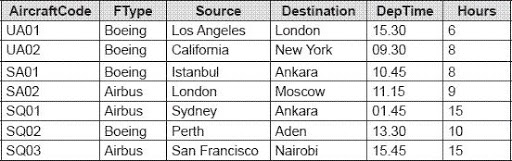
**Tạo CSDL là TravelandTours**

Người điều hành “World Class Travel and Tours” muốn lưu trữ thông tin của tất cả các chuyến bay quan trọng trên thế giới. Người điều hành này còn muốn cung cấp các thông tin sớm nhất cho hành khách.

Q1. Tạo một bảng Flights như hình sau để lưu trữ chi tiết tất cả các chuyến bay.



Q2. Sử dụng câu lệnh INSERT để chèn vào Bảng Flights các thông tin sau:



Q3. Sử dụng câu lệnh thích hợp để hiển thị các tác vụ sau:

* Hiển thị tất cả các bản ghi từ bảng Flights
* Hiển thị AircraftCode, Source, Destination, và DepTime từ bảng Flights
* Chèn một bản ghi có cùng thông tin chi tiết như bản ghi đầu tiên.
* Hiển thị 65% số bản ghi trên bảng Flights
* Hiển thị hai bản ghi đầu tiên từ bảng Flights
* Hiển thị tổng số các chuyến bay từ bảng Flights
* Nếu giá của một giờ bay là $150, sau đó tính giá của mỗi chuyến bay và hiển thị các chi tiết chuyến bay cùng với giá từ bảng Flights.

Gợi ý:  
 \* Sử dụng mệnh đề TOP  
 \* Sử dụng hàm COUNT.

Q4. Tạo bảng Passenger có cấu trúc sau:



Q5. Chèn các chi tiết sau vào bảng Passenger như sau:



Q6. Sử dụng câu lệnh thích hợp để hiển thị các tác vụ sau:

* Hiển thị PassName, Age và Sex từ bảng Passenger
* Chèn một bản ghi cho người có thông tin chi tiết giống với’Stefa Jones’.
* Hiển thị tất cả các bản ghi nhưng không hiển thị các bản ghi trùng nhau.
* Hiển thị tổng số hành khách.
* Hiển thị trung bình tuổi của các hành khách từ bảng Passenger
* Hiển thị 3 bản ghi trên cùng bằng cách khai báo biến integer và sử dụng nó với mệnh đề TOP.
* Tạo một bảng NewPassenger với cùng cấu trúc như bảng Passenger và chèn 3 bản ghi đầu tiên vào bảng NewPassenger từ bảng Passenger.

Q7. Tạo một bảng Reservation chứa đựng cấu trúc sau:

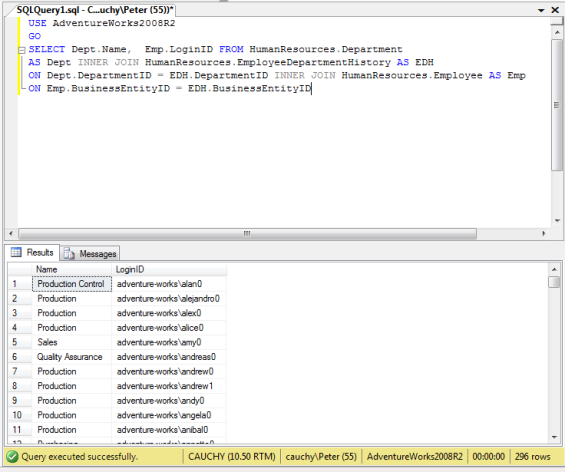


Q8. Chèn các chi tiết sau vào bảng Reservation



Q9. Sử dụng câu lệnh thích hợp để hiển thị các tác vụ sau:

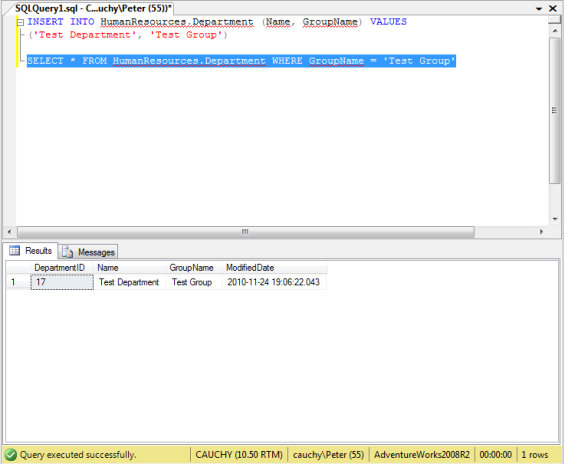
* Hiển thị Aricraft code và Journeydate từ bảng Reservation
* Hiển thị tổng số chỗ ngồi từ bảng Reservation
* Tạo một bảng có tên là NewReservation có cùng cấu trúc như bảng Reservation
* Sử dụng mệnh đề INSERT WITH để chèn các giá trị từ bảng Reservation vào bảng NewReservation.
* Hiệu chỉnh bảng Reservation và thêm một cột PassName với kiểu dữ liệu varchar với kích thước là 25.
* Hiệu chỉnh bảng Reservation và thiết lập trường PNRNO làm khoá chính.



