Capgemini

RDBMS - SQL Server

Lesson 03 : Working with Data Types, DDL ,DML and DCL statements

Lesson Objectives

- In this lesson, you will learn:
  - Working with Data Types
  - Working with SQL Server Schemas
  - Working with DDL, DML ,DCL statements
  - Implementing Data Integrity

Introduction to Data Types in SQL Server

- SQL Server has an extensive list of data types to choose from, including some that are new to respected versions.
- In SQL Server, each column, local variable, expression and parameter has a related data type
- A data type is an element that specifies the kind of data that the item can hold
- SQL Server introduces new additions of data types that can simplify your database development code
- You can also define your own data types in SQL Server
- These User Defined Data Types (UTDs)/ Alias Data Types are based on the system-supplied data types

What are System-Supplied Data Types?

- Integers (whole number)
  - int -2^31 to 2^31-1
  - smallint -2^15 to 2^15 -1
  - bigint -2^63 to 2^63-1
  - tinyint 0 to 255
  - bit 1 or 0
- numeric (Fixed precision)
  - decimal -10^38 +1 to 10^38 –1
  - numeric Equivalent to decimal
- Approximate numeric
  - float -1.79E + 308 to 1.79E + 308
  - real -3.40E + 38 to 3.40E + 38

What are System-Supplied Data Types?

- Monetary (with accuracy to a ten-thousandth of a monetary unit)
  - money -2^63 to 2^63 - 1
  - smallmoney -214,748.3648 through +214,748.3647
- Date and Time
  - datetime
  - smalldatetime
- Character (s)
  - char Fixed-length non-Unicode character data
  - varchar Variable-length non-Unicode data (max length 8,000)

What are System-Supplied Data Types?

- Binary
  - Binary — Fixed-length binary data ,max of 8000 bytes
  - Varbinary — Varying length binary data ,max of 8000 bytes
- Large Objects
  - Image
  - Text
- Unicode Data type storage is two times the byte size
  - Nchar
  - Nvarchar, nvarchar(max)
  - ntext

What are System-Supplied Data Types?

- Other Data Types
  - cursor - A reference to a cursor ,used only in procedures
  - table - A special data type used to store a result set for later processing
  - Sql_variant - Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
  - timestamp - A database-wide unique number that gets updated every time a row gets updated
  - uniqueidentifier - A globally unique identifier (GUID)
  - xml datatype - xml data type lets you store XML documents and fragments

What are System-Supplied Data Types?

- Large value data types:
  - varchar(max)
  - nvarchar(max)
  - varbinary(max)
- For non-unicode data MAX allows up to 2^31 -1 bytes
- For unicode data MAX allows up to 2^30 -1 bytes
- XML
  - To store data in XML Format
  - XML DML for performing insert, update and deletes on the XML based columns

SQL Server - New Data Types

- SQL Server Date Data Types
  - DATE
  - TIME
  - DATETIME2
  - DATETIMEOFFSET

SQL Server - Date Data Types

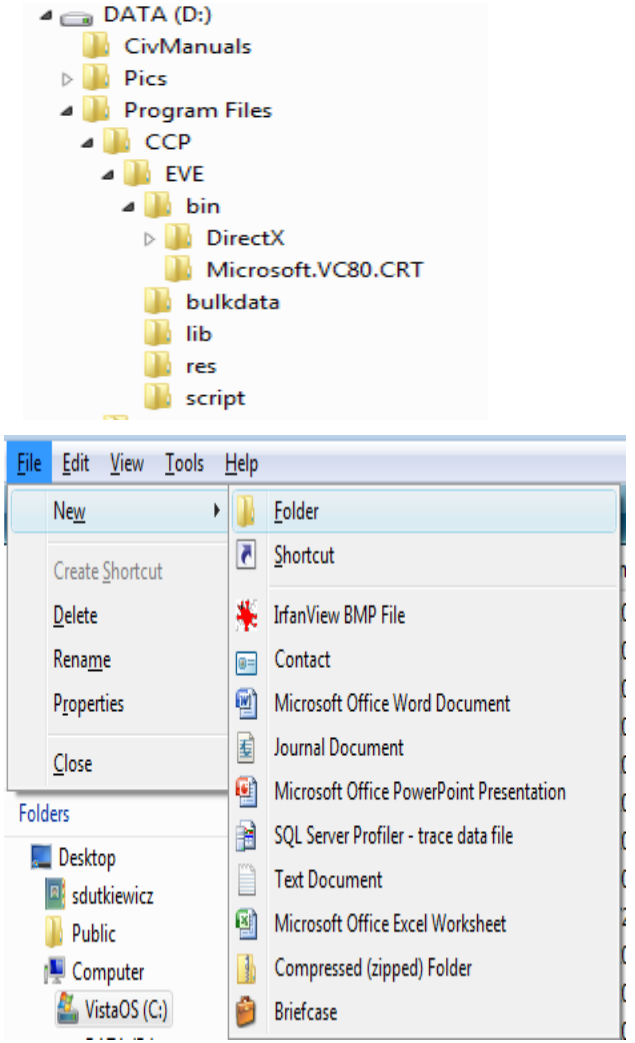
Data Type	Format	Range	Storage Size (bytes)
time	hh:mm:ss[.nnnnnnn]	00:00:00.0000000 through 23:59:59.9999999	5
date	YYYY-MM-DD	0001-01-01 through 9999-12-31	3 bytes, fixed
smalldatetime	YYYY-MM-DD hh:mm:ss	1900-01-01 through 2079-06-06	4 bytes, fixed.
datetime	YYYY-MM-DD hh:mm:ss[.nnn]	1753-01-01 through 9999-12-31	8 bytes
datetime2	YYYY-MM-DD hh:mm:ss[.nnnnnnn]	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999	6 bytes for precisions less than 3; 7 bytes for precisions 3 and 4. All other precisions require 8 bytes.
datetimeoffset	YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+|-]hh:mm	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 (in UTC)	10 bytes, fixed is the default with the default of 100ns fractional second precision

Spatial Data Type – What is Spatial Data?

- Information about the location and shape of a geometric object:
  - Store locations
  - Sales regions
  - Customer sites
  - Area within a specific distance of a location
- Two types:
  - Planar (or Euclidean) data for coordinate points on a flat, bounded surface. Distances are measured directly between points
  - Geodetic (or ellipsoidal) data for latitude and longitude points on the surface of the Earth. Distances are measured taking into account the curvature of the ellipsoidal surface

SQL Server 2008 – Hierarchyid Data Types

- SQL Server 2008 introduces a new system-provided data type "Hierarchyid" to encapsulate hierarchical relationships
- We can use hierarchyid as a data type to create tables with a hierarchical structure
- It is designed to store values that represent the position of nodes of a hierarchal tree structure
- This type is internally stored as a VARBINARY value
- Examples where the hierarchyid type makes it easier to store and query hierarchical data include the following:
  - Organizational structures
  - A set of tasks that make up a larger projects (like a GANTT chart)
  - File systems (folders and their sub-folders)
  - A graphical representation of links between web pages



Introduction to Alias Data Types (User Defined Data Types)

- An alias data type is a user defined custom data type based on system-supplied data type
- In SQL Server alias data types offers a great deal of data consistency while working with various tables or databases
- You should create an alias data type when :
  - You need to define a commonly used data element with specific format
  - Database tables must store the same type of data in a column
  - These columns have identical data type, length & nullability
- Example

```
CREATE TYPE EmailAddress
FROM varchar(30) NOT NULL;
```

Dropping Alias Data Types

- You can drop alias data type by using Object Explorer in SQL Server Management Studio
- You can also use DROP TYPE Transact-SQL Statement to remove the alias data type from a database
- The DROP TYPE statement removes an alias data type from the current database
- You cannot drop a user-defined type until all references to that type have been removed
- Example

```
DROP TYPE EmailAddress
```

Demo

- Creating alias data types (UDTs)
- Dropping alias data types (UDTs)

Data Types in SQL Server –Best Practices

- If column length varies, use a variable data type
- Use tinyint appropriately
- For numeric data types, commonly use decimal
- If storage is greater than 8000 bytes, use text or image
- Use money for currency
- Do not use float or real as primary keys
- If your character data type columns use the same or a similar number of characters consistently, use fixed length data types (char, nchar)
- Choose the smallest numeric or character data type required to store the data
- If you are going to be using a column for frequent sorts, consider an integer-based column rather than a character-based column

What is a Database Schema?

- A Database Schema is a way to logically group SQL Server objects such as tables, views, stored procedures etc
- A schema is a distinct namespace, a container of SQL Server objects, distinct from users those who have created those objects
- In earlier releases of SQL Server, an object's namespace was determined by the user name of its owner
- An object owned by a database user is no longer tied to that user
- By introducing Schemas in database objects can now be manipulated independently of user
- A SQL Server user can be defined with default schema
- If no default schema is defined, sql server will assume dbo as the default schema

Creating Database Schema

- Example : Creating Database Schema

```
Use AdventureWorks
GO
CREATE SCHEMA Sales
GO
```

- Example : Assigning a Default Schema

```
ALTER USER Anders WITH DEFAULT_SCHEMA = Sales
```

Demo

- Creating Database Schema

Create Table

- Once you define all the data types in your database, you can create the Tables to store your business data
- The Table is the most important and central object of any RDBMS
- These tables may be temporary, lasting only for a single interactive SQL Session
- They also can be permanent, lasting weeks or for months
- Creating a database table involves :
  - Naming the table
  - Defining the columns
  - Assigning properties to the column
- In SQL Server, you can use CREATE TABLE Transact-SQL Statement for creating table

Creating Table

- Given a Table Structure for Employee

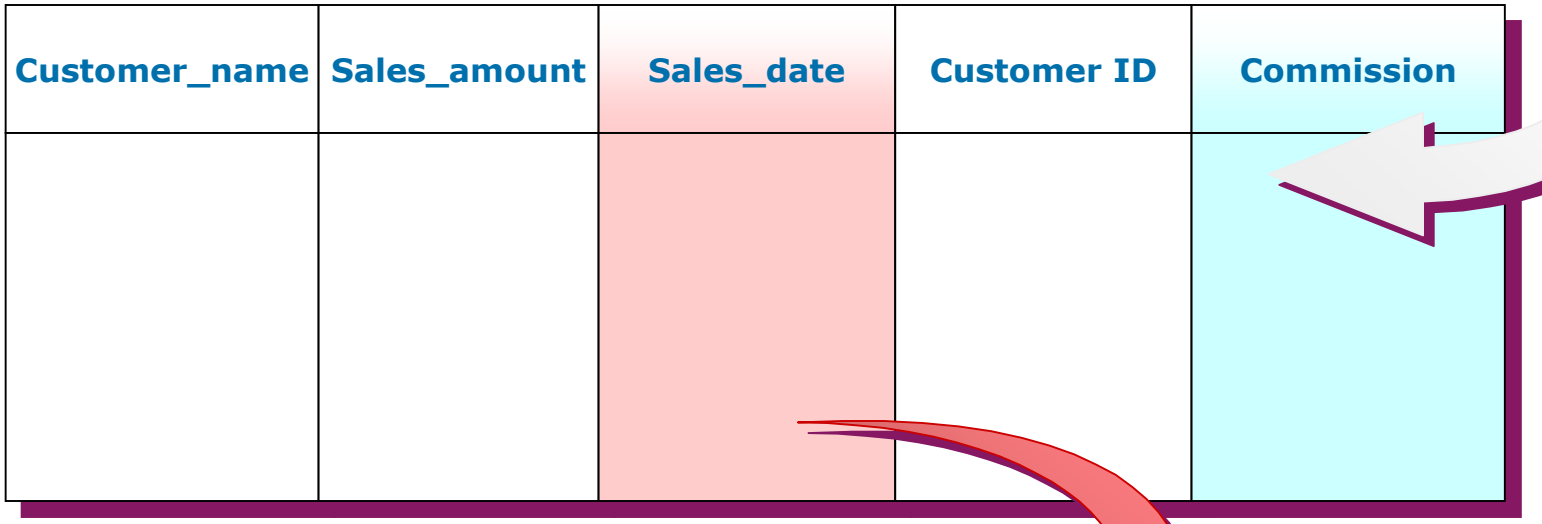
Column Name	Data Type	Nullability
Employee_Code	Int	NOT NULL
Employee_Name	Varchar(40)	NOT NULL
Employee_DOB	Datetime	NOT NULL
Employee_EmailID	Varchar(20)	NULL

```
CREATE TABLE Employee
(
    Employee_Code      int             NOT NULL,
    Employee_Name      varchar(40)     NOT NULL,
    Employee_DOB       datetime        NOT NULL,
    Employee_EmailID   varchar(20)     NULL
)
```

Adding and dropping a column

ADD

```
ALTER TABLE CategoriesNew
ADD Commission money null
```



DROP

```
ALTER TABLE CategoriesNew
DROP COLUMN Sales_date
```

Special Types of Columns in SQL Server

- Computed Columns

```
CREATE TABLE Marks
( Test1 int, Test2 int,
 TestAvg AS (Test1 + Test2)/2 )
```

- Identity Columns

```
CREATE TABLE Printer
(PrinterId int IDENTITY (1000, 1) NOT NULL)
```

- Uniqueidentifier Columns

```
CREATE TABLE Customer ( CustomerID uniqueidentifier NOT
NULL)
INSERT INTO Customer Values (NewID())
```

SEQUENCE

- SQL Server 2012 introduces Sequence as database object.
- Sequence is an object in each database and is similar to IDENTITY in its functionality.
- It can have start value, incrementing value and an end value defined in it.
- It can be added to a column whenever required rather than defining an identity column individually for tables.
- Limitations of using the Identity property can be overcome by the introduction of this new object SEQUENCE.

SEQUENCE

- CREATESEQUENCE SQ1 AS INT
- START WITH 100
- INCREMENT BY 25
- MINVALUE 100
- MAXVALUE 200
- CYCLE
- CACHE 10
- GO
- SELECT NEXT VALUE FOR SQ1

SEQUENCE

```
CREATE SEQUENCE mySeq
START WITH 10
INCREMENT BY 5;

SELECT NEXT VALUE FOR mySeq;
SELECT NEXT VALUE FOR mySeq;
SELECT NEXT VALUE FOR mySeq;

create table myTable
(sid int, Sname varchar(15))

INSERT INTO myTable(sid, sname)
VALUES (NEXT VALUE FOR mySeq, 'Tom’);

select * from myTable

INSERT INTO myTable(sid, sname)
VALUES (NEXT VALUE FOR mySeq, 'Moody');
```

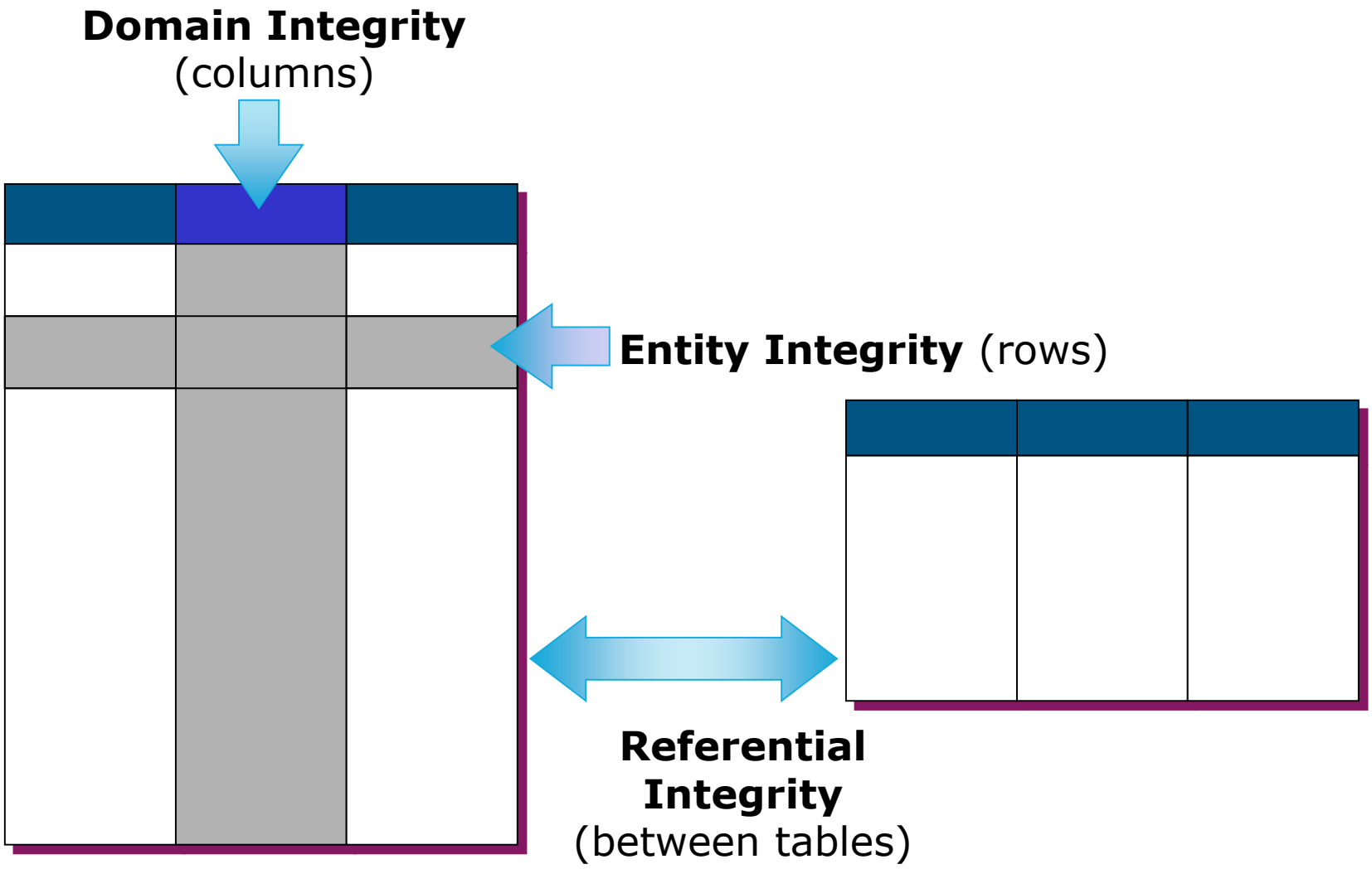
Demo

- Creating Table

Enforcing Data Integrity

- The term Data Integrity refers to correctness and completeness of the data in a database
- To preserve the consistency and accuracy of data, every RDBMS imposes one or more data integrity constraints in a database
- These constraints restricts the wrong or invalid data values that can be inserted or updated in a database
- How and what kind of integrity is enforced in the database depends on the type integrity being enforced
- Enforcing data integrity ensures quality of the data in the database
- The quality of data refers to :
  - Accuracy
  - Completeness
  - Consistent
  - Correct

Types of Data Integrity in SQL Server



Ways of implementing Data Integrity in SQL Server

- Declarative Data Integrity
  - Criteria defined in object definitions
  - SQL Server enforces automatically
  - Implement by using constraints, defaults, and rules
  - Simplest integrity check
- Procedural Data Integrity
  - Implement by using triggers, stored procedures & application code
  - Data validation criteria will be defined in script
  - Allows more advanced data integrity checks

Determining the type of constraint to be used

Type of integrity	Constraint type
Domain	DEFAULT
	CHECK
	REFERENTIAL
Entity	PRIMARY KEY
	UNIQUE
Referential	FOREIGN KEY
	CHECK

Creating Constraints

- Use CREATE TABLE or ALTER TABLE
- Can Add Constraints to a Table with existing Data
- Can Place Constraints on Single or Multiple Columns
  - Single column, called column-level constraint
  - Multiple columns, called table-level constraint

Consideration for using constraints

- Can be changed without recreating a table
- Require error-checking in applications and transactions
- Verify existing data

Types constraints

- DEFAULT Constraints
- CHECK Constraints
- PRIMARY KEY Constraints
- UNIQUE Constraints
- FOREIGN KEY Constraints
- Cascading Referential Integrity

DEFAULT constraints

- Apply only to INSERT statements
- Only one DEFAULT constraint per column
- Cannot be used with IDENTITY property or rowversion data type
- Allow some system-supplied values

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT DF_contactname DEFAULT 'UNKNOWN'
FOR ContactName
```

CHECK constraints

- Are used with INSERT and UPDATE statements
- Can reference other columns in the same table
- Cannot:
  - Be used with the rowversion data type
  - Contain subqueries

```
USE Northwind
ALTER TABLE dbo.Employees
ADD
CONSTRAINT CK_birthdate
CHECK (BirthDate > '01-01-1900' AND BirthDate < getdate())
```

PRIMARY KEY constraints

- Only one PRIMARY KEY constraint per table
- Values must be unique
- Null values are not allowed

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT PK_Customers
  PRIMARY KEY NONCLUSTERED (CustomerID)
```

UNIQUE constraints

- Allow One Null Value
- Allow Multiple UNIQUE Constraints on a Table
- Defined with One or More Columns

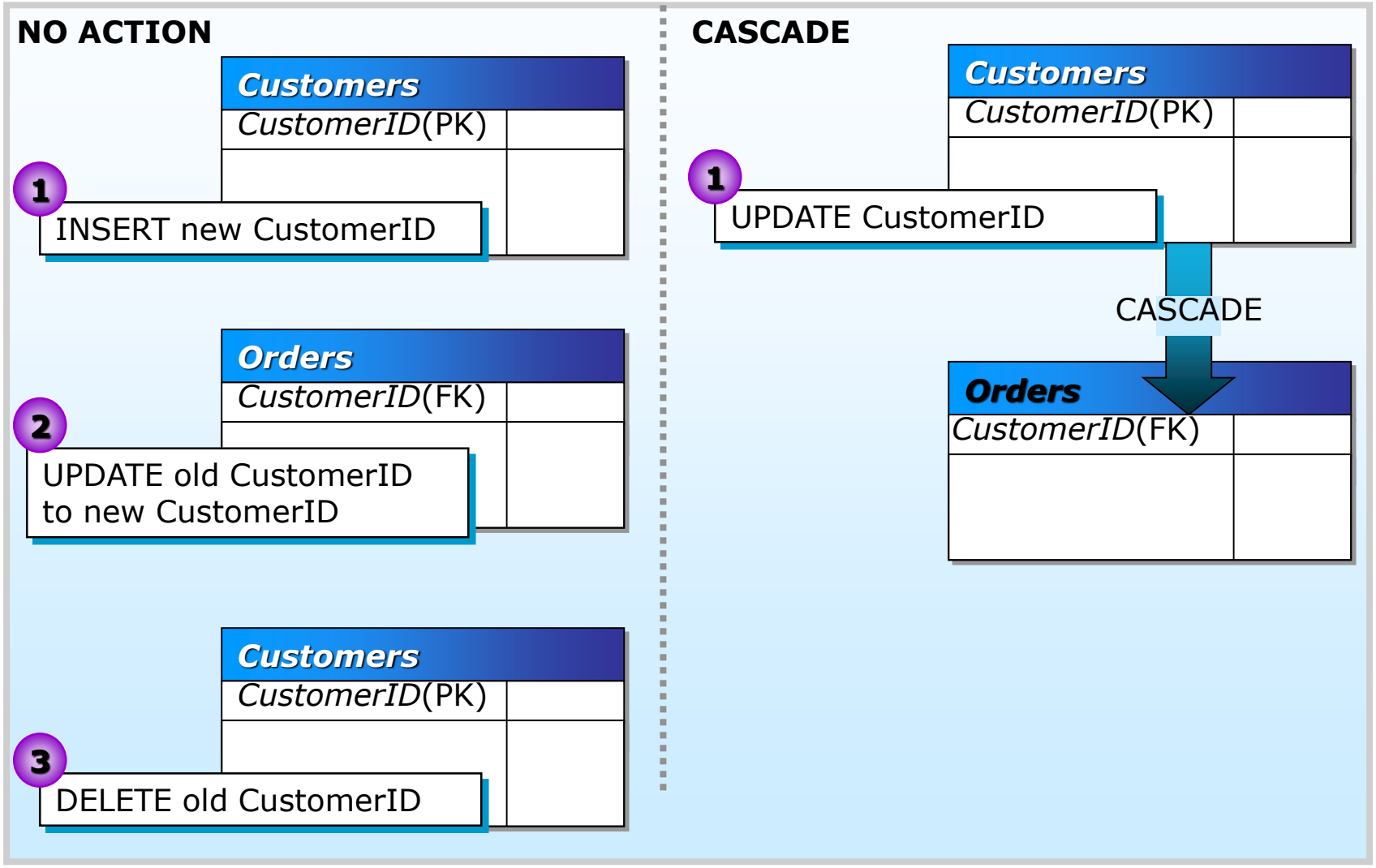
```
USE Northwind
ALTER TABLE dbo.Suppliers
ADD
CONSTRAINT U_CompanyName
  UNIQUE NONCLUSTERED (CompanyName)
```

FOREIGN KEY constraints

- Must Reference a PRIMARY KEY or UNIQUE Constraint
- Provide Single or Multicolumn Referential Integrity
- Do Not Automatically Create Indexes
- Users Must Have SELECT or REFERENCES Permissions on Referenced Tables
- Use Only REFERENCES Clause Within Same Table

```
USE Northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
  FOREIGN KEY (CustomerID)
  REFERENCES dbo.Customers(CustomerID)
```

Cascading Referential Integrity Constraints



Demo

- Demonstration on all types of SQL Server Constraints

Disabling Constraints

- Disabling constraint checking on existing data
- Disabling constraint checking when loading new data

Disabling Constraint Checking on Existing Data

- Applies to CHECK and FOREIGN KEY constraints
- Use WITH NOCHECK option when adding a new constraint
- Use if existing data will not change
- Can change existing data before adding constraints

```
USE Northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
  ADD CONSTRAINT FK_Employees_Employees
  FOREIGN KEY (ReportsTo)
  REFERENCES dbo.Employees(EmployeeID)
```

Disabling constraint checking when loading new data

- Applies to CHECK and FOREIGN KEY constraints
- Use when:
  - Data conforms to constraints
  - You load new data that does not conform to constraints

```
USE Northwind
ALTER TABLE dbo.Employees
NOCHECK
  CONSTRAINT FK_Employees_Employees
```

Using DEFAULTS & RULES

- As independent objects they:
  - Are defined once
  - Can be bound to one or more columns or user-defined data types

```
CREATE DEFAULT phone_no_default
  AS '(000)000-0000'
GO
EXEC sp_bindefault phone_no_default,
  'Customers.Phone'
```

```
CREATE RULE regioncode_rule
  AS @regioncode IN ('IA', 'IL', 'KS', 'MO')
GO
EXEC sp_bindrule regioncode_rule,
  'Customers.Region'
```

Demo

- Disabling Constraints
- Working with DEFAULTS & RULES

Deciding - Enforcement method to use

Data integrity components	Functionality	Performance costs	Before or after modification
Constraints	Medium	Low	Before
Defaults and rules	Low	Low	Before
Triggers	High	Medium-High	After
Data types, Null/Not Null	Low	Low	Before

TRUNCATE

- Removes all rows from a table
- TRUNCATE TABLE is similar to the DELETE statement with no WHERE clause

```
TRUNCATE TABLE  Employees
```

Dropping table

- At times you need to delete a table, for example when you want to implement a new design or free up space in the database
- You can use DROP TABLE Transact-SQL statement to drop the table from Database

```
DROP TABLE   Employee
```

- You can reference multiple tables in a single DROP TABLE command by separating the table names with comma

Commonly used T-SQL Statement

- Transact-SQL statements can be used for:
  - Creating a database
  - Creating and altering table
  - Insert, update & delete data in the table
  - Reading data from the table

Data Manipulation Language

- INSERT - Adds a new row to a table
- UPDATE - Changes existing data in a table
- DELETE - Removes rows from a table
- MERGE- Merges the data

Using INSERT

- INSERT statement adds one or more new rows to a table

  INSERT [INTO] table or view [(column_list)] data_values

```
INSERT INTO MyTable (PriKey, Description)
VALUES (123, 'A description of part 123.')
GO
```

- INSERT using SELECT statement

```
INSERT INTO MyTable (PriKey, Description)
SELECT ForeignKey, Description
FROM SomeTable
GO
```

UPDATE statement

- UPDATE statement can change data values in single rows, groups of rows, or all the rows in a table or view

```
UPDATE products
SET unitprice=unitprice*1.1
WHERE categoryid=2
GO
```

DELETE statement

- Removes one or more rows in a table or view
- A simplified form of the DELETE syntax is:

```
DELETE table_or_view
FROM table_sources
WHERE search_condition
```

- If a WHERE clause is not specified, all the rows in table_or_view are deleted

MERGE statement (SQL Server 2008 onwards)

- In a typical database application, quite often you need to perform INSERT, UPDATE and DELETE operations on a TARGET table by matching the records from the SOURCE table
- To accomplish this, In previous versions of SQL Server, we had to write separate statements to INSERT, UPDATE, or DELETE data based on certain conditions
- Though it seems to be straight forward at first glance, but it becomes cumbersome when you have do it very often or on multiple tables
- Even the performance degrades significantly with this approach
- Now you can use MERGE SQL command to perform these operations in a single statement
- Using MERGE statement we can include the logic of such data modifications in one statement that even checks when the data is matched then just update it and when unmatched then insert it

MERGE statement

- The Merge statement comprises Five clauses
  - MERGE - Used to specify the Target tables to be inserted/updated/deleted
  - USING - Used to specify the Source table
  - ON - Used to specify the join condition on source and target tables
  - WHEN - Used to specify the action to be taken place based on the result of the ON clause
  - OUTPUT - Used to return the value of Insert/Update/Delete operations using the INSERTED / DELETED magic tables
- MERGE Statement - Syntax

```
MERGE <target_table> [AS TARGET]
USING <table_source> [AS SOURCE]
ON <search_condition>
[WHEN MATCHED THEN <merge_matched> ]
[WHEN NOT MATCHED [BY TARGET] THEN
<merge_not_matched> ]
[WHEN NOT MATCHED BY SOURCE THEN <merge_ matched> ];
```

MERGE statement

- MERGE Statement - Example

```
MERGE INTO dbo.Customers AS TGT
USING dbo.CustomersStage AS SRC
ON TGT.custid = SRC.custid
WHEN MATCHED THEN
UPDATE SET
TGT.companyname = SRC.companyname,
TGT.phone = SRC.phone,
TGT.address = SRC.address
WHEN NOT MATCHED THEN
INSERT (custid, companyname, phone, address)
VALUES (SRC.custid, SRC.companyname, SRC.phone,
SRC.address)
WHEN NOT MATCHED BY SOURCE THEN
DELETE
OUTPUT
$action, deleted.custid AS del_custid, inserted.custid AS
ins_custid;
```

MERGE statement

- The given example does the following
  - MERGE statement defines the Customers table as the target for the modification
  - CustomersState table as the source
  - The MERGE condition matches the custid attribute in the source with the custid attribute in the target
  - When a match is found in the target, the target customer’s attributes are overwritten with the source customer attributes
  - When a match is not found in the target, a new row is inserted into the target, using the source customer attributes
  - When a match is not found in the source, the target customer row is deleted

Demo

- Using MERGE Statement

DCL Statements

- DCL (Data Control Language)
  - DCL Statement is used for securing the database
  - DCL Statement control access to database
  - DCL statements supported by T-SQL are GRANT , REVOKE and DENY

DCL Statements

- GRANT authorizes one or more users to perform an operation or a set of operations on an object.
  - GRANT PRIVILEGES ON object-name TO endusers;
  - GRANT SELECT, UPDATE ON My_table TO some_user, another_user;
  - GRANT INSERT (empno, ename, job) ON emp TO endusers;
- REVOKE removes or restricts the capability of a user to perform an operation or a set of operations.
  - REVOKE PRIVILEGES ON
    object_name FROM endusers
  - REVOKE INSERT, DELETE ON
    emp FROM enduser
- DENY denies permission to a user/group , even though he may belong to a group/role having the permission
  - DENY SELECT ON Employees TO Ruser1

Summary

- In this lesson, you have learnt:
  - Different datatypes to store numeric, character, monetary and date time data
  - New data types in SQL Server 2008 - Hierarchyid & Spatial
  - Creating UDTS
  - Creating Tables ,Manipulating Data and Assigning and revoking privileges.
  - MERGE provides an efficient way to perform multiple DML operations
  - Implement constraints like – Primary key, Foreign key, check, default and unique
  - Alter table to change structure, enable/disable constraints

Review Question

- Question 1: To create a user defined data type we use _____.
- Question 2: If we want the column data to have unique values with null then we should use _____ constraint.
- Question 3: _____ and _____ are objects like constraints that are bound to the table.
- Question 4: _____ option is used to create a constraint that should not be temporarily checked for.
- Question 5: Use _____ as a data type to create tables with a hierarchical structure.
- Question 6: _____ Used to return the value of Insert/Update/Delete operations using the INSERTED / DELETED magic tables