Case Study
Journey to Developer Certification

Part #1

Implementing Tables and Views

Lands of the Journey

- Implementing Tables and Views
- Implementing Programming Objects
- Working with Query Fundamentals
- Applying Additional Query Techniques
- Working with Additional SQL Server Components
- Working with XML Data
- Gathering Performance Information

Agenda

- Data Types
- Create and Alter Tables
- Create and Alter Views
- Create and Alter Indexes
- Create and Modify Constraints
- Implement New Data Types

Exact Numerics

Type	From	To
BigInt	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
Int	-2,147,483,648	2,147,483,647
Smallint	-32,768	32,767
Tinyint	0	255
Bit	0	1
Decimal	-10^38+1	10^38 - 1
Numeric	-10^38+1	10^38 - 1
Money	-922,337,203,685,477.5808	922,337,203,685,477.507
Smallmoney	-214,748.3648	214,748.3647

Approximate Numerics

Type	From	To
Float	-1.79E + 308	1.79E + 308
Real	-3.04E + 38	3.40E + 38

Date and Time

Type	To	From
Date	01/01/0001	12/31/9999
DateTime2	01/01/0001 00:00:00.0000000	12/31/9999 23:59:59.9999999
DateTime	01/01/1753	12/31/9999
DateTimeOffSet	01/01/0001 00:00:00.0000000	12/31/9999 23:59:59.9999999
Time	00:00:00.0000000	23:59:59.9999999

DateTimeOffSet is in UTC

Character Strings

Type	Description
Char	Fixed-length non-Unicode character data with a maximum length of 8,000 characters.
Varchar	Variable-length non-Unicode data with a maximum of 8,000 characters.
Varchar(max)	Variable-length non-Unicode data with a maximum length of 2^{31} characters
Text	Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

Unicode Character Strings

Type	Description
Nchar	Fixed-length Unicode data with a maximum length of 4,000 characters.
nvarchar	Variable-length Unicode data with a maximum length of 4,000 characters.
Nvarchar(max)	Variable-length Unicode data with a maximum length of 2^{30} characters
ntext	Variable-length Unicode data with a maximum length of 1,073,741,823 characters.

Binary Strings

Type	Description
Binary	Fixed-length binary data with a maximum length of 8,000 bytes.
Varbinary	Variable-length binary data with a maximum length of 8,000 bytes.
Varbinary(max)	Variable-length binary data with a maximum length of 2^{31} bytes
Image	Variable-length binary data with a maximum length of 2,147,483,647 bytes.

Other Data Types

Type	Description
SQL_Variant	Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
HierarchyID	Is a variable length, system data type. Used to represent position in a hierarchy. This does not automatically represent a tree. It is up to the application to generate and assign values in such a way that the desired relationship between rows is reflected in the values.
Timestamp	Stores a database-wide unique number that gets updated every time a row gets updated.
Uniqueidentifier	Stores a globally unique identifier (GUID).
Xml	Stores XML data. You can store xml instances in a column or a variable
Cursor	A reference to a cursor.
Table	Stores a result set for later processing.

Case Study

Create Member Information

Business Rules

- Create a storage area for the Member's
  - Store Member Name
  - Store Member Contact Information
  - Store Date the Member Started.

Create Storage Area

```
USE master
GO

DROP DATABASE SQLUserGroup
GO

CREATE DATABASE SQLUserGroup
GO

USE SQLUserGroup
GO

CREATE SCHEMA UserGroup AUTHORIZATION dbo
  DENY SELECT TO PUBLIC
GO
```

DEMO

Create Storage Area

```
CREATE TABLE SSUG_Member
(
    Mbr_id                  TINYINT
  , Mbr_name                VARCHAR(50)
  , Mbr_Email               VARCHAR(100)
  , Mbr_StartedOn           DATETIME
  , CreatedOn               DATETIME
)
```

DEMO

New Business Rule

- Need to have a Member Identifier that the member can use to query for the record

Computed Field

A computed column is computed from an expression that can use other columns in the same table. The expression can be a non-computed column name, constant, function, and any combination of these connected by one or more operators. The expression cannot be a subquery.

Unless otherwise specified, computed columns are virtual columns that are not physically stored in the table. Their values are recalculated every time they are referenced in a query.

Calculation for MemberID

Using the ID field and Pre-append ‘0’ to the ID field to produce a character string of size 10.

```
REPLICATE('0', 10 - LEN(RTRIM(CONVERT(VARCHAR(10), ID)))) +
CONVERT(VARCHAR(10), ID)
```

DEMO

New Business Rule

- Need to create a way for the marketing department to produce a view of member's email address for electronic mailing.

Make sure table does not change

SchemaBinding

Binds the view to the schema of the underlying table or tables. The base tables cannot be modified in a way that would affect the view definition. The view definition itself must first be modified or dropped to remove dependencies on the table to be modified. When you use SCHEMABINDING, the *select_statement* must include the two-part names (*schema*.*object*) of tables, views, or user-defined functions that are referenced.

All referenced objects must be in the same database.

Create a View

```
CREATE VIEW UserGroup.VW_MemberMailing
(
    Email
  , MemberStarted
)
WITH SCHEMABINDING
AS
SELECT Mbr_Email
     , Mbr_StartedOn
  FROM UserGroup.SSUG_Member
 WHERE BroadCastEmail = '1'
```

DEMO

INDEXES

SQL Server collects statistical information about indexes and column data stored in the database. These statistics are used by the query optimizer to choose the most efficient plan for retrieving or updating data. Good statistics allow the optimizer to accurately assess the cost of different query plans, and choose a high-quality plan.

Index Information Stored

All information about a single statistics object is stored in several columns of a single row in the **sysindexes** table, and in a statistics binary large object (**statblob**) kept in an internal-only table. In addition, information about statistics can be found in the metadata views **sys.stats** and **sys.indexes**

Filtered Indexes

There is now a way to create index for only subset rows in a table. Although some value is needed to query. Imagine that a table has million number of rows and only certain value in a column is queried in filtering using WHERE clause of SELECT query. It needs additional space to keep track of those millions of rows and rebuild or reorganize those index will take longer to finish. SQL Server 2008 introduces filtered index and statistics that will give you better option to generate index only subset of rows that are needed to satisfy query with certain criteria.

Ways to Create Statistcs

- The optimizer automatically creates single-column statistics as needed as a side effect of optimizing SELECT, INSERT, UPDATE, and DELETE statements when AUTO_CREATE_STATISTICS is enabled, which is the default setting.
- CREATE INDEX statement.
- Use sp_createstats to create statistics for all eligible columns (all except XML columns) for all user tables in the current database. A new statistics object will not be created for columns that already have a statistics object.
- Use dbcc dbreindex to rebuild one or more indexes for a table in the specified database.
- In Management Studio, expand the folder under a Table object, right click the Statistics folder, and choose New Statistics.
- Use the Database Tuning Advisor (DTA) to create indexes.

DEMO

Final Tweeks

- Primary Keys
- Check
- Unique
- Defaults

DEMO

New
Data Types

New Data Types

- Date
- Time
- HierarchyID
- XML
- Spatial
- FileStream

Date/Time Data Types

- Larger Range of Dates
- Larger Fractional Second Percision
- Time Zone Awareness
- Database Migration Support
- ANSI SQL Standard Compliant

Date and Time

Type	To	From
Date	01/01/0001	12/31/9999
DateTime2	01/01/0001 00:00:00.0000000	12/31/9999 23:59:59.9999999
DateTime	01/01/1753	12/31/9999
DateTimeOffSet	01/01/0001 00:00:00.0000000	12/31/9999 23:59:59.9999999
Time	00:00:00.0000000	23:59:59.9999999

DateTimeOffSet is in UTC, Calendar type is Gregorian

HierarchyID

Microsoft has introduced its first system-defined CLR data type. The hierarchyid data type, as its name implies, is designed to store parent-child relationships between records in a single table.

