Microsoft SQL Server/Print version

Contents

System Requirements

This software is available on Microsoft Windows, its 32-bit editions are different to the 64-bit editions[1].

32-bit editions of SQL Server 2005

Minimum processor: PIII 600 (1 GHZ or faster recommended).

Minimum OS:

Enterprise

- Windows Server 2003 (Standard, Enterprise or Datacenter) with SP1 or later.
- Windows Small Business Server 2003 (Standard or Premium) with SP1 or later.
- Windows 2000 (Server, Advanced Server, Datacenter Server) with SP4 or later.

In short: all OS after Windows XP.

Standard

All of Enterprise, but also non server Windows OS.

Windows Professional with SP4 and Windows XP with SP2 or later.

Workgroups

All of standard and Windows XP Media and Tabled Edition.

Express

All of the above + Windows Home Edition with SP2 or later and Windows Server 2004 Web Edition with SP1 or later.

Developer

1+2+3 + 4.5 Windows Home Edition with SP2.

Memory Requirements

- Enterprise, Development Standard and Workgroup: 512 min, 1 GB more recommended.
- Express: 192 min, 512 or more recommended.

Harddisk requirement

350 MB full and 390 for sample.

Internet

- Explorer SP1 required for all installations (Required for Microsoft Management Console (MMC) and HTML Help.
- IIS Required for XML applications and for Reporting Services.
- TCP/IP must be enabled.

64-bit editions of SQL Server 2005

Processor

- IA64 Minimum: 1 GHZ or faster Itanium.
- X64 minimum: 1 GHz or faster AMD Opteron, AMD Athlon 64, Intel Xenon with Intel EM64T support or PIV with EM64T support.

References

1. https://msdn.microsoft.com/en-us/library/ms143506.aspx

Installing

There is too choose between a default or named instance. You can install only one default instance, but more named instances.

Understanding SQL Server 2005 Editions

There are five different editions of the SQL Server 2005. Three of them have 64-bit native versions, the rest is 32-bit only.

Enterprise Edition

Largest, 32-bit and 64-bit, supports the largest online transaction processing (OLTP (http://en.wikipedia.org/wiki/OLTP))

Standard Edition

32-bit and 64-bit, small and medium enterprises

Workgroup Edition

Small Enterprises, no limits on size of users, for small servers

Developer Edition

Same as Enterprise without production license

Express Edition

MSDE for SQL Server 2005; free(also for clients access); replacement for Access

Some hints to remember Features

Express; Workgroup; Standard; Enterprise

Number of CPUS: 1;2;4;unlimited. Think of a formula such as CPUs of a Edition == Previous Edition *2 expect for Enterprise.
Memory 1GB;3GB, Limit based on OS, Limit based on OS
64-bit support: Express and Workgroup can only have 64-bit support via WOW
Max DB Size: Express has 4 GB limit

Partitioning is only supported by Enterprise Edition
Database Mirroring is supported by Standard and Enterprise
Fail Over Clustering is supported by Standard(two Nodes) and Enterprise
PDF Rule: Remember Partitioning, Database Mirroring, Failover P == 1 only one edition supports

Log Shipping: All but Express
Management Studio All but Express
Database Tuning Advisor: Standard and Enterprise
Full Text Search: All but Express
SQL Server Agent Job Scheduling: All but Express
Best Practives Advisor: All
Notification Services Standard and Enterprise
Service Broker: All (Express Subsriber only)

Merge Replication: All (Restriction on Express and Workgroup)
Transactional Replication: Restriction on Express and Workgroup
Oracle Replication: Enterprise
Web Services (Http Endpoints): Standard and Enterprise
Report Server: All
Report Builder All but Express
BI Development Studio Express and Workgroup have Report Designer only
Enterprise Managemnet Tool: All but Express
Native Support for Web Services: Reporting Services for Express and Workgroups only
Analysis Services for Standard and Enterprise only

Rules of thumb:

- Express Edition is limited nearly everywhere: It has only Report Server and Best Practices Analyzer and in some only a feature as subscriber only
- Enterprise Edition has everything
- Standard does not support: Oracle Replication and Partitioning and has some Restrictions on Fail Over clustering:
- Workgroup: hardest to Remember: It does not include the features to maintain large databases

How to Install MS SQL Server 2005-2012

Installation

The Minimum System Requirements

Windows 32-bit System (Minimum Vista with SP2 or Server with SP2 installed), Intel or other supporting processor with at least 2.66 GHz Speed (My personal suggestion), RAM should be 2 GB or more (if you want to run several applications such as Visual Studio with this server) ”The more you have, the more it will be better”.

[Note:32-bit Systems do not exceed more than 3 GB RAM, if you use 4 GB then it will utilize only 3 GB RAM]

Here we go

Is .Net Framework 3.5 or with SP1 or 4.0 installed? If not then here is the official link (I don’t like unofficial, mind you…). Read the instructions,when downloading and installing and make sure it has been installed. Now , its time to install your SQL Server Database.

1. Right click on the SQLEXPRWT_x64_ENU.exe or SQLEXPRWT_x86_ENU.exe SQLEXPRWT_x64_ENU.exe and click ” Run As Administrator”
2. The installation wizard will appear and it will ask for several options.Set your desired language.Choose that option which indicates
   “New Standalone SQL Installation or Add Features ” and click Next to proceed. A small auto-check will run and when it gets approval for all the pre-requisites, you will be asked to use the evaluation version or if you have bought a key then enter this and use the full version which has whole lots of benefits.Click Next
3. Accept Terms and conditions and check Updates checkbox (recommended) and click Next.

4. If updates are present then you won't get any error. Otherwise, an error will be shown and you have to click next in order to install the updates (That's why I was telling that enable auto update man….). If you have the connectivity issues, check the firewall and allow for both inbound and outbound traffic in the allow rule for this type of application. After the errors have been removed, you have to select the features to install.
5. Everything after this will go smoothly. You will be asked to add either the default instance or the named instance .There is nothing to get worried about those instances sort of creatures.
   "Take the easiest example. We have the main SQL Server Database which resides in my server computer and it is called manager-pc.The other employees of my company have specific needs from this server .
   I assign the instances of my Server as follows:
   Employee1 has been assigned the instance from my Database Server as manager-pc/instance1 for writing queries etc. Employee2 has been given more rights in the form of manager-pc/instance2 instance and so on.
   - If someone has to show the reports from the instance of the server assigned to Employee2; he has to address the instance2 of manager-pc. So basically, he needs to refer to manager-pc/instance2.
   - If someone has to connect to the default instance, he needs to go to manager-pc directly. The instances are really great from preventing data corruption or intrusion,hacking, SQL injection related issues Just relax.It's simple :)"
6. The disk space requirements screen is quite obvious to show you the required space as well as available
7. Now, here comes the screen asking you to configure the network settings.I will suggest you to assign account to each service on individual basis as it is recommended by the Microsoft. However, I use my own server's administrator account for all the services. It is up to you to set the services to start manually or automatically.It needs some study material from here when making this decision.Click the Next button to proceed
8. The Database Engine Configuration screen gives you the option to choose the type of authentication. Here ,we have the default user login id sa which is kept intact and we choose Mixed Mode and assign a good password . Click Next.
9. The next screen will take you to either change the path of the Data Directory or keep it unchanged and click??? Click what? obviously NNNNN…Next
10. In order to store the unstructured data in your database such as images or files etc, you should enable filestream by checking it and then click Next button
11. Let the Multidimensional Data Mining option checked and click Next
12. Error Reporting should be checked for descriptive explanation about the condition of the errors, if they occur in your server. this helps us to send the errors to Microsoft directly and yes; they help.
13. Installation configuration rules auto-checking screen appears and you have to……..You have to do nothing dear but click Next after the check completes.
14. Click Next on Install screen
15. Installation …….. sit back and relax. After successful completion ; click Next. Here, you may have to do one more thing ! Shout out lower and say "Oh my" if the installation gets failed!!!!!
16. Installed… congrats
17. After installation, check to see whether the required services are working ; SQL Server Configuration Manager is there to help.

Graphical interface

SQL Server Management Studio (SSMS) is the official administration interface for Microsoft SQL Server. Its 2008 version can be downloaded on https://www.microsoft.com/download/confirmation.aspx?id=7593.

