Database Fundamentals

NashTech Viet Nam

09th Apr 2020

Warm up - Introductions

- Your role
- Your background and experience in the subject

Course Audience and Prerequisite

- Audience:
  - Fresh Graduate, Junior Engineer
- Prerequisite
  - None

Agenda

- Module 1: Introducing Core Database Concepts
- Module 2: Understanding Data Modeling
- Module 3: Creating Database and Objects
- Module 4: Manipulating and Querying Data
- Module 5: Administering Database

Introducing Core Database Concepts

Nash Tech.

Module Overview

- Databases Concepts
- Normalization
- Referential integrity
- Constraints

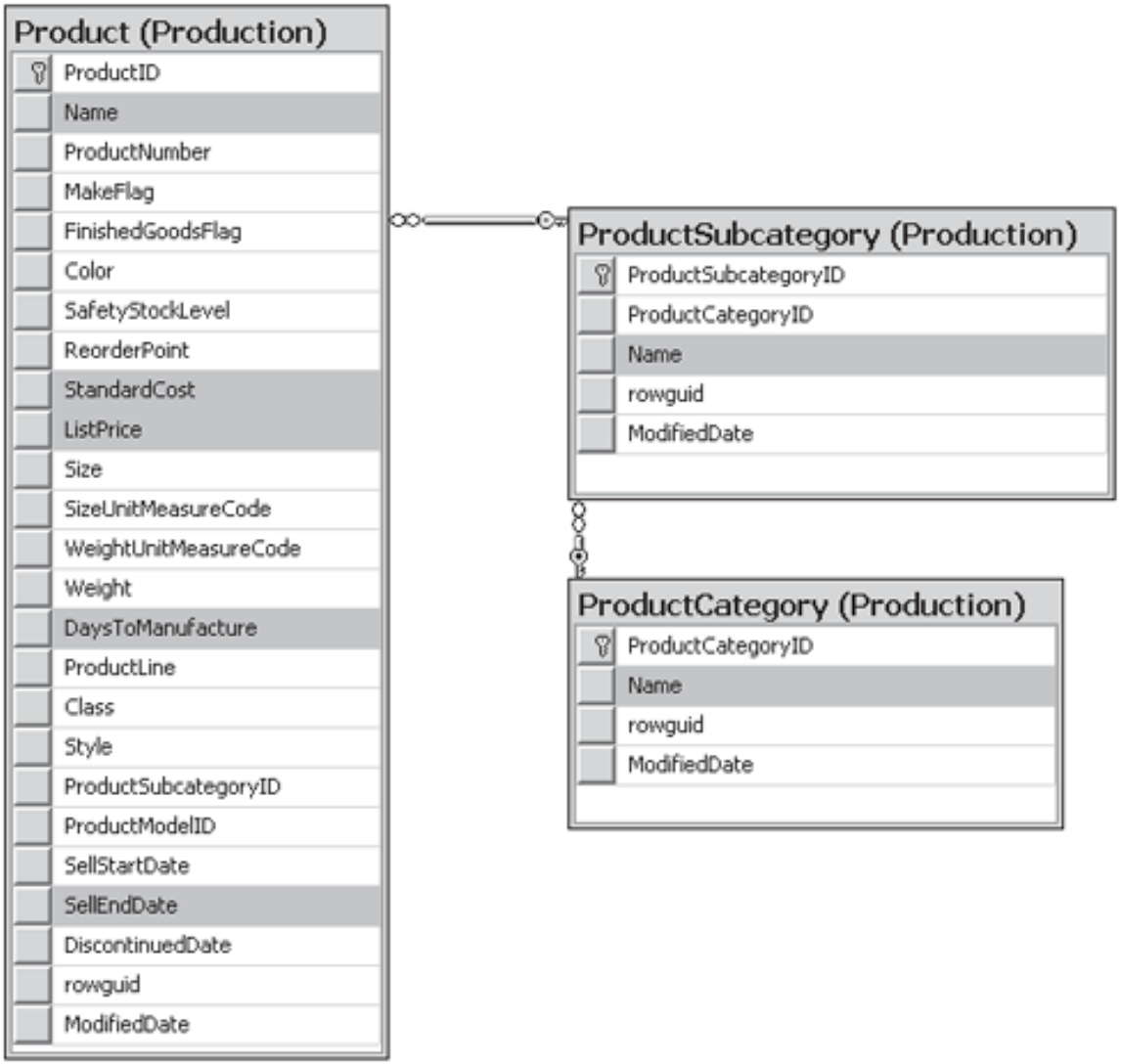
Database

- A ***database (db)*** is an organized collection of data, typically stored in electronic format
  - It allows you to input, manage, organize, and retrieve data quickly
  - Traditional databases are organized by records (rows), fields (columns) stored in tables which are stored in the database files

Relational databases

- A ***relational database*** a collection of tables of data all of which are formally described and organized according to the relational model. Each table must identify a column or group of columns, called the ***PRIMARY KEY***, to uniquely identify each row

Sample relational structure



Referential integrity

- ***Referential Integrity*** (**RI**) is a database concept used to ensure that the relationships between your database tables remains synchronized during data modifications.