Hierarchy Methods

Method Name	Definition
GetAncestor	The GetAncestor method is used to find parents and grantparents of the current record or value.
GetDescendant	The GetDescendant method is used to find the child record of the given record or value.
GetLevel	The GetLevel method is used to find the number of levels deep that a record or value is. This method can also be used to filter a result set to pull records above or below a certain number of levels.
GetRoot	The Get Root method is used to find the root of the hierarchy tree. While an employee tree should have a single root, other uses may have more than one root. One example might be product categories which can have categories nested beneath them.

Hierarchy Methods Contd.

Method Name	Definition
IsDescendant	The IsDescendant method is used to determine if a record or value is a child of a second record or value.
Parse	The parse method is used to convert the canonical version of the data type to a hierarchyid value.
Read	The read method is only available within .NET CLR code. It is used to read the binary representation of the hierarchyid value.
Reparent	The Reparent method is used to either verify that the current tree and a proposed to tree are valid, as well as to move the records from a current tree to a new tree.
ToString	The ToString method is used to show the tree in a human readable slash (/) notated format.

HierarchyID Classic Example

Employee to manager relationship that we often see defined in Human Resources (HR) type of database application.

Hierarchy Table

```
-- Make the table
CREATE TABLE Employee
(
    EmployeeID              INT IDENTITY (1,1) PRIMARY KEY
  , LastName                NVARCHAR(50)
  , FirstName               NVARCHAR(50)
  , Title                   NVARCHAR(50)
  , HireDate                DATETIME
  , PhoneExtention          INT
  , EmailAddress            VARCHAR(255)
  , NodeRec                 HIERARCHYID
)
GO
```

Add First Employee

```
-- Define needed variables
DECLARE @NodeRec hierarchyid
SET @NodeRec = hierarchyid::GetRoot()

-- Add 1st employee
INSERT INTO Employee
(    LastName
   , FirstName
   , …
   , EmailAddress
   , NodeRec
) VALUES(
     'Smith‘
   , 'Frank‘
   , …
   , 'frank.smith@northwindtraders.com‘
   , @NodeRec.GetDescendant(null, null))
```

Add Next Employee

```
DECLARE @Manager hierarchyID, @Employee hierarchyID

-- Define needed variables
SELECT @Manager = NodeRec  FROM Employee WHERE EmployeeID = 1

SELECT @Employee = max(NodeRec)  FROM Employee
 WHERE NodeRec.GetAncestor(1) = @Manager

INSERT INTO Employee
(LastName, FirstName, HireDate, Title, PhoneExtention, EmailAddress, NodeRec)
VALUES
('Jefferson', 'John', '3/23/2006', 'VP Sales', 65647,
'john.jefferson@northwindtraders.com', @Manager.GetDescendant(@Employee, null))
```

DEMO

XML

Previous Versions

- SQL Server 2000 introduced XML-related capabilities with the FOR XML and OPENXML Transact-SQL keywords, which enabled developers to write Transact-SQL code to retrieve a query result as a stream of XML, and to shred an XML document into a rowset.
- SQL Server 2005 introduced native **xml** data type that supports XSD schema validation, XQuery-based operations, and XML indexing.

SQL Server 2008

- Improved schema validation capabilities
- Enhancements to XQuery support
- Enhanced functionality for performing XML data manipulation language (DML) insertions

XML Schema

- Support for lax validation
  - XML Schemas support wildcard sections in XML documents through the any, anyAttribute, and anyType declarations.
- Full support for dateTime, time and date validation, including preservation of time zone information
- Improved Support for union and list types

DEMO

Next Time

- Data Types
  - FileStream
  - Spatial
  - Collations
- Partitioning Solutions
  - Tables & Indexes
    - (constraints, partition functions, partition schemes, MERGE, SPLIT, SWITCH)
  - Views
    - (constraints, linked servers)

Questions

If you have any questions, please email me at spheffron@gmail.com

Thank
You