References

- http://www.technofranchise.com/simplest-steps-to-install-sql-server/

Connecting to MS SQL Server from *nix

This tutorial assumes you are using a Debian based distribution of Linux; if not, you might have to modify the instructions appropriately for you distro.

Downloading

To install both `freetds` and the drivers for `iODBC` and `unixodbc` use `apt-get`, i.e.,

```
apt-get install tdsodbc
```

If you plan to use iODBC, you'll want to run

```
apt-get install libiodbc2 libiodbc2-dev
```

If you're using Perl, after you install FreeTDS you'll want to install the DBD for Sybase:

```
export SYBASE=/etc
cpan DBD::Sybase
```

Glossary

TDS
: Designed by Sybase, the **T**abular **D**ata **S**tream Protocol (TDS) describes how to communicate with MS-SQL and Sybase databases.

Troubleshooting

Many of the errors here are to assist those with Google searches and the users.

Failing to set SYBASE

The following error is a result of not setting the sybase configuration variable

```
[ERROR] [Thu Mar 13 11:07:42 2008] Could not run '/usr/local/bin/perl Makefile.PL': Please set SYBASE in
set the $SYBASE environment variable at /root/.cpanplus/5.10.0/build/DBD-Sybase-1.08/Makefile.PL line 10
line 44.
```

External Links

1. iODBC (http://www.iodbc.org)
2. FreeTDS (http://www.freetds.org/)

References

1. Haynes, Tim (2004-03-23). "ODBC-Perl HOWTO". OpenLink Software. http://www.iodbc.org/index.php?page=languages/perl/odbc-perlHOWTO. Retrieved 2008-03-04.
2. Bruns, Brian (2006). "FreeTDS User Guide: A Guide to Installing, Configuring, and Running FreeTDS". http://www.freetds.org/userguide/. Retrieved 2008-03-04.

Using Instances

On one machine, there can be more editions

Default Instance: Only one. Please note some of the MS products using the SQL Server request the default instance (For instance the Team Foundation Server).

Determining Multiple or Single Instances

Multiple Instances mean more administration overhead. Additional Instances require additonal resources: Memory and processor capacity-

Advantages of multiple instances

- testing multiple versions
- testing service packs, dev databases and applications
- different customers require their own system and user databases with full administrative control or their instance
- desktop engine is embedded in the applications, because every application can install each own instance

Installing Instances

Decide on Security and collation

SQL Server Agent and SQL Server run as Windows Services.

They run in a context of a user account. Determining which account is an important decision.

Questions:

- Should use use a separate account for the SQL Server service and the SQL Server Agent service, or should you use the same account for both?
- Should you use a built-in system account or a domain user account?

Remember: The SQL Server jobs require to interact with the SQL Server Agent service. When the SQL

Server Agent service must interact with different servers, DBAs create different accounts for these two services to avoid giving the SQL Server service more permission than it needs.

Build-in System account or Domain user account

You can choose between: Network service account, local system account, or a dedicated domain user account.

The Network service account is a special built-in system account that is similar to authenticated user accounts. This account has the same level of access to system resources and objects as members of the Users group. Services that run under this account will use the credentials of the computer account to access network resources. Not recommended to use.

The local system account is a Windows OS account that has full adminstrativve rights on the local computer but has no network rights. You can use this account for development or testing of servers that you do not integrate with other server applications or to interact with any network resources. Not recommended.

Recommended: Create and use one or two dedicated domain user accounts for the SQL Server and SQL Server Agent services.

Authentication mode

Windows and Mixed mode. Default is Windows: Only users that have previously authenticated to the Windows OS can connect to the SQL Server 2005 instance.

Mixed is recommended for Legacy.

Determining Collation Setting

You define the default collation for a SQL Server at installation. The SQL Server uses the collation setting to determine how non-Unicode character data is stored and how to sort and compare Unicode and non-Unicode data.

Default will be selected of Windows.

Configuring Files

data files contain data and objects (tables, indexes and so on) and log files contain transaction log.

filegroups are grouped datafiles for easier admin.

Data Files

There are primary and secondary Data Files.

The primary has the extension mdf and contains data and all information regarding data (such as information on the secondary data files). For optimal performance do not store data there.

The secondary has the extension ndf. No administrative data is stored here. There is a maximum of 32,766 secondary data files.

Log Files

Extension ldf. Each db needs at least one log file. You can create more than one

Filegroups

Logical structure of data files.

Also there are primary and secondary filegroups. The primary contains the primary data file and every secondary data file not stored in a specific filegroup. There can be up to 32,766 secondary file groups.

Filegroups can be configured as read only.

The filegroups are created with the SSMS or with the CREATE DATABASE command.

Five parameters have to be considered: Name, Filename, Size, Maxsize and Filegrowth

Configuring Raid Systems

There are Raid Levels 0,1,5 and 10. It has to be considered to be used in high performance environments. Configuring Database Files with RAID Systems RAID systems are arrays of disk drives that provide fault tolerance, more storage capacity, and better performance for the disk subsystem, depending on the configuration. Although RAID hardware systems are not part of the SQL Server configuration, they directly affect SQL Server's performance. There are a variety of RAID levels, each of which uses a different algorithm for fault tolerance. The most common RAID levels used with SQL Server are 0, 1, 5, and 10.

- RAID 0 is also known as disk striping because it creates a disk file system called a stripe set. RAID 0 gives the best performance for read and write operations because it spreads these operations across all the disks in the set. However, RAID 0 does not provide fault tolerance; if one disk fails, you lose access to all the data on the stripe set.
- RAID 1, also known as disk mirroring, provides a redundant copy of the selected disk. RAID 1 improves read performance but can degrade the performance of write operations.
- RAID 5, the most popular RAID level, stripes the data across the disks of the RAID set as does RAID 0, but it also adds parity information to provide fault tolerance. Parity information is distributed among all the disks. RAID 5 provides better performance than RAID 1. However, when a disk fails, read performance decreases. 58 Chapter 2 Configuring SQL Server 2005
- RAID 10, or RAID 1+0, includes both striping without parity and mirroring. RAID 10 offers better availability and performance than RAID 5, especially for write-intensive applications. The RAID configuration that is best for your database files depends on several factors, including performance and recoverability needs. RAID 10 is the recommended RAID system for transaction log, data, and index files. If you have budget restrictions, keep transaction log files in a RAID 10 system, and store data and index files in a RAID 5 system.

Best practices

1. Do not put data files on the same drive as OS files
2. Separate transaction log files from data files
3. database tempdb should be on a separate drive (Raid 10 or RAID 5)

Configuring Mail

First of all: SQL Server Mail does not depend on Extended Mapi.

Prerequesites for Database Mail:

- Database Mail has to be enabled via the Surface Area Configuration Tool, Database Mail Configuration Wizard or the sp_configure[1].
- Service Broker needs to be enabled in the Database Mail host database msdb
- The Database Mail externals executables need access to the SMTP Server

Architecture

Four Components: Configuration Component, Messaging Component, Database Mail executable, Logging and auditing component.

The configuration components consists of a *database Mail account* and a *Database Mail profile*. The Database Mail account contains the information that SQL Server uses to send e-mail messages to the SMTP server, such as the SMTP server name, the authentication type, and the e-mail address.

References

1. http://technet.microsoft.com/en-us/library/ms191207%28v=sql.105%29.aspx

System databases

In this chapter you will learn the purpose of the *system databases* installed with every instance of Microsoft SQL Server.