As you can see, we can span INNER JOINs across tables as we need. Nifty.

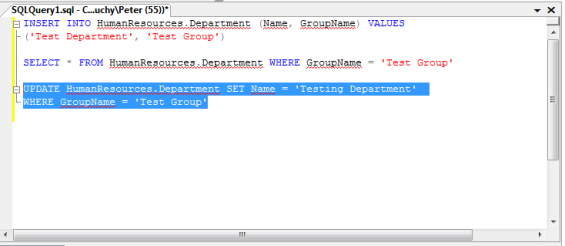
So what next? Inserts, updates, and deletes? Why not? We’ve come this far, let’s just power through this last little bit and then we’ll be on to Mathematica as a front end for querying SQL Server! Understanding how to interact with databases directly will give us a background to all things we do. Fundamentals! It’s all about fundamentals.

We refer to adding new data to a database as an INSERT. The syntax is usually quite basic. You’ll often see the syntax in two forms. INSERT INTO <Table> VALUES (<Value 1>, <Value 2>, … , <Value k>). You’ll use this syntax when you’re inserting a value for every single column in the table. The database will insert each value (assuming that it doesn’t break any rules) in a manner of left to right. Sometimes, when you’re inserting data, you’ll have NULL values for a particular column. If that’s the case, you’ll want to use the following syntax, INSERT INTO <Table> (<Column 1>, <Column 2>, … , <Column k>) VALUES (<Value 1>, <Value 2>, … , <Value k>). The assumption here is that there’s a break in the sequence. Let’s do a quick example.

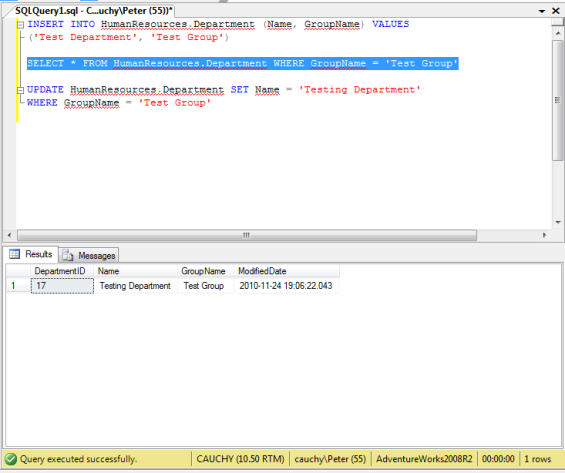


The DepartmentID column has an identity so it auto-incremented when we inserted the Name and GroupName and the default constraint took over and inserted a timestamp since I didn’t have a value in its place. The SELECT query shows that the values took. That’s an insert.

UPDATES follow the same in the same fashion. UPDATE < Table> SET <Column> = <Value> WHERE <Other or same column> = <Value>. The WHERE clause is really important here. Without it being present, every row would be updated. Eek.

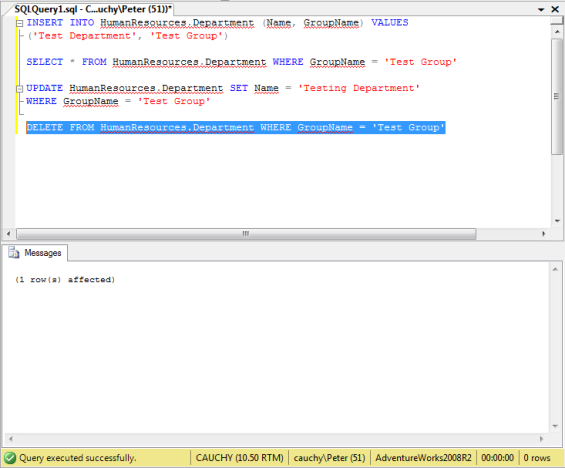


And to verify:



You might note that I have multiple queries and I’ve highlighted one in each of the screenshots I’ve taken. In SSMS (and in many other database management tools), if you highlight a query or parts of a query and attempt to execute that query, SQL Server will only try to execute the portion of the query that’s highlighted. This is very handy for running queries and testing to ensure that you wrote the query accurately.