- RI can be used to ensure the data is clean, may be helpful in optimizing your database environment and can assist in early detection of errors.

- A combination of **PRIMARY KEY** and **FOREIGN KEY** constraints can be used to help enforce referential integrity of your database.

Database Management System (DBMS)

- **Database Management System** (***DBMS***) is a software system designed to allow the definition, creation, querying, and updating of data stored in databases.

- ***DBMS*** is used by the users to access the data stored in database files. A DBMS is also used to perform administrative tasks on the databases and objects contained within the database.

- A **RDBMS** also provides what DBMS provides but above that it provides relationship integrity.
- In short, we can say **RDBMS = DBMS + Referential Integrity**

Database servers

- Databases are stored on ***database servers*** which are dedicated physical or virtual servers that host the database files and provide high-level performance for users who are accessing the data.

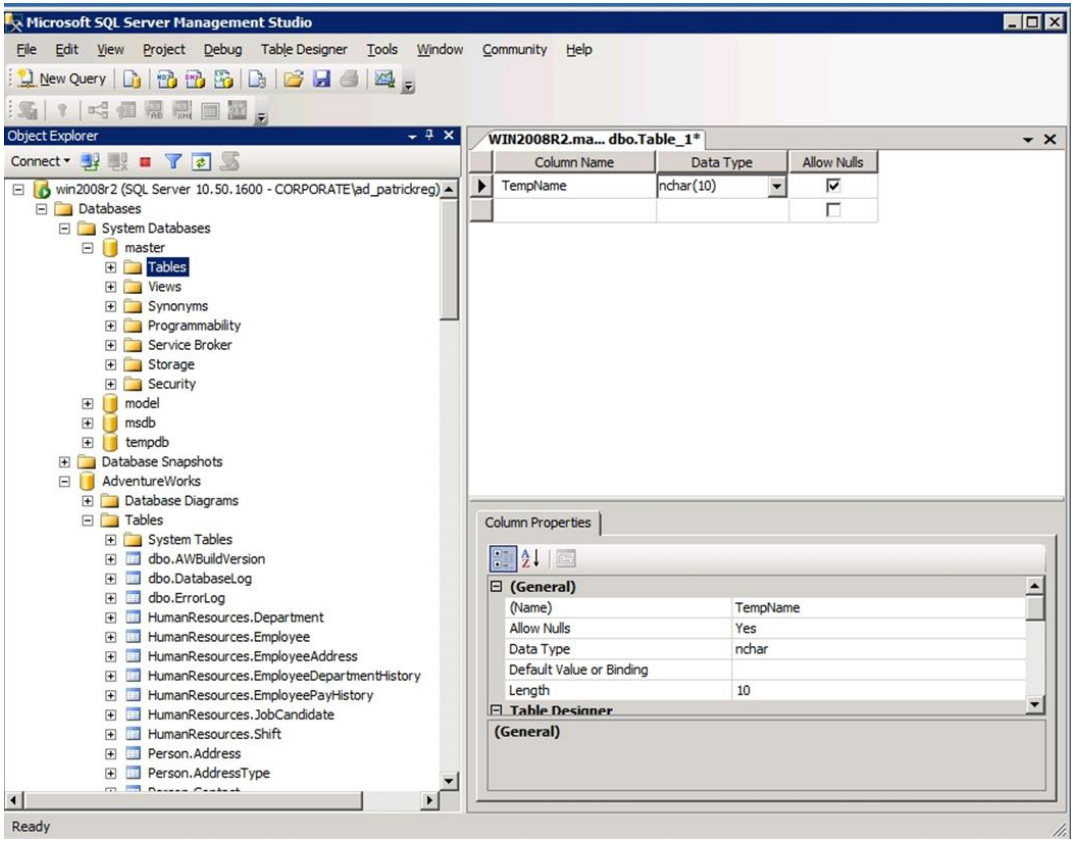
- Database servers contain the DBMS used to manage the data and administer the SQL Server environment.

- A database server can have one default instance and several named instances of SQL Server. A SQL Server instance is a copy of the sqlservr.exe program that runs as a Windows operating system service.

- Often multiple database servers are deployed to provide high availability and improve performance

SQL Server Management Studio (SSMS)

- A graphical user interface (GUI) used to browse, select, and manage the SQL Server instance and any of the objects within that SQL Server instance.



Normalizing a database

- Normalization the process of organizing data in a database that includes creating tables and establishing relationships between the tables
- Process is used to help eliminate redundant data
- Three typical ***normalization forms*** (NFs)
  - 1NF: Eliminate Repeating Groups
  - 2NF: Eliminate Redundant Data
  - 3NF: Eliminate Columns Not Dependent on Key

First normal form (1NF)

- A database is in first normal form if it satisfies the following conditions:
  - Contains only atomic values
  - There are no repeating groups

- Do not use multiple fields in a single table to store similar data

Example of 1ST Normal Form(1NF)

- Un-normalized table:

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

- 1ST Normal Form: No repeating group

TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

Example of normalization

- Un-normalized table

Student#	Advisor	Adv-Room	Class1	Class2	Class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	201-01	211-02	214-01

- First Normal Form: No Repeating Groups

Student#	Advisor	Adv-Room	Class#
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

Second normal form (2NF)

- A database is in second normal form if it satisfies the following conditions:
  - It is in first normal form
  - All non-key attributes are fully functional dependent on the primary key

Example of 2nd Normal Form(2NF)

- Un-normalized table:

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

- 2nd Normal Form: eliminate redundant data

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

Third normal form (3NF)

- A database is in third normal form if it satisfies the following conditions:
  - It is in second normal form
  - There is no transitive functional dependency

Example of 3rd Normal Form(3NF)

- Un-normalized table:

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

- 3rd Normal Form: Third Normal Form: eliminate data not dependent on the key

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Methods for enforcing referential integrity

- There are several methods available in SQL Server to help maintain database integrity:
  - Primary key constraint
  - Foreign key constraint
  - Unique constraint
  - Indexes
  - Triggers

- Any of these methods can be created as a ***composite key*** which is an index or constraint created using more than one column.

PRIMARY KEY constraint

- An important concept of designing a database table is the use of a ***PRIMARY KEY*** — an attribute or set of attributes used to uniquely identify each row

- A table can only have one primary key which is created using a primary key constraint and enforced by creating a unique index on the primary key columns

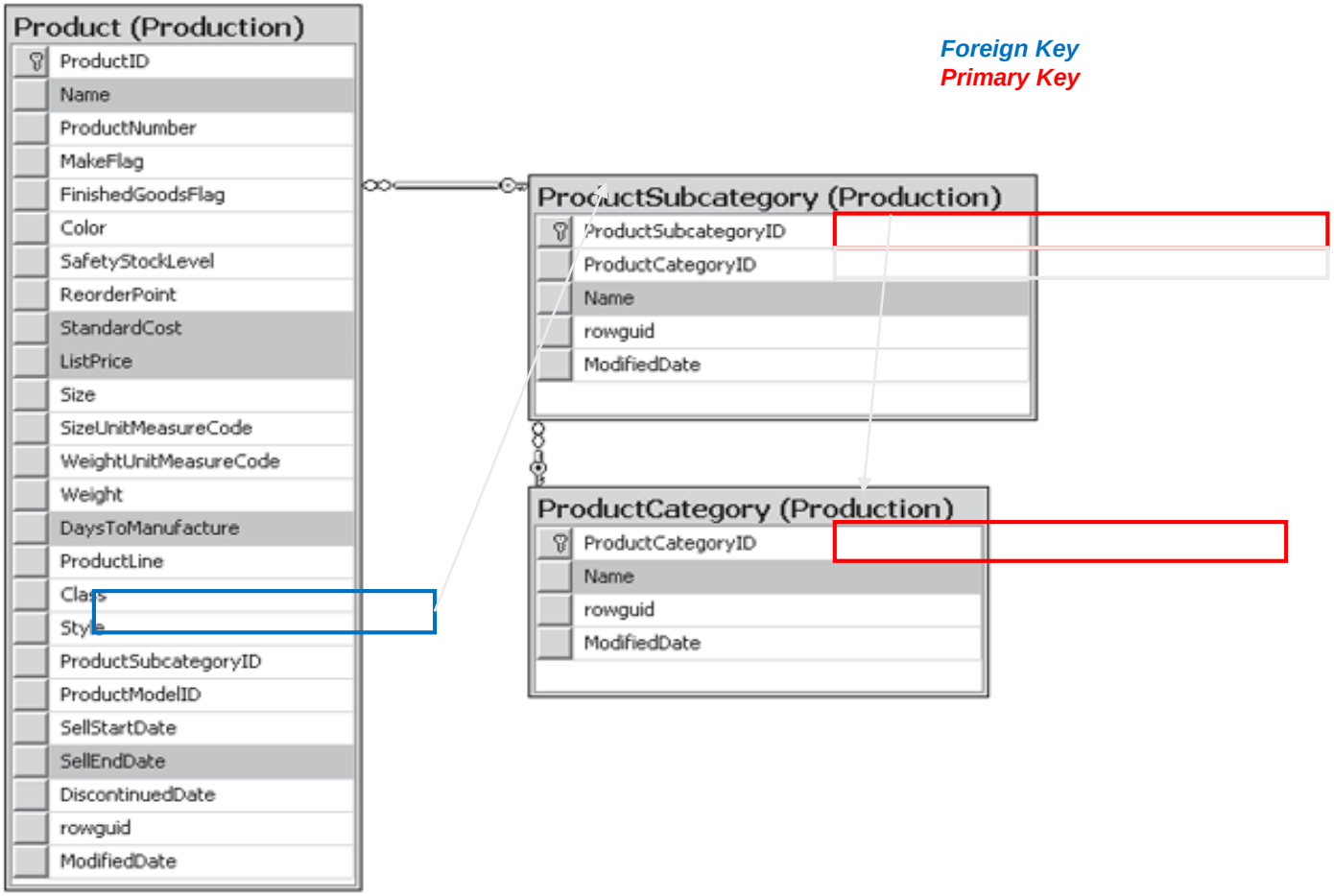
- A column that participates in the primary key constraint cannot accept null values

FOREIGN KEY constraint

- A ***FOREIGN KEY*** is a column or combination of columns that are used to establish a link between data in two tables.

- A foreign key does not have to reference a primary key, it can be defined to reference a unique constraint in either the same table or in another table

Relational structure with keys



Agenda

- Module 1: Introducing Core Database Concepts
- Module 2: Understanding Data Modeling
- Module 3: Creating Database and Objects
- Module 4: Manipulating and Querying Data
- Module 5: Administering Database

Module Overview

- Data Modeling
- Data Model
- Different Data Models
  - Conceptual Data Model
  - Logical Data Model
  - Physical Data Model

Data Modeling

- Process of creating data model for an Information System by applying formal data modeling techniques

- Process used to define and analyze data requirement needed to support business process

- Who involves
  - Data modelers
  - Business Stakeholders
  - Potential users of Information System

What is Data Model?

- Data Model is a collection of conceptual tool for describing data, data relationship, data semantics and consistency constraint

Different Data Model

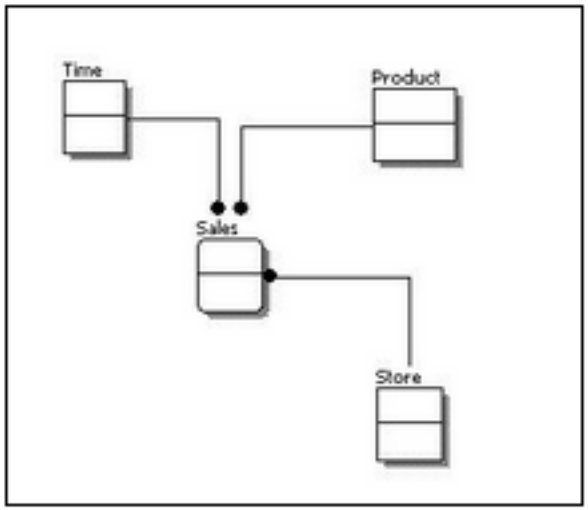
- Conceptual: defines WHAT the system contains
- Logical: describe HOW the system will be implemented, regardless of DBMS
- Physical: describe HOW the system will be implemented using a specific DBMS
- Data Model elements
  - Entity: a real world thing or an interaction between 2 or more real world things
  - Attribute: an atomic piece of information that we need to known about Entity
  - Relationship: How entity depend on each other
    - ONE-TO-ONE
    - ONE-TO-MANY
    - MANY-TO-MANY

Different features in Data Model

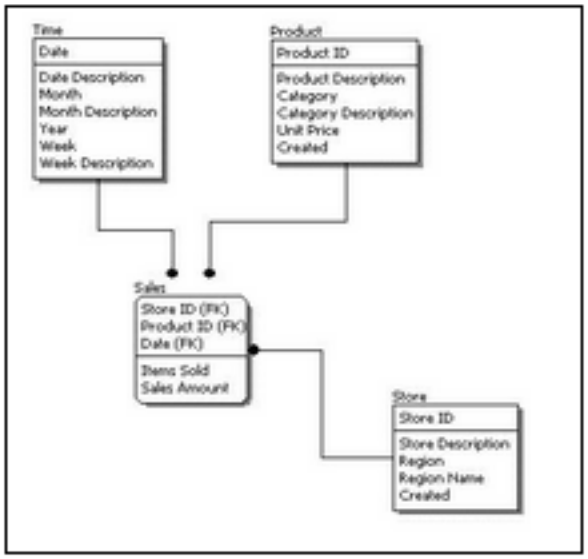
Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	✓
Foreign Keys		✓	✓
Table Names			✓
Column Names			✓
Column Data Types			✓

Conceptual, logical, and physical versions of a single data model

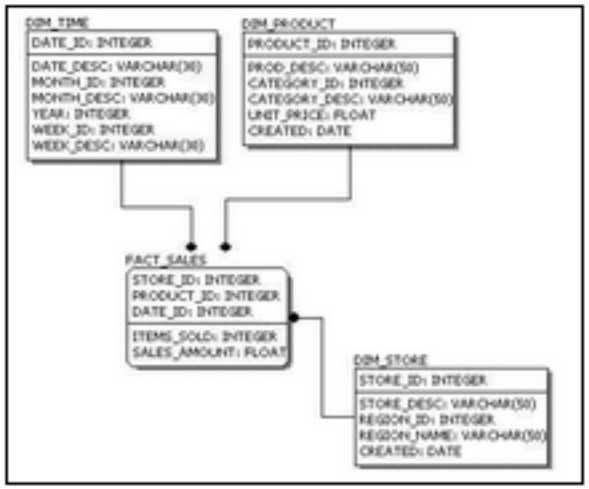
Conceptual Model Design



Logical Model Design



Physical Model Design



Agenda

- Module 1: Introducing Core Database Concepts
- Module 2: Understanding Data Modeling
- Module 3: Creating Database and Objects
- Module 4: Manipulating and Querying Data
- Module 5: Administering Database

Module Overview

- Data types
- Database objects
- DDL statements

Data types

- A ***data type*** is an attribute that specifies the type of data that an object can hold as well as the number of bytes of information that can be stored in the object

- If you have similar data types to choose from but they only differ in byte size, use the data type that has a larger range of values and/or has increased precision

- Exact numeric data types (int, tinyint) are the most common SQL Server data types used to store numeric information.

- Approximate Numerics include precision (p) which is the total number of decimal digits that could be stored, both to the left and right of the decimal point.

Built-in data type categories

- SQL Server built-in data types are organized into the following categories:
  - Exact numerics – (bigint, bit, decimal, int, money, numeric, smallint)
  - Approximate numerics (float, real)
  - Date and time (date, datetime2, datetime, datetimeoffset, time)
  - Character strings (char, varchar, text)
  - Unicode character strings (nchar, ntext, nvarchar)
  - Binary strings (binary, varbinary, image)
  - Other data types (cursor, timestamp, uniqueidentifier, table)
  - Large valued data types (varchar(max), nvarchar(max))
  - Large object data types (text, ntext, image, xml)

Data types

- ***Money*** - used where you'll store money or currency values
- ***Int*** - used to store whole numbers and when performing mathematical computations
- ***Float*** - commonly used in the scientific community and is considered an approximate-number data type
- ***Datetime*** - used to store date and time values in one of many different formats

Data types

- ***Char*** – fixed length non-unicode string data type where *n* defines the string length
- ***Varchar*** – variable length non-unicode string data type that indicates the actual storage size of the data
- ***Bit*** (Boolean) – integer that can have a null, 0 (False), or 1 (True) value

Data types storage size

Data Type	Use/Description	Storage Size
Int	Integer data from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)	4 bytes
Float	Approximate number - 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depends on the value of n
Datetime	Date Range January 1, 1753, through December 31, 9999 Time Range 00:00:00 through 23:59:59.997	8 bytes
Char	Fixed-length, non-Unicode string data. Can be a value from 1 through 8,000	Defined width
Varchar	Variable-length non-Unicode string. Can be a value from 1 through 8,000	Actual length + 2 bytes
Bit	Integer with a value of 0 or 1.	1 byte for every 8 bit columns

Implicit and explicit conversions

- Implicit data type conversions occurs when the SQL Server expression evaluator automatically converts data from one data type to another to complete an operation like a comparison of two values

- Explicit data type conversions require the use of the CONVERT or CAST function to convert data from one data type to another before an operation like a comparison can be completed.

- To convert a numeric value into a character string
  - ✓ **CAST** ($157.27 **AS VARCHAR**(10))

- Not all data types conversions are supported
  - ✓ ***nchar*** cannot be converted to ***image***

- Use CAST instead of CONVERT to adhere to ISO
- Use CONVERT instead of CAST to take advantage of the style functionality

Tables

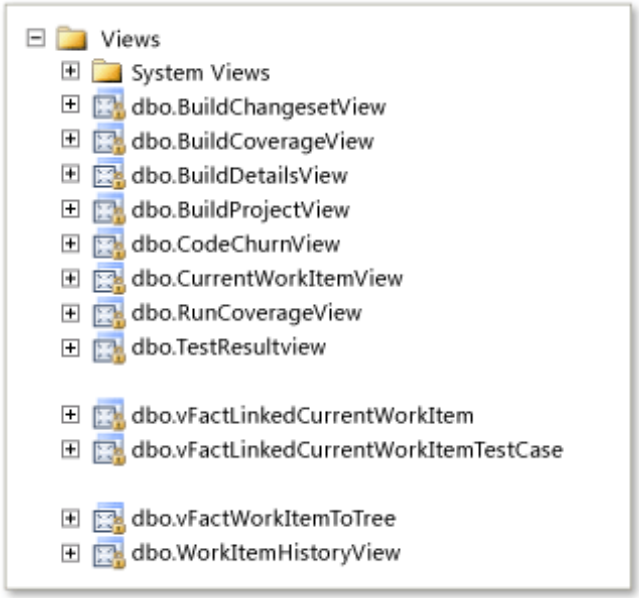
- A ***table*** is a collection of rows and columns that is used to organize information about a single topic. Each row within a table corresponds to a single record and contains several attributes that describe the row.

EmployeeID	LastName	FirstName	Department
100	Smith	Bob	IT
101	Jones	Susan	Marketing
102	Adams	John	Finance

Views

- A ***view*** is simply a virtual table consisting of different columns from one or more tables.

- Unlike a table, a view is stored in the database as a query object; therefore, a view is an object that obtains its data from one or more underlying tables.



Stored procedures

- A ***stored procedure*** is a group of Transact-SQL statements that have been compiled and saved so it can be run several times.

- Parameters can be passed to and returned from a stored procedure so they can be reused with different values.

```
IF (@QuantityOrdered < (SELECT QuantityOnHand
            FROM Inventory
            WHERE PartID = @PartOrdered) )
  BEGIN
  -- SQL statements to update tables and process order.
  END
ELSE
  BEGIN
  -- SELECT statement to retrieve the IDs of alternate items
  -- to suggest as replacements to the customer.
  END
```

User-Defined functions

- ***User-defined functions (udf)*** are routines that takes zero or more parameters, completes an operation, and return the result of the operation as a value.

- There are three types of functions
  - ✓ Scalar – returns a single data value
  - ✓ Table-valued – returns a table data type
  - ✓ System – Provided by SQL Server, cannot be modified

Primary differences between SP and UDF

- Stored Procedures(SP)
  - ✓ Called independently using EXEC statement
  - ✓ Cannot JOIN stored procedures
  - ✓ Can be used to modify SQL Server configuration
  - ✓ Can use nondeterministic functions such as GETDATE()

- User-defined Functions(UDF)
  - ✓ Called from within another SQL statement
  - ✓ Can JOIN UDF's
  - ✓ Cannot be used to modify SQL Server configuration
  - ✓ Always stops execution of T-SQL code if error occurs

Common DDL statements

- CREATE – define new entities
- ALTER – modify existing entities
- DROP – remove existing entities

CREATE statement

- Used to create new entities in SQL Server including some of the most common entities
  - ✓ Database
  - ✓ Table
  - ✓ Default
  - ✓ Index
  - ✓ Login
  - Procedure
  - Trigger
  - View
  - User
  - Role

```
CREATE DATABASE Sales ON ( NAME = Sales_dat, FILENAME =
'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\sales.mdf', SIZE =
10, MAXSIZE = 50, FILEGROWTH = 5 )
LOG ON ( NAME = Sales_log, FILENAME = 'C:\Program Files\Microsoft
SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\salelog.ldf', SIZE
= 5MB, MAXSIZE = 25MB, FILEGROWTH = 5MB ) ;
```

Create new table

- USE SALES
- GO
- --Create new table called Products

```
CREATE TABLE dbo.Products1
(
    ProductID int NULL,
    ProductName varchar(20) NULL,
    UnitPrice money NULL,
    ProductDescription varchar(50) NULL
);
```

ALTER statement

- Used to modify existing entities in SQL Server including
  - ✓ Database
  - ✓ Table
  - ✓ Index
  - ✓ Login
  - ✓ Procedure
  - Trigger
  - View
  - User
  - Role
  - Schema

- ALTER DATABASE Sales
- Modify Name = SalesForecast ;

DROP statement

- Used to delete existing entities in SQL Server including
  - ✓ Database
  - ✓ Table
  - ✓ Index
  - ✓ Login
  - ✓ Procedure
  - Trigger
  - View
  - User
  - Role
  - Schema
- DROP DATABASE SalesForecast

Agenda

- Module 1: Introducing Core Database Concepts
- Module 2: Understanding Data Modeling
- Module 3: Creating Database and Objects
- Module 4: Manipulating and Querying Data
- Module 5: Administering Database

Module Overview

- Introducing DML statements
- Using the SELECT statement
- Modifying data using DML statements

Common DML statements

- SELECT – retrieve data
- INSERT – add data
- UPDATE – modify data
- DELETE – remove data
- BULK INSERT – Import a data file

Using the basic SELECT statement

- The ***SELECT*** statement is used to retrieve rows and columns from a table

  SELECT * FROM tablename

- The SELECT statement requires the name of the table and either the * (retrieves all columns) or specific column names

- To limit the number of rows returned you can include the WHERE clause in the SELECT statement

Sample SELECT statement

```
SELECT BusinessEntityID, JobTitle, Gender
FROM HumanResources.Employee
WHERE BusinessEntityID <= 50
```

- Returns the following results:

```
BusinessEntityID        Title                                   Gender
----------------        -------------                           ---------
1                       Chief Executive Officer                  M
2                       Vice President of Engineering            F
3                       Engineering Manager                      M
4                       Senior Tool Designer                     M
```

Multiple WHERE clauses

- You can combine several WHERE clauses in one query statement to create more specific queries.

```
SELECT BusinessEntityID, Jobtitle, VacationHours
FROM HumanResources.Employee
WHERE JobTitle = 'Design Engineer' AND gender = 'F' AND
HireDate >= '2000-JAN-01'
```

```
SELECT BusinessEntityID, Jobtitle, VacationHours
FROM HumanResources.Employee
WHERE VacationHours  > 80 OR BusinessEntityID <= 50
```

Using the BETWEEN clause

- Retrieving rows within a date range using the BETWEEN clause

```
SELECT BusinessEntityID, Jobtitle, VacationHours
FROM HumanResources.Employee
WHERE VacationHours BETWEEN 75 AND 100
```

Sorting the result set using ORDER By

- Sorting the result set by using the ORDER BY to specify what field to sort by.

```
SELECT BusinessEntityID, Jobtitle, VacationHours
FROM HumanResources.Employee
WHERE VacationHours BETWEEN 75 AND 100
ORDER BY VacationHours
```

- You can sort in descending order by using the DESC clause.

```
SELECT BusinessEntityID, Jobtitle, VacationHours
FROM HumanResources.Employee
WHERE VacationHours BETWEEN 75 AND 100
ORDER BY VacationHours DESC
```

Using the NOT clause

- Write a query to return data that specifies what you don’t want returned

```
SELECT BusinessEntityID, Jobtitle, Gender
FROM HumanResources.Employee
WHERE NOT Gender = ‘M’
```

UNION clause

- The ***UNION*** clause allows you to combine the rows returned from multiple SELECT statements into a single result set

```
SELECT BusinessEntityID, Jobtitle, HireDate
FROM HumanResources.Employee
WHERE JobTitle = 'Design Engineer'
UNION
SELECT BusinessEntityID, Jobtitle, HireDate
FROM HumanResources.Employee
WHERE HireDate BETWEEN '2005-01-01' AND '2005-12-31'
```

EXCEPT and INTERSECT clauses

- The ***EXCEPT*** clause returns distinct values from the left query that are not found on the right query

```
SELECT ProductID
FROM Production.Product
EXCEPT
SELECT ProductID
FROM Production.WorkOrder ;
```

- The ***INTERSECT*** clause returns any distinct values returned by both the query on the left and right sides of intersect operand

```
SELECT ProductID
FROM Production.Product
INTERSECT
SELECT ProductID
FROM Production.WorkOrder ;
```

JOIN clause

- The ***JOIN*** clause allows you to combine related data from multiple tables into one result set

  - ***INNER JOINS*** uses a comparison operator to match rows from two tables based on values in a common column that exists in both tables

  - ***OUTER JOINS*** (left, right, or full) includes rows from one or both tables even if they don't have matching values

  - ***CROSS JOINS*** return all rows from the left table with all rows from the right table. WHERE conditions should always be included.

Aggregate sample

- SQL Server provides **aggregate functions** to assist with the summarization of large volumes of data

```
SELECT COUNT (DISTINCT SalesOrderID) AS UniqueOrders,
    AVG(UnitPrice) AS Avg_UnitPrice,
    MIN(OrderQty)AS Min_OrderQty,
    MAX(LineTotal) AS Max_LineTotal
    FROM Sales.SalesOrderDetail;
```

Inserting data

- You can add a new row to a table using the INSERT statement

```
INSERT INTO Production.UnitMeasure
VALUES (N'FT', N'Feet', '20080414')
```

- You can add multiple rows to a table using the following INSERT statement

```
INSERT INTO Production.UnitMeasure
VALUES (N'FT2', N'Square Feet ', '20080923'),
        (N'Y', N'Yards', '20080923'),
        (N'Y3', N'Cubic Yards', '20080923'
```

- **BULK INSERT** can be used to import a data file into a table with a user-specified format.

Update statement

- The ***UPDATE*** statement is used to modify the data that is already stored in a table

```
UPDATE Sales.SalesPerson
SET Bonus = 6000, CommissionPct = .10, SalesQuota = NULL
WHERE sales.SalesPerson.BusinessEntityID = 289
```

DELETE statement

- The ***DELETE*** statement is used to delete rows from a table

```
DELETE FROM Production.UnitMeasure
WHERE Production.UnitMeasure.Name = ‘Feet’
```

- A **DELETE** statement without a WHERE clause will cause all rows to be deleted

```
DELETE FROM Sales.SalesPersonQuotaHistory;
```

Agenda

- Module 1: Introducing Core Database Concepts
- Module 2: Understanding Data Modeling
- Module 3: Creating Database and Objects
- Module 4: Manipulating and Querying Data
- Module 5: Administering Database

Module Overview

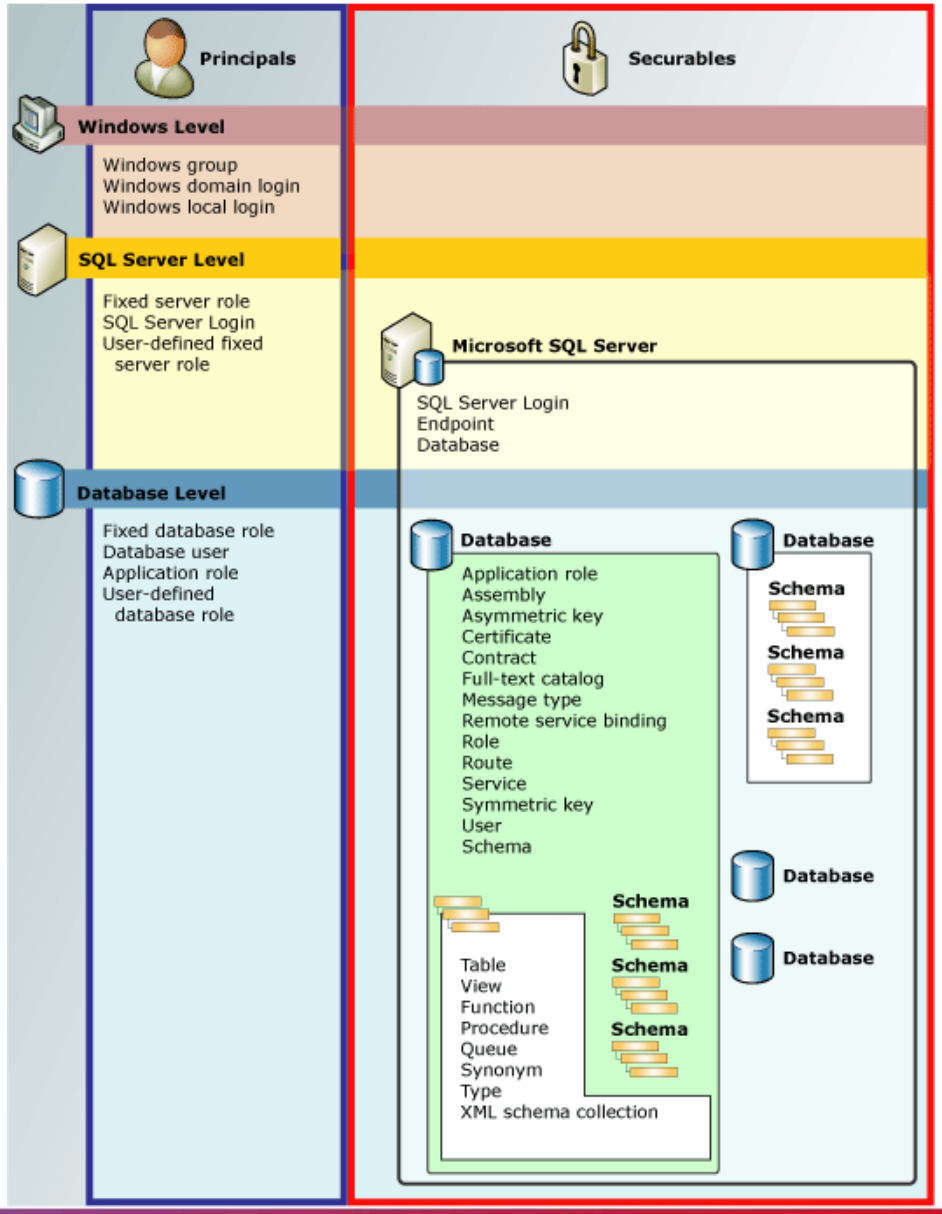
- Understanding SQL Server Security
- Securing SQL Server databases and objects

Database security

Securing your database content is a critical part of a DBA's job. The design, testing, and implementation of security is necessary to ensure that confidentiality is not compromised

- ***Securables*** are the server, database, and objects a database contains
- ***Principals*** are the individuals, groups, and processes granted access to SQL Server
- ***Permissions*** are granted to a principal for every SQL Server securable

Permissions Hierarchy



Logins and accounts

- Three tiered approach to accessing content

  1. SQL Server access - a ***login*** is a security principal that can be authenticated by a secure system to provide a user access to SQL Server

  2. Database access - a ***database user*** is mapped to a SQL login and provides a user or group access to a database

  3. Object access – ***permissions*** are applied at the object level to provide the appropriate access to the objects within the database

Server-level security

- ***Authentication*** is the act of verifying a user or system identity and allowing them to login using:
  - Windows Authentication
  - Mixed-Mode (Windows and SQL logins)

- Logins can be populated into the fixed server roles or in user-defined server roles

Fixed server roles

- SQL Server includes several fixed server roles:
  - **Sysadmin** – perform any activity on the server
  - **Dbcreator** – create, alter, drop, restore databases
  - **Securityadmin** –manage logins and their properties

- You can also create user-defined server roles that have specific permissions applied to the roles

Database-level security

- A database user is a database level security principal that must be mapped to a login at the server level in order for the user to connect to the database
- A login can be mapped to different databases as different users but can only be mapped as one user in each database
- Database users can be populated into the fixed database roles or in a user-defined database role
- All users are automatically members of the *public* database role and cannot be removed

Fixed database roles

- SQL Server includes several fixed database roles
  - db_owner: perform all configuration activities
  - db_datareader: read all data from all user tables
  - db_datawriter: add, delete, or change data

- You can also create user-defined database roles that have specific permissions applied to the roles

Guest logon accounts

- The ***guest*** user account is included in every database and is used by any user who accesses the database but does not have a user account within the database

- The guest user account cannot be dropped but it can be disabled by revoking it's connect permission

```
REVOKE connect FROM guest
```

Managing object permissions

- Permissions to an object can be managed by using the following commands
  - ***Grant -*** provides a level of access to the object
  - ***Deny*** - overrides any grant permission
  - ***Revoke*** - removes the previously assigned permission, regardless of whether it was a deny or grant permission

Object permissions

- ***Object permissions*** are the permissions that allow a user to perform actions on database objects (such as tables, stored procedures, and views):
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - EXECUTE (stored procedures)


Thank you