Introduction

When you install Microsoft SQL Server, five system databases are automatically created on every SQL Server instance. These system databases allow the database engine and administrative applications to properly manage the system:

- Resource (*SQL Server 2005 and higher only*)
- master
- model
- msdb
- tempdb

Resource[1]

The Resource database is a read-only database that contains all SQL Server system objects. This system database is not displayed in the SQL Server Management Studio object explorer, so users may not be aware of it. Interaction with the Resource database is indirect, through master database system views and functions that reference these objects. Because the Resource database is read-only, it is only modified by the system when a SQL Server hotfix, service pack, or upgrade is installed on an instance. The Resource database is installed only with SQL Server 2005 and higher.

master[2]

The master database records the server-wide configuration information for a SQL Server instance, including all logins and database engine configuration defaults. The master database stores metadata including references to all other databases on the instance. This database records the initialization information for SQL Server, and an instance of SQL Server cannot start up if the master database is missing, corrupt, or otherwise unavailable. Prior to SQL Server 2005 all system objects were stored in the master database. Beginning with SQL Server 2005, the master database provides access to system objects through views, functions, and stored procedures that reference the read-only Resource database.

tempdb[3]

The tempdb database is considered a *global resource* that all connections and all users can access. The tempdb database holds user-created temporary database objects, such as temporary tables and temporary stored procedures. This database is also used heavily by the database engine to serialize intermediate results and to hold other temporary data and objects that are generated during processing. Tempdb is always recreated from when the SQL Server service is first started. Because tempdb is so heavily used by the system, many system-wide performance optimizations are necessarily focused on making tempdb as efficient as possible.

model[4]

The model database is a template for all newly-created databases on an instance. When a database is created it is initialized as a copy of the model database and then modified to the correct size and settings specified at creation time. Any objects in the model database are automatically copied to the new database. For this reason, some database administrators (DBAs) create functions or other database objects in the model database that they want to exist in all newly created databases.

Prior to SQL Server 2005 the database initialization process required allocating and wiping out ("zero-filling") all pages when a database was created. Beginning with SQL Server 2005 using the "instant initialization feature" improves performance by skipping the zero-fill when new data pages are allocated.

The tempdb database is recreated from a copy of the model database every time SQL Server is started, so the model database must always exist on a SQL Server system.

msdb[5]

The msdb database is a system database used to store configuration and processing data and metadata for SQL Server features and applications including SQL Server Agent (job schedules and alerts), SQL Server Integration Services (ETL packages), Database Mail, and Service Broker.

Usage

Below is one request to get all the server databases by sizes:

```
EXECUTE master.sys.sp_MSforeachdb 'USE [?]; EXEC sp_spaceused'
```

Summary

SQL Server installs five system databases with every server instance: **Resource** (SQL Server 2005 and higher only), *master*, *tempdb*, *model*, and *msdb*. Each provides a specific set of system-wide functionality to the SQL Server instance, including access to server-wide configuration metadata, utility configuration and processing data, and the ability to create and utilize temporary objects.

References

1. Resource database (Books Online) (http://technet.microsoft.com/en-us/library/ms190940.aspx)
2. master database (Books Online) (http://technet.microsoft.com/en-us/library/ms187837.aspx)
3. tempdb database (Books Online) (http://technet.microsoft.com/en-us/library/ms190768.aspx)
4. model database (Books Online) (http://technet.microsoft.com/en-us/library/ms186388.aspx)
5. msdb database (Books Online) (http://technet.microsoft.com/en-us/library/ms187112.aspx)

Table manipulation

Introduction

The DDL and DML respect the SQL-86 norm. However, in addition to the requests `SELECT, UPDATE, INSERT` we find `MERGE` since the 2008 version[1].

Create a table

In SSMS, a right click on the folder "Tables" of a database allows to add one.

A right click on a particular table lets choose between:

1. Modify the table structure (add a column, modify a type).
2. Select its 1,000 first records (`TOP`), or the 1,000 last (`ORDER BY id DESC`).
3. Edit its 200 first.

Otherwise in SQL one must enter[2]:

```
CREATE TABLE [dbo].[table1] (
    [Nom] [varchar](250) NULL,
    [Prénom] [varchar](250) NULL,
    [identifiant] [int] IDENTITY(1,1) NOT NULL)
```

Filling the first columns[3]:

```
INSERT INTO table1 VALUES ('Doe', 'Jane', 1), ('Doe', 'John', 2)
```

To aim some other columns, the fields must be precised. For example, by filling the first name, the last name

will be null:

```
INSERT INTO table1 (First_name, id) VALUES ('Jane', 3)
```

From another table:

```
INSERT INTO table1 (First_name, id)
SELECT First_name, ID FROM table2
```

Update:

```
UPDATE table1
SET First_name = 'Janet'
WHERE ID = 3
```

```
UPDATE table1
SET First_name = t2.First_name, Last_name = t2.Last_name
FROM table1 t1
INNER JOIN table2 t2 on t1.ID = t2.ID_t1
```

Create an index

The software PK abbreviation means "primary key".

To create a foreign key, drop down the table, in the menu *Keys*, right click, *new foreign key*..., the list of all the table foreign keys appears in a small window (named by default "FK_..." for "foreign key").

In *General*, *Tables and columns specification*, click on "..." to select the table and its field to link.

Add a unique id

Normally each table should own at least one unique id (primary key). However, it's impossible to modify an existing column to attribute the property `AUTOINCREMENT` needed to such a key.

So to add one:

```
ALTER TABLE table1 ADD id int NOT NULL IDENTITY (1,1) PRIMARY KEY
```

Copy a table

The selection below clones a table with the same fields sizes:

```
SELECT * INTO table2 FROM table1
```

Knowing that the table `spt_values` from the system database `master` contains a sequential field `number`, it becomes possible to generate tables with this counter:

```
SELECT DISTINCT number
FROM master.dbo.spt_values
WHERE number BETWEEN 2 AND 10
```

Where:

```
SELECT DISTINCT 'Line ' + convert(varchar, number, 112) as N into #BlankTable
FROM master.dbo.spt_values
WHERE number BETWEEN 2 AND 10

SELECT * from #BlankTable
```

```
N
Line 10
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
```

Import a table

From an array (Excel or Calc) converted for example in CSV encoded in PC DOS, to import it as a new table[4] :

```
CREATE TABLE Array_to_Table (
  [Champ1] [varchar](500) NULL,
  [Champ2] [varchar](500) NULL,
  [Champ3] [varchar](500) NULL
)
GO
BULK INSERT Array_to_Table
FROM 'C:\Users\superadmin\Desktop\Array1.csv'
WITH (
  FIELDTERMINATOR = ';',
  ROWTERMINATOR = '\n'
)
GO
-- Displays the result
SELECT * from Array_to_Table
GO
```

Delete a table

To delete a whole table (data and structure):

```
DROP TABLE table1
```

To truncate a table, that is to say to conserve only the headers and columns types, by removing all records:

```
TRUNCATE TABLE table1
--or
DELETE table1
```

To delete certain lines from a table:

```
DELETE table1 WHERE Condition
```

NB: by adding `OUTPUT deleted.*` before the `WHERE`, we get the deleted content instead of the deleted lines

number.}}

Research a table

To research a table which we know the exact name, on all the server databases:

```
sp_MSforeachdb 'USE ?
IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id = OBJECT_ID(N''[MyTable]'') AND OBJECTPROPERTY(id, N''I
BEGIN
  PRINT ''Table found in the database: ?''
END'
```

Research in all the tables

SSMS 10 doesn't propose any research function, like one could find it in phpMyAdmin for MySQL for example.

Table research

This script passes through each database to return the tables which names contain the specified string of characters (at the end):

```
ALTER Proc FindTable
@TableName nVarchar(50)
As
/*
Purpose : Search for a Table in all databases
Author : Sandesh Segu
Date : 17th July 2009
Version : 1.0
More Scripts  : http://sanssql.blogspot.com
*/
ALTER Table #temp (DatabaseName varchar(50),SchemaName varchar(50),TableName varchar(50))

Declare @SQL Varchar(500)
Set @SQL='Use [?] ;
if exists(Select name from sys.tables where name like '''+@TableName+''')
insert into #temp
Select ''?'' AS DatabaseName ,SS.Name AS SchemaName ,ST.Name AS TableName from sys.tables as ST , sys.sc
where ST.Schema_ID=SS.Schema_ID and ST.name like '''+@TableName+''''

EXEC sp_msforeachdb @SQL

Select * from #temp

Drop table #temp
GO

/*
Usage: If the exact table name is known then specify the table name else include the wild cards
EXEC FindTable 'Employee'*/
EXEC FindTable '%String of characters to research%'
```

Research a value

A field value research in all the tables take a few time[5]:

```
CREATE TABLE #result(
  id      INT IDENTITY,
  tblName VARCHAR(255),
  colName VARCHAR(255),
  qtRows  INT
```

```
)
go

DECLARE @toLookFor VARCHAR(255)
SET @toLookFor = '%String of characters%'

DECLARE cCursor CURSOR LOCAL FAST_FORWARD FOR
SELECT
  '[' + usr.name + '].[' + tbl.name + ']' AS tblName,
  '[' + col.name + ']' AS colName,
  LOWER(typ.name) AS typName
FROM
  sysobjects tbl
    INNER JOIN(
      syscolumns col
        INNER JOIN systypes typ
        ON typ.xtype = col.xtype
    )
    ON col.id = tbl.id
    --
    LEFT OUTER JOIN sysusers usr
    ON usr.uid = tbl.uid

WHERE tbl.xtype = 'U'
  AND LOWER(typ.name) IN(
        'char', 'nchar',
        'varchar', 'nvarchar',
        'text', 'ntext'
      )
ORDER BY tbl.name, col.colorder
--
DECLARE @tblName VARCHAR(255)
DECLARE @colName VARCHAR(255)
DECLARE @typName VARCHAR(255)

DECLARE @sql  NVARCHAR(4000)
DECLARE @crlf CHAR(2)

SET @crlf = CHAR(13) + CHAR(10)

OPEN cCursor
FETCH cCursor
INTO @tblName, @colName, @typName

WHILE @@fetch_status = 0
BEGIN
  IF @typName IN('text', 'ntext')
  BEGIN
    SET @sql = ''
    SET @sql = @sql + 'INSERT INTO #result(tblName, colName, qtRows)' + @crlf
    SET @sql = @sql + 'SELECT @tblName, @colName, COUNT(*)' + @crlf
    SET @sql = @sql + 'FROM ' + @tblName + @crlf
    SET @sql = @sql + 'WHERE PATINDEX(''%'' + @toLookFor + ''%'', ' + @colName + ') > 0' + @crlf
  END
  ELSE
  BEGIN
    SET @sql = ''
    SET @sql = @sql + 'INSERT INTO #result(tblName, colName, qtRows)' + @crlf
    SET @sql = @sql + 'SELECT @tblName, @colName, COUNT(*)' + @crlf
    SET @sql = @sql + 'FROM ' + @tblName + @crlf
    SET @sql = @sql + 'WHERE ' + @colName + ' LIKE ''%'' + @toLookFor + ''%''' + @crlf
  END

  EXECUTE sp_executesql
            @sql,
            N'@tblName varchar(255), @colName varchar(255), @toLookFor varchar(255)',
            @tblName, @colName, @toLookFor

  FETCH cCursor
  INTO @tblName, @colName, @typName
END

SELECT *
FROM #result
WHERE qtRows > 0
ORDER BY id
GO

DROP TABLE #result
go
```

References

1. https://msdn.microsoft.com/en-us/library/bb510625.aspx
2. https://msdn.microsoft.com/en-us/library/ms174979.aspx
3. https://msdn.microsoft.com/en-us/library/ms174335.aspx
4. https://msdn.microsoft.com/en-us/library/ms188365.aspx
5. http://stackoverflow.com/questions/591853/search-for-a-string-in-an-all-the-tables-rows-and-columns-of-a-db

Functions

min, max

The functions Min() and Max() respectively return the minimum and the maximum of one field list.

```
select min(Date) from Calendar where RDV = 'Important'
```

cast

Modify a variable type:

```
cast(Champ as decimal(12, 6)) -- otherwise '9' > '10'
```

convert

Modify a variable type in first parameter, and its length in second.

```
convert(varchar, Field1, 112)
convert(datetime, Field2, 112)       -- otherwise impossible to go through the calendar (eg: D + 1)
```

Attention: all the variable types are not compatible between them[1].

Problem examples:

```
select Date1
from Table1
where Date1 between '01/10/2013' and '31/10/2013'
```

Dates are not systematically recognized without `convert`. The solution is to store them in the `datetime` format:

```
select Date1
from Table1
where Date1 between convert(varchar,'20131001',112) and convert(varchar,'20131031',112)
```

On the other hand, if an above paragraph date is stored in `varchar` with slashes, it becomes mandatory to

reformat it to be able to compare.

Numerous date formats are available[2].

left, right, and substring

Allow to cut strings according to some of their characters positions[3].

```
select substring('13/10/2013 00:09:19', 7, 4) -- returns the hour character after the seventh, so "2013
```

For example with the slashes date case above:

```
select Date1
from Table1
where right(Date1, 4) + substring(Date1, 4, 2) + left(Date1, 2) between convert(varchar,'20131001',112)
```

replace and stuff

Search and replace: allow to replace some string characters according to their values[4].

For example, to update a given folder path[5] :

```
update Table1
SET Field1 = replace(Field1,'\Old_path\','\New_path\')
where Field1 like '%\Old_path\%'
```

isnull

Returns *true* the the variable is null.

```
select Field1 = case when isnull(@Column,'')='' then '*'  else @Column end
from Table1
```

Dates

Date format

The function `GETDATE` is used to get the current date. To get another date in the good format, it's necessary to use `CONVERT`:

```
select convert(smalldatetime, '2016-01-02', 121)
```

Date cut

The function `DATEPART` extracts a date part without specifying manually its position[6].

However, three functions allow to accelerate these extractions writing:

```
-- Day
select day(getdate())
-- Month
select month(getdate())
-- Year
select year(getdate())
-- Previous year
select str(year(getdate()) - 1)
```

Days addition and subtraction

Herewith two dates manipulation functions[7]:

- `DATEDIFF` calculates the interval between two dates[8].
- `DATEADD` returns the date resulting from another plus an interval[9].

```
-- Last day of the previous month
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE()),0))
-- Last day of the current month
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+1,0))
-- Last day of the previous month
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,GETDATE())+2,0))
```

Example:

```
SELECT DATEADD(s,-1,DATEADD(mm, DATEDIFF(m,0,'20150101'),0)) as date
```

gives:

```
date
2014-12-31 23:59:59.000
```

References

1. man CONVERT (https://technet.microsoft.com/en-us/library/aa226054%28v=sql.80%29.aspx)
2. http://stackoverflow.com/questions/74385/how-to-convert-datetime-to-varchar
3. man SUBSTRING (http://msdn.microsoft.com/en-us/library/ms187748.aspx)
4. man STUFF (http://msdn.microsoft.com/en-us/library/ms188043.aspx)
5. man REPLACE (http://msdn.microsoft.com/en-us/library/ms186862.aspx)
6. man DATEPART (http://msdn.microsoft.com/en-us/library/ms174420.aspx)
7. http://blog.sqlauthority.com/2007/08/18/sql-server-find-last-day-of-any-month-current-previous-next/
8. man DATEDIFF (http://msdn.microsoft.com/en-us/library/ms189794.aspx)
9. man DATEADD (http://msdn.microsoft.com/en-us/library/ms186819.aspx)

Best Practices

- Always qualify objects by owner.=
- Use query "with (nolock)" when you don't require high transactional consistency.
- Do not use GOTO.
- Avoid CURSOR use because it's significantly slower. If necessary, always declare the correct type of

cursor (FAST_FORWARD).
- Avoid SELECT INTO for populating temp tables. Create the table then use INSERT SELECT.
- Always use ANSI join syntax.
- Always check for object existence.
- Use SCOPE_IDENTITY() instead of @@IDENTITY.
- Always check @@TRANCOUNT and commit/rollback as necessary.
- Order DML to avoid deadlocks.
- Always check @@ERROR and @@ROWCOUNT by assigning to a variable.
- Always check sp return values.
- Do not create cross-database dependencies.
- Avoid table value UDF – performance problems.
- Avoid dynamic SQL – if necessary use sp_executesql over EXEC.
- Avoid using NULL values.
- When there are only two values, ISNULL is more efficient than COALESCE.
- Always specify columns; try to avoid "SELECT *". Exceptions include these two cases: "WHERE EXISTS (SELECT * ...)" and aggregate functions.

Geospatial Data

Introduction

Microsoft's 2008 release of SQL Server finally delivers Geospatial support to the SQL Server product suite.

This allows the storage of spatial data in SQL tables (in the form of points, lines and polygons) and a set of functions to allow the manipulation of this data. Also included are new spatial indexes to support the execution of these functions.

Example Database

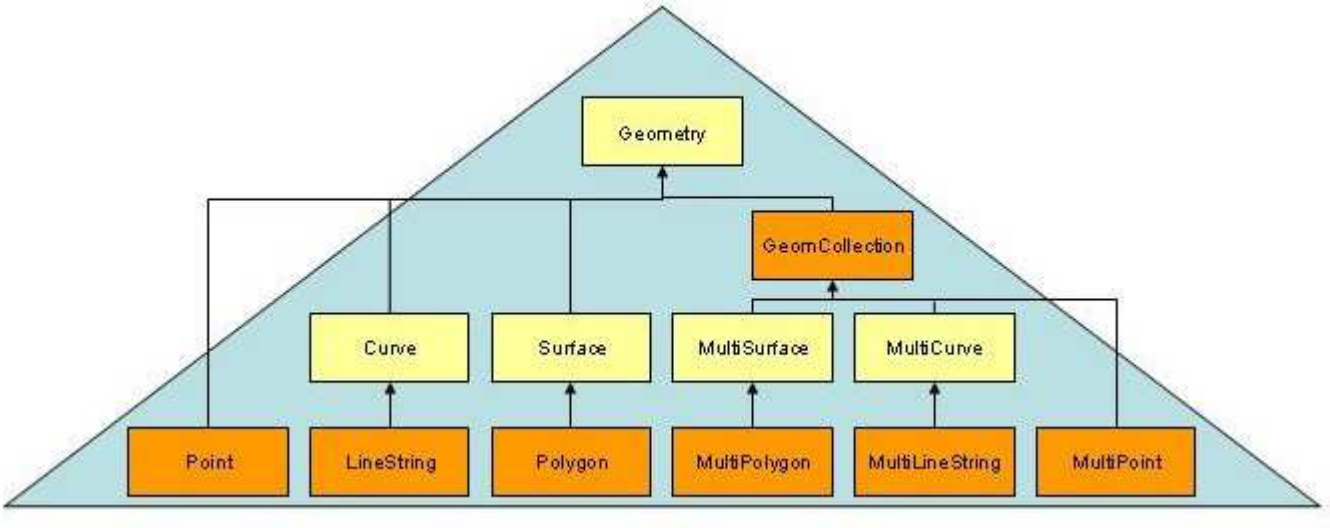
This book will use an example database for the majority of its examples. The code for creating this can be found in Appendix A and should be run, as per the instructions there, in order to be able use the example code throughout this book. What follows here is a brief description of that database and the data contained within. If you wish, then you can skip this section and get straight into the interesting stuff, working out what the database is doing as you go.

Geospatial Data Types

SQL Server 2008 supports two different spatial data types: GEOMETRY and GEOGRAPHY.

- GEOMETRY - This data type stores data in projected planar surfaces.
- GEOGRAPHY - This data type stores data in an ellipsoidal model.

The Geometry Classes define a hierarchy as follows:



From the diagram we can see that there are seven types of instantiable spatial data types (in orange). Namely:

Point

A point is an object representing a single location. It always has an X and Y co-ordinate and may additionally have an elevation Z and a measure M.

MultiPoint

A MultiPoint object is a collection of points. It differs from a LineString and a Polygon as there is no implied connections between the points in the collection. Because of this the boundary of a MultiPoint object is empty.

LineString

A LineString is again a collection of points. However this differs from the Multipoint object, as the points are in sequence and the LineString object also represents the line segments connecting the points.

MultiLineString

A MultiLineString is simply a collection of LineStrings.

Polygon

A Polygon is a collection of points representing a two dimensional surface. A Polygon may consist of a exterior ring and a number of interior rings. For a Polygon object to be a valid instance the interior rings cannot cross one another.

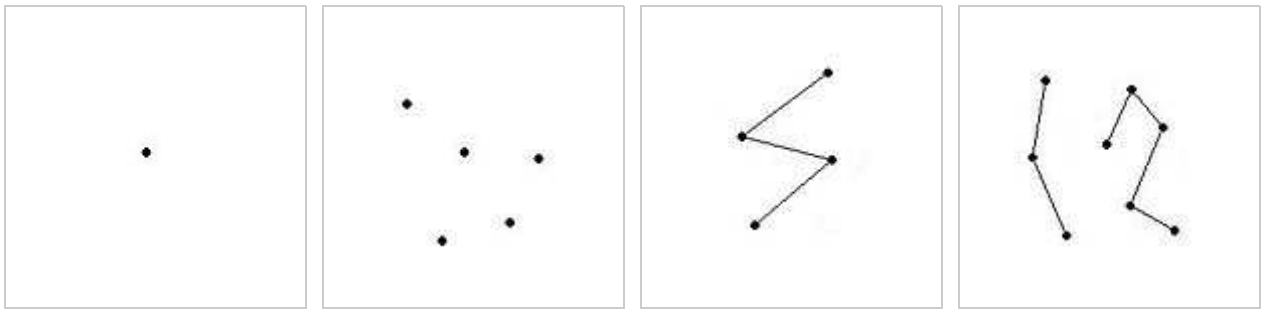
MultiPolygon

A MultiPolygon is a collection of Polygons.

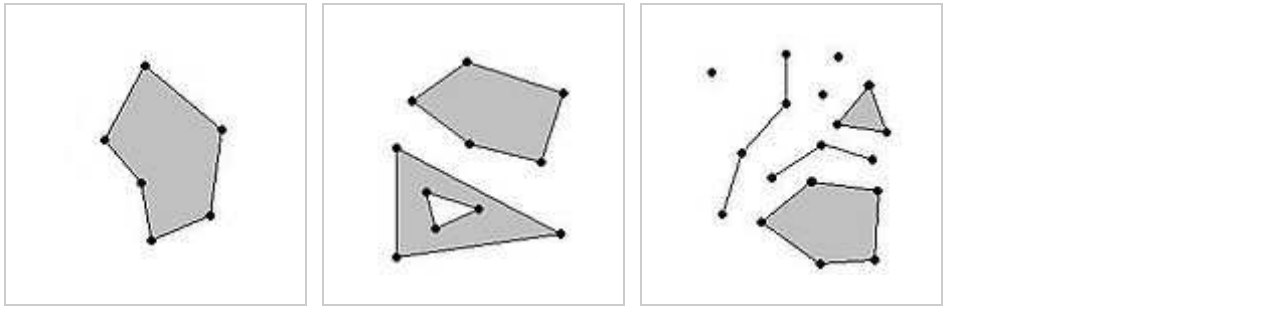
GeometryCollection

A GeometryCollection is a collection of geometry (or geography) objects.

Visually



Point | MultiPoint | LineString | MultiLineString



Polygon | MultiPolygon | GeometryCollection

Importing GeoSpatial Data into SQL Server

Data can be imported directly into SQL server from the following formats: WKT, WKB and GML. For more detail of these formats please see the appendices.

So I can't import my data from shape files?

Currently you cannot do this directly, whether this will change in future versions remains to be seen. If you have your data in shape files, or other formats then you will need to find a way to convert it. There are a number of Microsoft Spatial partners who offer tools which will allow you to do this, but obviously this will involve another license fee. Free versions I am sure will begin to appear as SQL Server 2008 releases. Currently Morten Nielsen has such a free tool posted on his blog here [[1] (http://www.sharpgis.net/2007/12/23/ShapefileToSqlServer2008ToWPF.aspx)] and Tillmann Eitelberg has published a SSIS shape file source under Ms-PL on CodePlex [[2] (http://shapefilesource.codeplex.com/)]. AutoConViz (by Sugam Sharma, U Sunday Tim and Shashi Gadia) also offers the GUI based spatial format conversion (shape file format to GML) as one of its core functions[[3] (http://www.tandfonline.com/doi/abs/10.1080/10095020.2012.714099#.UbSoMvm1G80)].

Import Functions

Importing spatial data into SQL Server 2008 is done via means of the ST*xxx*from*yyy* set of functions, where *xxx* can be one of the following:

- Geom - For importing any spatial data type.
- Point - For importing point data.

- Line - For importing line data.
- Poly - For importing polygon data.

- GeomColl - For importing a Geometry Collection.
- MPoint - For importing multi point data.
- MLine - For importing multi line data.
- MPoly - For importing multi poly data.

and *yyy* can be

- Text - For importing data in WKT format.
- WKB - For importing data in WKB format.
- Gml - For importing data in GML format. (Note the case of Gml is important and this function lacks the leading ST e.g. GeomfromGml(.....))

Example

```
CREATE TABLE Districts
( DistrictId int IDENTITY (1,1),
DistrictName nvarchar(20),
DistrictGeo geometry);
GO

CREATE TABLE Streets
( StreetId int IDENTITY (1,1),
StreetName nvarchar(20),
StreetGeo geometry);
GO

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Downtown',
geometry::STGeomFromText
('POLYGON ((0 0, 150 0, 150 150, 0 150, 0 0))', 0));

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Green Park',
geometry::STGeomFromText
('POLYGON ((300 0, 150 0, 150 150, 300 150, 300 0))', 0));

INSERT INTO Districts (DistrictName, DistrictGeo)
VALUES ('Harborside',
geometry::STGeomFromText
('POLYGON ((150 0, 300 0, 300 300, 150 300, 150 0))', 0));

INSERT INTO Streets (StreetName, StreetGeo)
VALUES ('First Avenue',
geometry::STGeomFromText
('LINESTRING (100 100, 20 180, 180 180)', 0))
GO

INSERT INTO Streets (StreetName, StreetGeo)
VALUES ('Mercator Street',
geometry::STGeomFromText
('LINESTRING (300 300, 300 150, 50 51)', 0))
GO
```

Geospatial SQL functions

- STRelate
- STDisjoint
- STIntersects

- STTouches
- STCrosses
- STWithin
- STContains
- STOverlaps
- STBuffer
- STConvexHull
- STIntersection
- STUnion
- STGeomFromText
- STPointFromText
- STLineFromText
- STPolyFromText
- STGeomFromWKB
- STPointFromWKB
- STLineFromWKB
- STPolyFromWKB

Appendices

A. Example Database Code

B. WKT (Well Known Text) format definition

C. GML (Geographic Markup Language) format definition

References

- Directions Magazine's interview with Ed Katibah [4] (http://www.directionsmag.com/editorials.php?article_id=2477&trv=1)
- Isaac Kunen's Blog [5] (http://blogs.msdn.com/isaac/archive/2007/05/29/one-type-two-types.aspx)
- Microsoft Spatial Forum [6] (http://social.msdn.microsoft.com/Forums/en-US/sqlspatial/threads)
- Microsoft SQL Server 2008 Spatial Data Overview [7] (http://www.microsoft.com/sqlserver/2008/en/us/spatial-data.aspx)
- Microsoft SQL Server 2008 Spatial Data Datasheet [8] (http://download.microsoft.com/download/c/8/4/c8470f54-d6d2-423d-8e5b-95ca4a90149a/SQLServer2008_SpatialData_Datasheet.pdf)
- Microsoft SQL Server 2008 Spatial Data White Paper [9] (http://download.microsoft.com/download/a/c/d/acd8e043-d69b-4f09-bc9e-4168b65aaa71/SpatialData.doc)
- Microsoft Books Online [10] (http://msdn2.microsoft.com/en-gb/library/bb933790(SQL.100).aspx)
- Wikipedia [11] (http://wikipedia.org)
- AutoConViz for GUI based spatial format conversion online[12] (http://www.tandfonline.com/doi/abs/10.1080/10095020.2012.714099#.UbSoMvm1G80)

Geospatial Data/Example Database Code

```
USE [master]
GO
```

```
/****** Object:  Database [Katmai]    Script Date: 05/26/2009 07:23:33 ******/
CREATE DATABASE [Katmai] ON  PRIMARY
( NAME = N'Katmai', FILENAME = N'c:\Program Files\Microsoft SQL Server\MSSQL10.SQLEXPRESS\MSSQL\DATA\Kat
 LOG ON
( NAME = N'Katmai_log', FILENAME = N'c:\Program Files\Microsoft SQL Server\MSSQL10.SQLEXPRESS\MSSQL\DATA
GO

ALTER DATABASE [Katmai] SET COMPATIBILITY_LEVEL = 100
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Katmai].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [Katmai] SET ANSI_NULL_DEFAULT OFF
GO

ALTER DATABASE [Katmai] SET ANSI_NULLS OFF
GO

ALTER DATABASE [Katmai] SET ANSI_PADDING OFF
GO

ALTER DATABASE [Katmai] SET ANSI_WARNINGS OFF
GO

ALTER DATABASE [Katmai] SET ARITHABORT OFF
GO

ALTER DATABASE [Katmai] SET AUTO_CLOSE OFF
GO

ALTER DATABASE [Katmai] SET AUTO_CREATE_STATISTICS ON
GO

ALTER DATABASE [Katmai] SET AUTO_SHRINK OFF
GO

ALTER DATABASE [Katmai] SET AUTO_UPDATE_STATISTICS ON
GO

ALTER DATABASE [Katmai] SET CURSOR_CLOSE_ON_COMMIT OFF
GO

ALTER DATABASE [Katmai] SET CURSOR_DEFAULT  GLOBAL
GO

ALTER DATABASE [Katmai] SET CONCAT_NULL_YIELDS_NULL OFF
GO

ALTER DATABASE [Katmai] SET NUMERIC_ROUNDABORT OFF
GO

ALTER DATABASE [Katmai] SET QUOTED_IDENTIFIER OFF
GO

ALTER DATABASE [Katmai] SET RECURSIVE_TRIGGERS OFF
GO

ALTER DATABASE [Katmai] SET  DISABLE_BROKER
GO

ALTER DATABASE [Katmai] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO

ALTER DATABASE [Katmai] SET DATE_CORRELATION_OPTIMIZATION OFF
GO

ALTER DATABASE [Katmai] SET TRUSTWORTHY OFF
GO

ALTER DATABASE [Katmai] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO

ALTER DATABASE [Katmai] SET PARAMETERIZATION SIMPLE
GO

ALTER DATABASE [Katmai] SET READ_COMMITTED_SNAPSHOT OFF
```

```
GO

ALTER DATABASE [Katmai] SET HONOR_BROKER_PRIORITY OFF
GO

ALTER DATABASE [Katmai] SET  READ_WRITE
GO

ALTER DATABASE [Katmai] SET RECOVERY SIMPLE
GO

ALTER DATABASE [Katmai] SET  MULTI_USER
GO

ALTER DATABASE [Katmai] SET PAGE_VERIFY CHECKSUM
GO

ALTER DATABASE [Katmai] SET DB_CHAINING OFF
GO

USE [Katmai]
GO

/****** Object:  Table [dbo].[ParkBoundaries]    Script Date: 05/26/2009 07:24:49 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[ParkBoundaries](
    [ID] [int] NOT NULL PRIMARY KEY,
    [Name] [nvarchar](255) NULL,
        [BoundaryType] [nvarchar](255) NULL,
    [geom] [geometry] NULL
)

GO

INSERT INTO [Katmai].[dbo].[ParkBoundaries]
           (ID,
           [BoundaryType]
           ,[Name]
           ,[geom])
     VALUES
 (970, 'Katmai National Preserve', 'National Preserve NPS', Geometry::STPolyFromText('POLYGON ((-155.590
,(982, 'Katmai National Park', 'National Park NPS', Geometry::STPolyFromText('POLYGON ((-155.11517333984
,(987, 'Katmai National Park', 'National Park NPS', Geometry::STPolyFromText('POLYGON ((-154.16101074218
,(991, 'Katmai Wilderness', 'Wilderness NPS', Geometry::STPolyFromText('POLYGON ((-155.85282897949219 59

,(1013, 'Katmai National Park', 'National Park NPS', Geometry::STPolyFromText('POLYGON ((-155.5207672119

,(1031, 'Katmai Wilderness', 'Wilderness NPS', Geometry::STPolyFromText('POLYGON ((-155.95281982421875 5
```

Geospatial Data/GML

The **Geography Markup Language** (**GML**) is the XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. Note that the concept of feature in GML is a very general one and includes not only conventional "vector" or discrete objects, but also coverages (see also GMLJP2) and sensor data. The ability to integrate all forms of geographic information is key to the utility of GML.

Standards

The OGC is an international voluntary consensus standards organization whose members maintain the *Geography Markup Language* standard. The OGC coordinates with the ISO TC 211 standards organization

to maintain consistency between OGC and ISO standards work. GML is in the process of being adopted as an ISO standard (ISO 19136) and is expected to be released as an International Standard in 2007.

GML is the XML data standard for the GeoWeb infrastructure, enabling Internet-connected devices to access geographical information, including, for example, merchant locations and traffic conditions.

GML can also be included in version 1.0 of the United States National Information Exchange Model.

GML model

The original GML model was based on the World Wide Web Consortium's Resource Description Framework (RDF). Subsequently, the OGC introduced XML schemas into GML's structure to help connect the various existing geographic databases, whose relational structure XML schemas more easily define. The resulting XML-schema-based GML retains many features of RDF, including the idea of child elements as properties of the parent object (RDFS) and the use of remote property references.

GML contains a rich set of primitives which are used to build application specific schemas or application languages. These primitives include:

- Feature
- Geometry
- Coordinate Reference System
- Time
- Dynamic feature
- Coverage (including geographic images)
- Unit of measure
- Map presentation styling rules

Profile

GML profiles are logical restrictions to GML, and may be expressed by a document, an XML schema or both. These profiles are intended to simplify adoption of GML, to facilitate rapid adoption of the standard. The following *profiles*, as defined by the GML specification, have been published or proposed for public use:

- A *Point Profile* for applications with point geometric data but without the need for the full GML grammar
- A *GML Simple Features profile* supporting vector feature requests and transactions, e.g. with a WFS
- A GML profile for GMJP2 (GML in JPEG 2000)
- A GML profile for RSS

Note that *Profiles* are distinct from application schemas. *Profiles* are part of GML namespaces and define restricted subsets of GML. Application schemas are XML vocabularies defined using GML and which live in an application-defined target namespace. Application schemas can be built on specific GML profiles or use the full GML schema set.

Profiles are often created in support for GML derived languages (see application schemas) created in support of particular application domains such as commercial aviation, nautical charting or resource exploitation.

The GML Specification (Since GML v3.) contains a pair of XSLT scripts (usually referred to as the "subset tool") that can be used to construct GML profiles.

GML Simple Features Profile

The *GML Simple Features Profile* is a more complete profile of GML than the above *Point Profile* and supports a wide range of vector feature objects, including the following:

1. A reduced geometry model allowing 0d, 1d and 2d linear geometric objects (all based on linear interpolation) and the corresponding aggregate geometries (gml:MultiPoint, gml:MultiCurve, etc.).
2. A simplified feature model which can only be one level deep (in the general GML model, arbitrary nesting of features and feature properties is not permitted).
3. All non-geometric properties must be XML Schema simple types – i.e. cannot contain nested elements.
4. Remote property value references (xlink:href) just like in the main GML specification.

Since the profile aims to provide a simple entry point, it does not provide support for the following:

- coverages
- topology
- observations
- value objects (for real time sensor data)
- nor support for dynamic features.

Nonetheless it supports a good variety of real world problems.

Subset tool

In addition, the GML specification provides a *subset tool* to generate GML profiles containing a user-specified list of components. The tool consists of a pair of XSLT scripts. The scripts generate a profile that a developer may extend manually or otherwise enhance through schema restriction. Note that as restrictions of the full GML specification, application schemas that a profile can generate must themselves be valid GML application schemas.

The subset tool can generate profiles for many other reasons as well. Listing the elements and attributes to include in the resultant profile schema and running the tool results in a single profile schema file containing only the user-specified items and all of the element, attribute and type declarations on which the specified items depend. Some Profile schemas created in this manner support other specifications including IHO S-57 and GML in JPEG 2000.

Application schema

In order to expose an application's geographic data with GML, a community or organization creates an XML schema specific to the application domain of interest(the *application schema*). This schema describes the object types whose data the community is interested in and which community applications must expose. For example, an application for tourism may define object types including monuments, places of interest, museums, road exits, and viewpoints in its *application schema*. Those object types in turn reference the primitive object types defined in the GML standard.

A list of known publicly available GML Application Schemas is being assembled.

Some other markup languages for geography use schema constructs, but GML builds on the existing XML schema model instead of creating a new schema language.

GML and KML

Note that the KML (Keyhole Markup Language) language made popular by Google is a complementary to GML. Whereas GML is a language to encode geographic content, by describing a spectrum of application objects and their properties (e.g. bridges, roads, buoys, vehicles etc.), KML is a language for the visualization of geographic information. KML can be used to carry GML content, and GML can be "styled" to KML for the purposes of presentation.

GML geometries

GML encodes the *GML geometries*, or *geometric characteristics*, of geographic objects as elements within GML documents. The geometries of those objects may describe, for example, roads, rivers, and bridges.

The key GML geometry object types in GML 1.0 and GML 2.0, are the following:

- Point
- LineString
- Polygon

Note that this geometry model is identical to the geometry model in KML.

Features

GML defines *features* distinct from *geometry objects*. A *feature* is an application object that represents a physical entity, e.g. a building, a river, or a person. A *feature* may or may not have geometric aspects. A *geometry object* defines a location or region instead of a physical entity, and hence is different from a *feature*. The distinction between *features* and *geometry objects* in GML contrasts with models used in other geographic information systems (GIS) that make no such distinction. That is, although some other GIS define *features* and *geometry objects* interchangeably as items on a map, GML maintains them as separate entity types.

In GML, a *feature* can have various geometry properties that describe geometric aspects or characteristics of the feature (e.g. the feature's *Point* or *Extent* properties). GML also provides the ability for *features* to share a geometry property with one another by using a *remote property reference* on the shared geometry property. Remote properties are a general feature of GML borrowed from RDF. An *xlink:href* attribute on a GML geometry property means that the value of the property is the resource referenced in the link.

For example, a *Building* feature in a particular GML application schema might have a position given by the primitive GML geometry object type *Point*. However, the *Building* is a separate entity from the *Point* that defines its position. In addition, a *feature* may have several geometry properties (or none at all), for example an *extent* and a *position*.

Coordinates

Coordinates in GML represent the coordinates of *geometry objects*. Coordinates can be specified by any of the following GML elements:

- *<gml:coordinates>*
- *<gml:pos>*
- *<gml:posList>*

GML has multiple ways to represent coordinates. For example, the *<gml:coordinates>* element can be used, as follows:

```
<gml:Point gml:id="p21" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
```

```
   <gml:coordinates>45.67, 88.56</gml:coordinates>
</gml:Point>
```

Note that, when expressed as above, the individual coordinates (e.g. *88.56*) are not separately accessible through the XML Document Object Model since the content of the <gml:coordinates> element is just a single string.

To make GML coordinates accessible through the XML DOM, GML 3.0 introduced the *<gml:pos>* and *<gml:posList>* elements. (Note that although GML versions 1 and 2 had the *<gml:coord>* element, it is treated as a defect and is not used.) Using the *<gml:pos>* element instead of the *<gml:coordinates>* element, the same point can be represented as follows:

```
<gml:Point gml:id="p21" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
   <gml:pos dimension="2">45.67 88.56</gml:pos>
</gml:Point>
```

The coordinates of a *<gml:LineString>* geometry object can be represented with the *<gml:coordinates>* element:

```
<gml:LineString gml:id="p21" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
   <gml:coordinates>45.67, 88.56 55.56,89.44</gml:coordinates>
</gml:LineString >
```

The *<gml:posList>* element is used to represent a list of coordinate tuples, as required for linear geometries:

```
<gml:LineString gml:id="p21" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
   <gml:posList dimension="2">45.67 88.56 55.56 89.44</gml:posList>
</gml:LineString >
```

For GML data servers (WFS) and conversion tools that only support GML 1 or GML 2 (i.e. only the <gml:coordinates> element), there is no alternative to *<gml:coordinates>*. For GML 3 documents and later, however, *<gml:pos>* and *<gml:posList>* are preferable to *<gml:coordinates>*.

For more information on the *srsName* attribute, see *Coordinate Reference System* below.

Coordinate Reference System

A **Coordinate Reference System** (CRS) determines the geometry of each geometry element in a GML document.

Unlike KML or GeoRSS, GML does not default to a coordinate system when none is provided. Instead, the desired coordinate system must be specified explicitly with a *Coordinate Reference System* (CRS) or Spatial Reference System (SRS). The elements whose coordinates are interpreted with respect to such a CRS include the following:

- <gml:coordinates>
- <gml:pos>
- <gml:posList>

An *srsName* attribute attached to a geometry object specifies the object's CRS, as shown in the following example:

```
<gml:Point gml:id="p1" srsName="#srs36">
```

```
    <gml:coordinates>100,200</gml:coordinates>
</gml:Point>
```

The value of the *srsName* attribute is a Uniform Resource Identifier (URI). It refers to a definition of the Coordinate Reference System that is used to interpret the coordinates in the geometry. The CRS definition may be in a document (i.e. a *flat file*) or in an online web service.

The *srsName* URI may also be a Uniform Resource Name (URN) for referencing a common CRS definition. The OGC has developed a URN structure and a set specific URNs to encode some common Coordinate Reference Systems. A URN resolver resolves those URNs to GML CRS definitions.

Examples

Polygons, Points, and *LineString* objects are encoded in GML 1.0 and 2.0 as follows:

```
    <gml:Polygon>
        <gml:outerBoundaryIs>
                <gml:LinearRing>
                        <gml:coordinates>0,0 100,0 100,100 0,100 0,0</gml:coordinates>
                </gml:LinearRing>
       </gml:outerBoundaryIs>
    </gml:Polygon>
    <gml:Point>
       <gml:coordinates>100,200</gml:coordinates>
    </gml:Point>
    <gml:LineString>
       <gml:coordinates>100,200 150,300</gml:coordinates>
    </gml:LineString>
```

Note that *LineString* objects, along with *LinearRing* objects, assume linear interpolation between the specified points.

Features using geometries

The following GML example illustrates the distinction between *features* and *geometry objects*. The *Building* feature has several *geometry objects*, sharing one of them (the *Point* with identifier *p21*) with the *SurveyMonument* feature:

```
<abc:Building gml:id="SearsTower">
    <gml:name>Sears Tower</gml:name>
    <abc:height>52</abc:height>
    <abc:position>
        <gml:Point>
            <gml:coordinates>100,200</gml:coordinates>
        </gml:Point>
    </abc:position>
    <app:extent>
        <gml:Polygon>
            <gml:exterior>
                <gml:LinearRing>
                    <gml:coordinates>100,200</gml:coordinates>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </app:extent>
</abc:Building>
<abc:Building gml:id="SearsTower">
    <abc:position xlink:type="Simple" xlink:href="#p21"/>
</abc:Building>
<abc:SurveyMonument gml:id="g234">
    <abc:position>
        <gml:Point gml:id="p21">
            <gml:coordinates>100,200</gml:coordinates>
        </gml:Point>
```

```
    </abc:position>
</abc:SurveyMonument>
```

Note that the reference is to the shared *Point* and not to the *SurveyMonument*, since any *feature* object can have more than one *geometry object* property.

Point Profile

The GML *Point Profile* contains a single GML geometry, namely a <*gml:Point*> object type. Any XML Schema can use the *Point Profile* by importing it and referencing the subject <*gml:Point*> instance:

```
<PhotoCollection xmlns="http://www.myphotos.org" xmlns:gml="http://www.opengis.net/gml"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.myphotos.org
     MyGoodPhotos.xsd">
    <items>
        <Item>
            <name>Lynn Valley</name>
            <description>A shot of the falls from the suspension bridge</description>
            <where>North Vancouver</where>
            <position>
                <gml:Point srsDimension="2" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
                       <gml:pos>49.40 -123.26</gml:pos>
                 </gml:Point>
             </position>
        </Item>
    </items>
</PhotoCollection>
```

Note that when using the *Point Profile*, the only geometry object is the '<gml:Point>' object. The rest of the geography is defined by the photo-collection schema.

Geospatial Data/WKT

Well-known text (WKT) is a text markup language for representing vector geometry objects on a map, spatial reference systems of spatial objects and transformations between spatial reference systems. A binary equivalent, known as **well-known binary (WKB)** is used to transfer and store the same information on databases, such as PostGIS. The formats are regulated by the Open Geospatial Consortium (OGC) and described in their Simple Feature Access and Coordinate Transformation Service specifications.

Geometric Objects

Geometric objects that can be represented with WKT are: points, lines, polygons, TINs and Polyhedron]. Multi geometries are also available to represent more than one geometry of the same dimension in a single object, and geometries of different dimensions can be stored in a geometry collection.

Coordinates for geometries may be 2D (x, y), 3D (x, y, z), 4D (x, y, z, m) with a m value that is part of a linear reference system or 3D with a m value (x, y, m). Three dimensional geometries are designated by a Z after the geometry type and geometries with a linear reference system have a M after the geometry type.

WKT geometries are used throughout OGC specifications and are present in applications that implement these specifications. For example, PostGIS contains functions that can convert geometries to and from a WKT representation, making them human readable.

The following are some example geometric WKT strings.

```
POINT(6 10)
LINESTRING(3 4,10 50,20 25)
POLYGON((1 1,5 1,5 5,1 5,1 1),(2 2, 3 2, 3 3, 2 3,2 2))
MULTIPOINT(3.5 5.6,4.8 10.5)
MULTILINESTRING((3 4,10 50,20 25),(-5 -8,-10 -8,-15 -4))
MULTIPOLYGON(((1 1,5 1,5 5,1 5,1 1),(2 2, 3 2, 3 3, 2 3,2 2)),((3 3,6 2,6 4,3 3)))
GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))
POINT ZM (1 1 5 60)
POINT M (1 1 80)
```

Spatial Reference Systems

A WKT string for a spatial reference system describes the datum, geoid, coordinate system, and map projection of the spatial objects.

Well-known text is used extensively throughout many GIS programs. ESRI uses WKT in the shapefile `*.prj` file.

The following is an example of a spatial reference system WKT string.

```
COMPD_CS["OSGB36 / British National Grid + ODN",
    PROJCS["OSGB 1936 / British National Grid",
        GEOGCS["OSGB 1936",
            DATUM["OSGB_1936",
                SPHEROID["Airy 1830",6377563.396,299.3249646,AUTHORITY["EPSG","7001"]],
                TOWGS84[375,-111,431,0,0,0,0],
                AUTHORITY["EPSG","6277"]],
            PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
            UNIT["DMSH",0.0174532925199433,AUTHORITY["EPSG","9108"]],
            AXIS["Lat",NORTH],
            AXIS["Long",EAST],
            AUTHORITY["EPSG","4277"]],
        PROJECTION["Transverse_Mercator"],
        PARAMETER["latitude_of_origin",49],
        PARAMETER["central_meridian",-2],
        PARAMETER["scale_factor",0.999601272],
        PARAMETER["false_easting",400000],
        PARAMETER["false_northing",-100000],
        UNIT["metre",1,AUTHORITY["EPSG","9001"]],
        AXIS["E",EAST],
        AXIS["N",NORTH],
        AUTHORITY["EPSG","27700"]],
    VERT_CS["Newlyn",
        VERT_DATUM["Ordnance Datum Newlyn",2005,AUTHORITY["EPSG","5101"]],
        UNIT["metre",1,AUTHORITY["EPSG","9001"]],
        AXIS["Up",UP],
        AUTHORITY["EPSG","5701"]],
    AUTHORITY["EPSG","7405"]]
```

Transformations

A WKT format is defined to describe the transformation methods and parameters used to convert coordinates between two different spatial reference systems.

Below are two examples of WKT transformation descriptions.

```
PARAM_MT["Mercator_2SP",
    PARAMETER["semi_major",6370997.0],
    PARAMETER["semi_minor",6370997.0],
    PARAMETER["central_meridian",180.0],
    PARAMETER["false_easting",-500000.0],
    PARAMETER["false_northing",-1000000.0],
    PARAMETER["standard_parallel_1",60.0]]
```

```
PARAM_MT["Affine",
    PARAMETER["num_row",3],
    PARAMETER["num_col",3],
    PARAMETER["elt_0_1",1],
    PARAMETER["elt_0_2",2],
    PARAMETER["elt_1_2",3]]
```