Finally, there’s DELETE. If you want to delete all rows, just type DELETE FROM <Table>. Bingo! Empty table. Of course, the nuclear option isn’t usually the appropriate option. So you’ll want to use a WHERE clause to ensure that you’re removing the only row(s) you wish to remove. In that case, you do something like this, DELETE FROM <Table> WHERE <Column> = <Value>.



A simple SELECT query returns no results so we know we’ve been successful.

So there it is, the ultra-basics of querying in a nut-shell. Where to next? Well, as promised here’s some fantastic resources that I’ve used and still use as well today.

* <http://msdn.microsoft.com/en-us/library/ms189826.aspx> – MSDN documentation for T-SQL.
* <http://www.java2s.com/> – handy examples in the T-SQL section.
* <http://sqlserverpedia.com/> – great wiki with a wide variety of good information.

There’s a lot of great SQL Server blogs and websites out there as well.

* <http://sqlblog.com/> – SQL Server blogs and forums.
* <http://www.brentozar.com/> – SQL Server MVP, MCM, and [SQLSkills](http://sqlskills.com/) trainer.
* <http://www.sommarskog.se/> – A site maintained by Erland Sommarskog, a SQL Server MVP.
* <http://sqlinthewild.co.za/> – Gail Shaw’s blog, SQL Server MVP and active forum user on [sqlteam.com](http://www.sqlteam.com/).
* <http://www.sqlskills.com/blogs/paul/> – Paul Randal is another SQL Server MVP, wrote the code behind DBCC, and is a co-owner of SQLSkills.com.
* <http://www.sqlskills.com/blogs/kimberly/>- Kimberly Tripp is a SQL Server MVP and SQLSkills.com founder.

I highly suggest getting a book on SQL Server development as well. There’s really no replacing a book. Forums and searches are helpful but that’s information and information is a subset of knowledge. Books contain knowledge. Go to the source.

[About these ads](http://en.wordpress.com/about-these-ads/)

### Share this:

* [Facebook1](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=facebook&nb=1)
* [Reddit](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=reddit&nb=1)
* [Twitter](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=twitter&nb=1)
* [Email](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=email&nb=1)
* [Print](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/#print)
* [StumbleUpon](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=stumbleupon&nb=1)
* [Digg](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=digg&nb=1)
* [Google +1](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=google-plus-1&nb=1)
* [LinkedIn](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?share=linkedin&nb=1)

### Like this:

Like Loading...

### One Response to “Basic SQL Queries using AdventureWorks”

[Comments Feed](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/feed/)

1. [Installing SQL Server 2012 « Cold Logics](http://coldlogics.wordpress.com/2012/04/10/installing-sql-server-2012/)

[...] time to play! If you’ve never played with SQL Server before, you may wish to get started here. Happy querying! Share this:FacebookRedditTwitterEmailPrintStumbleUponDiggLinkedInLike this:LikeBe [...]

[Reply](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/?replytocom=92#respond)

Leave a Reply [Cancel reply](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/#respond)

Top of Form



Enter your comment here...



Fill in your details below or click an icon to log in:

[Gravatar](https://gravatar.com/site/signup/)

Email (required) (Address never made public)

Name (required)



Website



WordPress.com Logo



You are commenting using your WordPress.com account. ( [Log Out](javascript:HighlanderComments.doExternalLogout(%20'wordpress'%20);) / [Change](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/) )

Twitter picture



You are commenting using your Twitter account. ( [Log Out](javascript:HighlanderComments.doExternalLogout(%20'twitter'%20);) / [Change](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/) )

Facebook photo



You are commenting using your Facebook account. ( [Log Out](javascript:HighlanderComments.doExternalLogout(%20'facebook'%20);) / [Change](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/) )

[Cancel](javascript:HighlanderComments.cancelExternalWindow();)

Connecting to %s

Notify me of follow-up comments via email.









Bottom of Form

[Blog at WordPress.com](http://wordpress.com/?ref=footer). Theme: [Iceburgg](http://theme.wordpress.com/themes/iceburgg/) by [Devlounge](http://www.devlounge.net).

<div style="display: none;"><img src="//pixel.quantserve.com/pixel/p-18-mFEk4J448M.gif?labels=%2Clanguage.en%2Ctype.wpcom%2Cposttag.database%2Cposttag.microsoft-sql-server%2Cposttag.select-sql%2Cposttag.sql%2Cposttag.sql-server-2008%2Cposttag.sql-server-management-studio%2Cas" height="1" width="1" alt="" /></div>

Top of Form

Send to Email AddressYour NameYour Email Address

loading[Cancel](http://coldlogics.wordpress.com/2010/11/25/early-encounters-with-data/#cancel)

Post was not sent - check your email addresses!

Email check failed, please try again

Sorry, your blog cannot share posts by email.

Bottom of Form

%d bloggers like this: