Pika

Posebna e-izdaja
Marec 2011
Letnik XIV

Kompas Xnet – Pot do vašega znanja.

ISSN: 1408-7863

kompas xnet
&
SOLID QUALITY MENTORS

Kompas Xnet d.o.o.
Stegne 7, 1000 Ljubljana
01 5136 990
info@kompas-xnet.si
http://www.kompas-xnet.si

V Kompas Xnet-u že od ustanovitve dalje posvečamo izjemno skrb kvaliteti svojih storitev. Da bi lahko vam, našim cenjenim strankam, zagotovili najboljše storitve in rešitve, v svoji ekipi združujemo le vrhunske strokovnjake za posamezna področja.

Pred vami je posebna izdaja PiKE, s katero želimo izpostaviti zelo pomemben dogodek v našem podjetju:

Dejan SARKA, Dino ESPOSITO, Herbert ALBERT in ostali vrhunski predavatelji iz Solid Quality Mentors, so se pridružili naši ekipi.

Dokaz več, da smo ta hip zanesljivo najboljši izobraževalni center v Sloveniji. In vi si zares zaslužite najboljše.

Za to priložnost so se Dejan, Dino in Herbert posebej potrudili in napisali zanimive članke, ki jih boste zagotovo z navdušenjem prebrali. Spoznali boste, kako lahko preprosto razložiš tudi zelo zahtevne vsebine. To pa znajo le največji mojstri! Tudi v prihodnje bodo kolegi iz SolidQ prispevali svoje članke in z njimi obogatili redne in izredne izdaje naše in vaše PiKE.

Mimogrede: veste, da PiKA izhaja že 14. leto. Izdajamo jo 5x letno, v nakladi preko 6200 izvodov in ima skoraj 6000 rednih naročnikov.

Na naslednjih straneh boste našli še pregled aktualnih seminarjev, ki jih bodo izvajali ti vrhunski predavatelji.

Vabimo vas, da se čim prej prijavite in si zagotovite prosto mesto. Prisrčno vabljeni!

Mi izjemno cenimo vaš čas, ki ga boste investirali v svoje znanje, zato pri nas ne boste razočarani.

Solid Quality Mentors (SolidQ) združuje več kot 100 vrhunskih tehničnih strokovnjakov. Od leta 2002 je SolidQ postala zaupanja vreden globalni ponudnik naprednih svetovalnih, mentorskih in izobraževalnih rešitev za podatkovne, Business Intelligence, sodelovalne in razvojne platforme. SolidQ ponudba rešitev, izobraževanj in ad-hoc mentorstva so vzvodi za pospeševanje rezultatov vaših IT investicij.

Samoumevno je, da se vsi naši sodelavci nenehno izobražujejo in izpopolnjujejo, da so lahko v koraku z najsodobnejšimi IT tehnologijami, hkrati pa si nabirajo praktične izkušnje z delom na projektih.

Čeprav smo majhno podjetje, imamo specialiste za vsa ključna področja:

- **SharePoint: Robi Vončina, Uroš Žunič**
- **Infrastruktura (Windows Server, Exchange, ForeFront …): Jože Markič**
- **Azure: Rok Bermež**
- **Razvoj: Rok Bermež, Uroš Žunič**
- **SQL: Dejan Sarka & SolidQ, Rok Bermež**

Veseli smo, da se širi dober glas o nas in nam prinaša vedno več naročil za bolj ali manj zahtevne projekte: migracije, nadgradnje, virtualizacija, namestitve in konfiguracije, poslovne in spletne aplikacije, aplikacije »za oblak« …

Vedno pogosteje se na nas obračajo stranke, ki potrebujejo nujno pomoč za akutno težavo na svoji infrastrukturi – naši strokovnjaki so zanesljivi in učinkoviti. Prepričajte se sami.

Hvala za zaupanje.

Veselimo se sodelovanja z vami

Kazalo

Vsi programski jeziki SQL Serverja

Ko pomislimo na Microsoft SQL Server, ne glede na verzijo, najprej pomislimo na programski jezik Transact-SQL (T-SQL). Seveda je to najbolj pomemben jezik za delo s tem sistemom. Vendar pa SQL Server ne vsebuje samo relacijske baze; v paketu je še kar nekaj produktov, predvsem produktov poslovne inteligence. Vsak od teh produktov uporablja nek svoj jezik. Pa še osnovni del, torej relacijska baza, podpira kaj več kot samo T-SQL. Tale članek je sorazmerno preprost pregled programskih jezikov, ki nam lahko pridejo prav pri delu s produkti iz paketa SQL Server 2008 R2.

Relacijska baza (Database Engine)

Že samo relacijski del nam postreže s kopico jezikov. Posebej se je nabor razširil pri verziji 2005, ko je prišla intenzivna podpora za XML in .NET.

Transact-SQL (T-SQL)

Najbolj pomemben, osrednji jezik za delo s SQL Serverjem je T-SQL. To je Microsoftova različica standardnega ANSI Structured Query Language (SQL) jezika. Aplikacije komunicirajo z bazo preko T-SQL ukazov. Vse delo s podatki, tudi če gre za CLR proceduro znotraj SQL Serverja, gre preko T-SQL. Tudi elemente, napisane v drugih jezikih, mora aplikacija poslati preko T-SQL ukazov, torej znotraj T_SQL jezika.

T-SQL je kot vsi SQL jeziki v osnovi deklarativen jezik. To pomeni, da programska koda ne opisuje poteka aplikacije, kako naj se kaj izvede, ampak opisuje, kaj naj bi aplikacija izvedla. Ti opisi temeljijo na formalni logiki. Namesto, da bi pisali podrobno kodo izvajanja, uporabljamo velike gradnike, SQL stavke. Algoritmi izvajanja so skriti v SQL stavkih. V naslednji poizvedbi zahtevamo vsoto prodaje po kategorijah, podkategorijah in produktih za leto 2008, prvo četrtletje. Primer dela na testni bazi AdventureWorksDW2008R2, ki jo, kot druge demo baze in projekte, lahko naložimo s spletnega naslova http://msftdbprodsamples.codeplex.com/.

Kot vidimo, nismo nikjer opisali, kako naj SQL Server izvede povezavo (JOIN) med različnimi tabelami, kako naj izvede filter (WHERE), ali kako naj izvede grupiranje (GROUP BY) podatkov. Detaljni algoritmi so skriti v ukazih, izvajanje je prepuščeno SQL Serverju.

XQuery

```
SELECT pc.EnglishProductCategoryName Category,
       ps.EnglishProductSubcategoryName Subcategory,
       p.EnglishProductName Product,
       SUM(s.SalesAmount) Sales
 FROM dbo.FactInternetSales s
      INNER JOIN dbo.DimDate d
        ON d.DateKey = s.OrderDateKey
      INNER JOIN dbo.DimProduct p
        ON p.ProductKey = s.ProductKey
      INNER JOIN dbo.DimProductSubCategory ps
        ON ps.ProductSubcategoryKey = p.ProductSubcategoryKey
      INNER JOIN dbo.DimProductCategory pc
        ON pc.ProductCategoryKey = ps.ProductCategoryKey
 WHERE d.CalendarYear = 2008
       AND d.CalendarQuarter = 1
GROUP BY pc.ProductCategoryKey,
         pc.EnglishProductCategoryName,
         ps.ProductSubcategoryKey,
         ps.EnglishProductSubcategoryName,
         p.ProductKey,
         p.EnglishProductName;
```

SQL Server od verzije 2005 naprej podpira XML podatkovni tip. XML stolpec, torej ena XML instanca, lahko zasede do 2GB prostora. Bilo bi lahko precej neučinkovito, če bi brali celotnih 2GB, potrebovali bi pa samo nekaj malega znakov, npr. vrednost enega elementa. Zato SQL Server podpira poizvedovanje znotraj XML podatkov s pomočjo jezika XQuery.

XQuery je standardni jezik za delo z XML instancami. V SQL Serverju ga uporabljamo znotraj T-SQL ukazov, kot parametre metod XML podatkovnega tipa. V naslednjem primeru brskamo po SQL Server variabli, ki je XML podatkovnega tipa, z XQuery jezikom. V variabli deklariramo seznam kupcev z naročili, nato pa s pomočjo XQuery izraza vrnemo filtriran, sortiran in preoblikovan seznam naročil.

```
DECLARE @x AS XML;
SET @x = N'
<CustomersOrders>
  <Customer custid=«1«>
    <!-- Comment 111 -->
    <companyname>Customer NRZBB</companyname>
    <Order orderid=«10692«>
      <orderdate>2007-10-03T00:00:00</
orderdate>
    </Order>
    <Order orderid=«10702«>
      <orderdate>2007-10-13T00:00:00</
orderdate>
    </Order>
    <Order orderid=«10952«>
      <orderdate>2008-03-16T00:00:00</
orderdate>
    </Order>
  </Customer>
  <Customer custid=«2«>
    <!-- Comment 222 -->
    <companyname>Customer MLTDN</companyname>
    <Order orderid=«10308«>
      <orderdate>2006-09-18T00:00:00</
orderdate>
    </Order>
    <Order orderid=«10952«>
      <orderdate>2008-03-04T00:00:00</
orderdate>
    </Order>
  </Customer>
</CustomersOrders>';
SELECT @x.query('for $i in CustomersOrders/
Customer/Order
                 let $j := $i/orderdate
                 where $i/@orderid < 10900
                 order by ($j)[1]
                 return
                 <Order-orderid-element>
                  <orderid>{data($i/@
orderid)}</orderid>
                   {$j}
                 </Order-orderid-element>')
      AS [Filtered, sorted and reformatted
orders];
```

Kot lahko vidimo, je XQuery ukaz parameter .query metode XML podatkovnega tipa. Metodo kličemo v T-SQL SELECT stavku. XQuery je mešanica deklarativnega in proceduralnega jezika. V primeru imamo XQuery FLWOR (for – let – where – order by – return) zanko, ki ustreza for each zankam drugih proceduralnih jezikov. Obenem pa se deklarativno sprehodimo do nivoja naročil (CustomersOrders/Customer/Order), in pri tem ne navajamo, kako naj SQL Server pride do tja.

.NET

Deklarativni jeziki uporabljajo stavke, ki jih lahko gledamo kot velike gradnike aplikacije. Vendar moramo dostikrat narediti kakšna fina dela. Tu postanejo veliki gradniki nerodni, ali pa zaradi prevelikega števila parametrov kompliciran. T-SQL jezik je tudi omejen samo na delo s podatki v bazah, in ne moremo, na primer, pošiljati rezultatov T-SQL stavkov neposredno na tiskalnik. Zato od verzije 2005 naprej SQL Server omogoča pisanje določenih elementov v .NET jezikih, kot sta npr. najbolj popularna Visual C# in Visual Basic.

V .NET jezikih lahko napišemo funcije, procedure, trigerje, podatkovne tipe, in agregatne funkcije. Na ta način razširjamo zmogljivosti SQL Serverja. Podobno kot XQuery izraze, tudi .NET elemente izvajamo s pomočjo oziroma znotraj T-SQL stavkov.

V naslednjem primeru je funkcija, napisana v Visual C#, ki bo omogočala preverjanje pravilnosti niza znakov glede na podan vzorec. Vzorec podamo v obliki regular expression.

```
using System;
using System.Data;
using System.Data.SqlClient;
```

```
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Text.RegularExpressions;

public partial class CLRUtilities
{
    // Validate input string against regular
expression
    [SqlFunction(IsDeterministic = true,
DataAccess = DataAccessKind.None)]
    public static SqlBoolean fn_
RegexMatch(SqlString inpStr,
      SqlString regExStr)
    {
        if (inpStr.IsNull | regExStr.IsNull)
            return SqlBoolean.Null;
        else
            return (SqlBoolean)Regex.
IsMatch(inpStr.Value, regExStr.Value);
    }

};
```

Da lahko .NET funkcijo uporabljamo znotraj SQL Serverja, moramo importirati assembly v bazo, ter nato deklarirati funkcijo. Poleg tega moramo eksplicitno dovoliti izvajanje .NET kode znotraj SQL Serverja; privzeto je to prepovedano. Naslednja T-SQL koda dovoli uporabo .NETa, uvozi assembly, deklarira funkcijo, ter jo uporabi za preverjanje dveh e-mail naslovov. Pri prvem vrne vrednost True (1), ker je naslov veljaven, pri drugem pa False (2).

```
-- Create assembly
CREATE ASSEMBLY CLRUtilities
FROM 'C:\MDSBook\ProgrammingLanguages\
CLRUtilities.dll';
GO
-- Create function
CREATE FUNCTION dbo.fn_RegexMatch
  (@inpstr AS NVARCHAR(4000), @regexstr AS
NVARCHAR(4000))
RETURNS BIT
EXTERNAL NAME CLRUtilities.CLRUtilities.
fn_RegexMatch;
GO
-- Enable CLR
EXEC sp_configure 'clr enabled', 1
RECONFIGURE
GO
-- Test function
SELECT dbo.fn_RegexMatch(
        N'dsarka@solidq.com',
        N'^([\w-]+\.)*?[\w-]+@[\w-]+\.
([\w-]+\.)*?[\w]+$') AS ValidMail,
      dbo.fn_RegexMatch(
        N'dsarka$solidq.com',
        N'^([\w-]+\.)*?[\w-]+@[\w-]+\.
([\w-]+\.)*?[\w]+$') AS InvalidMail;
```

Mimogrede smo poleg Visual C# vpeljali še en »mini« jezik: Regular Expressions. To je jezik za definiranje vzorcev nizov, ki ga T-SQL sam po sebi ne podpira. No, s pomočjo C# funkcije smo razširili zmogljivosti T-SQL. Funkcijo lahko uporabljamo kot bolj napredno različico standardnega LIKE operatorja.

PowerShell

SQL Server vsebuje svojo command-line aplikacijo SQLCMD.EXE, ki omogoča pisanje T_SQL ukazov iz ukazne vrstice operacijskega sistema. Windows operacijski sistem ima svoje okolje za pisanje administrativnih skript, imenovano PowerShell. PowerShell omogoča precej več kot SQLCMD. PowerShell je razširljiv, in SQL Server pri instalaciji doda svoje elemente. PowerShell omogoča uporabo T-SQl stavkov, pa tudi .NET razredov, do katerih imamo znotraj PowerShella vmesnike, imenovane cmdlets.

Naslednji primer prikaže, kako v SQL PowerShell-u napišemo T-SQL ukaz. Ta ukaz nam bo samo prikazal trenutni datum in čas. SQL Server PowerShell okolje zaženemo iz Windows ukazne vrstice z ukazom SQLPS, nato pa vpišemo Invoke-Sqlcmd ali katerikoli drugi SQL PowerShell ukaz.

```
PS SQLSERVER:\> Invoke-Sqlcmd -Query
»SELECT GETDATE() AS TimeOfQuery;«
-ServerInstance »SOLIDQDS«
```

Ta primer je seveda zelo enostaven. Vendar lahko s pomočjo SQL PowerShella zelo avtomatiziramo administracijo, izvajanje skriptov, in še marsikaj drugega.

Skripting jeziki

Enostavni skripting jeziki, kot sta VBScript in Java Script, so dandanšnji že nekoliko zastareli. Napredni administratorji sistema SQL Server so jih uporabljali ali jih še uporabljajo za avtomza-

cijo administracije ter tudi za druge naloge. V sedanjem času je bolj priporočljivo uporabljati PowerShell. Zaradi kompatibilnosti za nazaj SQL Server še vedno podpira skripting jezike v korakih avtomatiziranih opravil, ki ji izvaja SQL Server Agent.

V naslednjem primeru kreiramo opravilo (scheduled job) z enim korakom (step). Step je tipa ActiveX Scripting step. Vse kreiramo s pomočjo T-SQL procedur, ki se nahajajo v Msdb sistemski bazi. VBScript koda je podana kot @command parameter procedure sp_add_jobstep.

```
USE msdb;
GO
-- Drop job if exists
IF  EXISTS (SELECT job_id
              FROM msdb.dbo.sysjobs_view
             WHERE name = N'VBScriptJob')
EXEC msdb.dbo.sp_delete_job @job_name =
N'VBScriptJob';
GO
-- Create job
DECLARE @jobId BINARY(16)
EXEC msdb.dbo.sp_add_job @job_
name=N'VBScriptJob',
        @enabled=1,
        @notify_level_eventlog=0,
        @notify_level_email=0,
        @notify_level_netsend=0,
        @notify_level_page=0,
        @delete_level=0,
        @description=N'No description
available.',
        @category_name=N'[Uncategorized
(Local)]',
        @owner_login_name=N'ADVENTUREWORKS\
Administrator',
           @job_id = @jobId OUTPUT;
-- Create VB Script step
EXEC msdb.dbo.sp_add_jobstep @job_id=@jobId,
@step_name=N'Rename a file',
        @step_id=1,
        @cmdexec_success_code=0,
        @on_success_action=1,
        @on_success_step_id=0,
        @on_fail_action=2,
        @on_fail_step_id=0,
        @retry_attempts=0,
        @retry_interval=0,
        @os_run_priority=0, @
subsystem=N'ActiveScripting',
        @command=N'Dim objFSO, strFolder,
objFolder, objFile, colFiles
                  Set objFSO =
CreateObject(»Scripting.FileSystemObject«)
                  strFolder = »C:\MDSBook\
ProgrammingLanguages«
                  Set objFolder = objFSO.
GetFolder(strFolder)
                  Set colFiles = objFolder.
Files
                  For Each objFile in
colFiles
                    If objFile.Name =
»FileToRename1.txt« Then
                       objFile.Name =
Replace(objFile.Name,«1«,«2«)
                    ElseIf objFile.Name =
»FileToRename2.txt« Then
                       objFile.Name =
Replace(objFile.Name,«2«,«1«)
                    End If
                  Next
                  Set objFSO = Nothing
                  Set objFolder = Nothing
                  Set colFiles = Nothing',
        @database_name=N'VBScript',
        @flags=0;
EXEC msdb.dbo.sp_update_job @job_id = @
jobId, @start_step_id = 1;
EXEC msdb.dbo.sp_add_jobserver @job_id = @
jobId, @server_name = N'(local)';
GO
Opravilu nismo dodali nobenega urnika,
zato ga moramo zagnati ročno. Primer ne
dela kaj posebnega – če v folderju C:\
MDSBook\ProgrammingLanguages obstaja
datoteka FileToRename1.txt, jo preimenuje v
FileToRename2.txt, če obstaja FileToRename2.
txt, pa jo preimenuje nazaj v FileToRename1.
txt. Primer vseno prikazuje, da lahko SQL
Server Agent servis izkoristimo tudi za kaj
več kot samo za redno izvajanje backupov, za
redno delanje rezervnih kopij.
```

Windows Management Instrumentation (WMI)

WMI je zbirka razredov, ki prihaja z operacijskim sistemom Windows. Ti razredi nam omogočajo programsko upravljanje operacijskega sistema. Med drugim omogočajo tudi lovljenje dogodkov, kot je na primer kreacije datoteke na folderju. SQL Server doda nekaj svojih WMI razredov. SQL Server Agent pa omogoča kreacijo opozoril (alertov), ki reagirajo na WMI dogodke.

Dogodke zajemamo z poizvedbami, ki jih pišemo z jezikom Windows Management Instrumentation Query Language (WQL). WQL je zelo enostaven jezik, spominja na SQL jezike; vendar,

naš pregled ne bi bil kompleten, če ne pokažemo WQL poizvedbe. V ta namen kreirajmo alert.

```
USE msdb;
GO
-- Drop alert if exists
IF  EXISTS (SELECT name
              FROM msdb.dbo.sysalerts
             WHERE name = N'WMIAlert')
EXEC msdb.dbo.sp_delete_alert @
name=N'WMIAlert';
GO
-- Create alert
EXEC msdb.dbo.sp_add_alert @name=N'WMIAlert',
        @message_id=0,
        @severity=0,
        @enabled=1,
        @delay_between_responses=0,
        @include_event_description_in=0,
        @category_name=N'[Uncategorized]',
        @wmi_namespace=
       N'\\.\root\Microsoft\SqlServer\
ServerEvents\MSSQLSERVER',
        @wmi_query=N'SELECT * FROM CREATE_
DATABASE',
        @job_name=N'VBScriptJob';
GO
```

WQL poizvedba je podana kot @wmi_query parameter sistemske procedure sp_add_alert. Poudarimo naj še, da je imenski prostor za SQL Server WMI objekte točno določen, kot je vpisan v parameter @wmi_namespace. Kot odgovor na dogodek kreacije SQL Server baze bo alert zagnal job VBScriptJob, ki smo ga naredili v prejšnjem delu, ko smo opisovali uporabo skripting jezikov v SQL Serverju. Alert testiramo tako, da kreiramo bazo (po testu seveda tudi pospravimo za sabo):

```
USE master;
GO
CREATE DATABASE TestWMI;
GO
DROP DATABASE TestWMI;
GO
```

Ker so alerti asinhroni, moramo počakati nekaj sekund, potem pa mora biti testna datoteka preimenovana.

Analysis Services

SQL Server Analysis Services (SSAS) je namenski podatkovni strežnik, namenjen za analize multidimenzionalnih struktur ter za data mining analize. Celoten strežnik je prilagojen hitrim in zahtevnim analizam. Multidimenzionalne strukture so namenjene sprotnim pregledom velike količine podatkov (On-Line Analytical Processing, OLAP). Strukturo si predstavljamo kot večdimenzionalno hiperkocko (seveda si kocko lahko predstavljamo samo v treh dimenzijah, vendar SSAS ni omejen s tremi dimenzijami), zato tudi govorimo o OLAP kockah. Data Mining del pa obsega matematično, predvsem statistično zahtevne analize, ki nam dajo vzorce v podatkih. Vzorce uporabljamo za napovedovanje neke ciljne variable v novih podatkih. Data Mining je najbolj napreden del poslovne inteligence, daje lahko najboljše rezultate, vendar zahteva tudi največ znanja.

MDX

Za preiskovanje OLAP kock uporabljamo jezik MultiDimensional eXpressions (MDX). Na hitro je jezik podoben T-SQL, saj je ravno tako deklarativen. Vendar moramo vedeti, da MDX SELECT lahko vrne do 128 dimenzij, torej lahko vrne hiperkocko, in ne samo tabele, kot T-SQL. Tabelo navsezadnje lahko gledamo kot dvodimenzionalno kocko z dimenzijama stolpci in vrstice.

MDX jezik je precej kompleksen. K sreči večino poizvedb naredimo z OLAP odjemalskimi orodji, kot je na primer Microsoft Office Excel 2010. Pivot tabelo ali graf oblikujemo preko uporabniškega vmesnika, Excel pa za nas kreira MDX poizvedbe. No, da bi polno izkoristili Analysis Services, je vendarle dobro vedeti, kako pišemo MDX poizvedbe in izraze. Naslednji primer MDX poizvedbe dela na AdventureWorks2008R2 testnem SSAS projektu, ki ga dobimo skupaj s testnimi bazami.

```
SELECT [Customer].[Customer Geography].
[Country] ON COLUMNS,
       NON EMPTY [Date].[Calendar].[Calendar
```

```
Year] ON ROWS
  FROM [Adventure Works]
 WHERE ([Measures].[Internet Sales Amount],
        [Product].[Product Categories].
[Category].[Bikes])
```

V tem primeru želimo pivotiranje vrednosti prodaje po državah v stolpcih in letih v vrsticah (pri tem izločimo leta brez prodaje), in sicer samo za produkte kategorije kolesa.

DMX

Za poizvedbe nad SSAS Data Mining modeli uporabljamo Data Mining eXtensions (DMX) jezik. S tem jezikom lahko pregledujemo vsebino modela, to je najdene vzorce, ter tudi napovedujemo ciljno variablo na novih podatkih.

DMX jezik je še bolj podoben T-SQL kot MDX. Pravzaprav gre za poenostavljen T-SQL z nekaj specifičnimi Data Mining izrazi. DMX SELECT tako kot T-SQL SELECT vrača dvodimenzionalne tabele. Vendar DMX lahko vrne gnezdene tabele v stolpcih zunanje tabele. Naslednja DMX poizvedba preiskuje vsebino Decision Trees modela, ki pride z AdventureWorksDW2008R2 testnim projektom.

```
SELECT NODE_CAPTION,
       NODE_DESCRIPTION,
       NODE_DISTRIBUTION
  FROM [TM Decision Tree].Content
 WHERE NODE_TYPE = 3;
```

Če se želimo znebiti gnezdenih tabel, lahko rezultat »sploščimo« s ključno besedo FLATTENED.

```
SELECT FLATTENED
       NODE_CAPTION,
       NODE_DESCRIPTION,
       NODE_DISTRIBUTION
  FROM [TM Decision Tree].Content
 WHERE NODE_TYPE = 3;
```

T-SQL

Morda je nekoliko manj znano, a Analysis Services podpira tudi nekaj Transact-SQL poizvedb. SSAS ima namreč vpoglede (view), takoimenovane Schema Rowsets, ki nam v tabelarični obliki prikazujejo sistemske podatke. SSAS Schema Rowsets lahko primerjamo s SQL Serverjevimi Dynamic Management Views. V naslednjem primeru uporabljamo DISCOVER_OBJECT_ACTIVITY schema rowset, da dobimo 20 najbolj uporabljanih objektov v SSAS bazi.

```
SELECT TOP 20
       OBJECT_PARENT_PATH,
       OBJECT_ID,
       OBJECT_CPU_TIME_MS AS CPU,
       OBJECT_READS AS Reads,
       OBJECT_ROWS_SCANNED AS [Rows]
  FROM $SYSTEM.DISCOVER_OBJECT_ACTIVITY
 ORDER BY OBJECT_CPU_TIME_MS DESC;
```

.NET

Tudi tale del za SSAS ni tako razglašen kot za SQL Server, pa je vendar vredno omembe: tudi Analysis Services podpira .NET procedure. Tule je primer dveh funkcij, ki bosta vrnili ime analitskega strežnika, na katerega smo se priklopili, in ime baze, v katere kontekstu se nahajamo.

```
using System.Text;
using System.Data;
using System.Data.OleDb;
using ADOMD = Microsoft.AnalysisServices.
AdomdServer;

namespace ServerProcs
{
    public class ASProcs
    {
        public static string GetSvrId()
        {
            return ADOMD.Context.
CurrentServerID;
        }

        public static string GetDbName()
        {
            return ADOMD.Context.
CurrentDatabaseName;
        }
    }
}
```

Poleg .NET procedur sta MDX in DMX jezika razširjena še z nekaj Visual Basic for Applications (VBA) in Excel funkcijami. Vendar je tule

pomembna razlika: VBA funkcije so že vgrajene v SSAS, medtem ko moramo za uporabo Excel funkcij Excel instalirati na isti računalnik kot Analysis Services. SSAS ima vgrajene samo vmesnike za klic Excel funkcij. Naslednji primer kaže uporabo .NET in VBA funkcij v DMX poizvedbi.

```
SELECT SSASProcs.GetSvrId() AS ServerName,
       SSASProcs.GetDbName() AS DatabaseName,
       VBA.Now() AS CurrentDateTime
  FROM [Customer Clusters];
```

XMLA

XML for Analysis (XMLA) je edini protokol, ki ga SSAS uporablja za komunikacijo z aplikacijami. XMLA je odprt standard za dostop do analitskih storitev preko spleta (da, SSAS je že sam po sebi Web Service). Preko XMLA klicev lahko naredimo resnično lahko odjemalsko aplikacijo, npr. kar znotraj Internet Explorerja.

MDX jezik je namenjen samo za poizvedbe in kreiranje izrazov za razne izračune. MDX nima ukazov za kreacijo OLAP kock ali za procesiranje (polnjenje in računanje agregatov) kock. Analysis Services uporablja XMLA za data definition, data manipulation, in data control ukaze. Zato XMLA za SSAS razširja osnovno XMLA 1.1 specifikacijo.

XMLA ukaze lahko pišemo neposredno v XMLA Query oknu v SQL Server Management Studiu (SSMS). S pomočjo SSMS lahko tudi kreiramo XMLA skripte na osnovi obstoječih objektov, ter tako naredimo rezervno kopijo definicij objektov. Kot primer tule prikazujemo XMLA skripto za procesiranje Adventure Works kocke iz testnega SSAS projekta.

DAX

PowerPivot je nov analitski produkt, ki je izšel skupaj s SQL Serverjem 2008 R2. Na voljo je kot brezplačen dodatek (add-in) za Excel 2010 ter za SharePoint 2010. PowerPivot je, da podamo kratko definicijo, »in-memory column-oriented relational database management system«. Torej, gre za sistem za upravljanje baz, ki temelji na relacijskem modelu, kar pomeni, da so podatki organizirani v tabele. Interno tabele ne shranjuje po vrsticah, kot jih mi vidimo, ampak vsak stolpec posebej, sortirano. Trenutno

```
<Batch xmlns=«http://schemas.microsoft.com/analysisservices/2003/engine«>
  <Parallel>
    <Process
      xmlns:xsd=
        »http://www.w3.org/2001/XMLSchema«
      xmlns:xsi=
        »http://www.w3.org/2001/XMLSchema-instance«
      xmlns:ddl2=
        »http://schemas.microsoft.com/analysisservices/2003/engine/2«
      xmlns:ddl2_2=
        »http://schemas.microsoft.com/analysisservices/2003/engine/2/2«
      xmlns:ddl100_100=
        »http://schemas.microsoft.com/analysisservices/2008/engine/100/100«
      xmlns:ddl200=
        »http://schemas.microsoft.com/analysisservices/2010/engine/200«
      xmlns:ddl200_200=
        »http://schemas.microsoft.com/analysisservices/2010/engine/200/200«>
      <Object>
        <DatabaseID>Adventure Works DW 2008R2</DatabaseID>
        <CubeID>Adventure Works</CubeID>
      </Object>
      <Type>ProcessFull</Type>
      <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
    </Process>
  </Parallel>
</Batch>
```

PowerPivot deluje izključno v pomnilniku – preden lahko karkoli delamo, mora PowerPivot vse podatke naložiti v pomnilnik.

Poleg podatkov, ki jih naložimo iz raznih virov, lahko dodamo izračunane vrednosti. Za te izračune potrebujemo programski jezik. PowerPivot uporablja Data Analysis Expressions (DAX). Ideja, zakaj imeti še en dodatni jezik, izhaja iz želje, da je delo s PowerPivot čim bolj podobno delu z Excelom in na ta način enostavnejše za uporabnike Excela.

Za hiter primer smo uporabili dbo.vTargetMail view iz AdventureWorksDW2008R2 testne baze. Podatke smo iz SQL Serverja uvozili v PowerPivot v Excelu 2010, nato pa izračunali število kupcev s pomočjo DAX COUNTA funkcije:

```
=COUNTA('vTargetMail'[CustomerKey])
```

Kot vidimo, se v DAX sklicujemo na neko vrednost s sintakso 'tabela'[stolpec].

Nadaljevanje članka v naslednji Piki...

Dejan Sarka

Dejan Sarka se osredotoča na razvoj baz podatkov in poslovnih aplikacij. Poleg projektov posveti približno polovico svojega časa za usposabljanje in mentorstvo. Je ustanovitelj slovenskih SQL Server in .NET Users Group. Dejan Sarka je glavni avtor ali soavtor devetih knjig o podatkovnih bazah in SQL strežnikih. Razvili je tudi dva tečaja in številne seminarje za Solid Quality Mentors.

Dejan Sarka		
4.-8. april	Microsoft BI Bootcamp 2010	5 dni
4.-6. maj	#2778 Writing Queries Using Microsoft SQL Server 2008 Transact-SQL	3 dni
6.-10. junij	#6232 Implementing a Microsoft SQL Server 2008 Database	5 dni
20.-24. junij	#6231: Maintaining a Microsoft SQL Server 2008 Database	5 dni
4.-8. julij	Advanced Transact-SQL	5 dni
11. julij **NOVO!**	Master Data Management with SQL Server 2008 R2 *Za udeležence tečaja Advanced Transact-SQL je cena tega tečaja samo 200,00 € + DDV!*	1 dan

Using Multicolumn Statistics to get better Query Plans

Columns statistics are an integral part of query optimization in SQL Server. SQL Server needs statistics to estimate the amount of data that needs to be processed at the different stages of the query execution. If SQL Server is missing good statistics there is a good chance that it will not find the most efficient plan. In this article we will see how multicolumn statistics can be used to get better query plans. It assumes that you have a basic understanding on statistics in SQL Server. You can find good information on how SQL Server creates and uses statistics in the whitepaper: Statistics Used by the Query Optimizer in Microsoft SQL Server 2008 (*http://msdn.microsoft.com/en-us/library/dd535534.aspx*)

Statistics can be created in 3 ways:

- Automatically for columns used a searchable arguments in queries when "Auto Create Statistics" is enabled in the database
- Statistics are created for every index created in the database
- Manually created statistics

Statistics can be created for single columns on a table to estimate the selectivity when applying a filter on that column. But if there are filters on multiple columns on a table, single column statistics are only of limited use. Consider you have a filter on COL1 and COL2 on a table with 200000 rows and the estimate for COL1 is 20000 and the estimate on COL2 is 10000. But as we don't know if there is any correlation between COL1 and COL2 the result can be anything between 0 and 10000 rows even when the single column statistics are 100% correct.

In that case multicolumn statistics can help to get more accurate statistical data as they are built over the value distribution over both columns. Multicolumn statistics are created automatically for composite indexes, or can be created manually, but they are never created automatically without a composite index. So in the case above we would not have a good statistic over both columns, if we don't have a composite index or create a statistic manually and SQL Server has to estimate something between 0 and 10000 based on the single column statistics it can create.

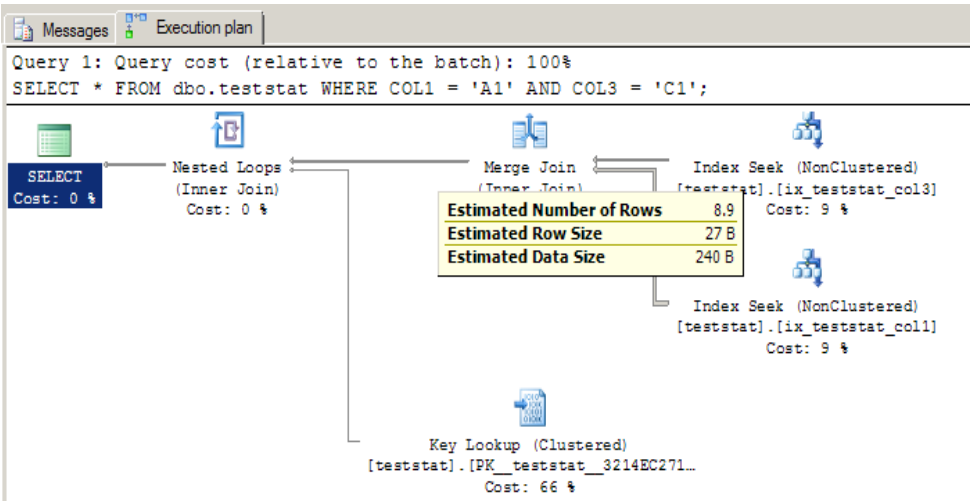
Let's have a look what SQL server does in that cases. You can download the script to create a sample table here: (please add a link) The table has the following layout where ID is the Primary Key and COL4 only serves as filler column:

```
CREATE TABLE dbo.teststat
    (ID int IDENTITY(1,1) PRIMARY KEY,
     COL1 varchar(10),
     COL2 varchar(10),
     COL3 varchar(10),
     COL4 char(1000));
```

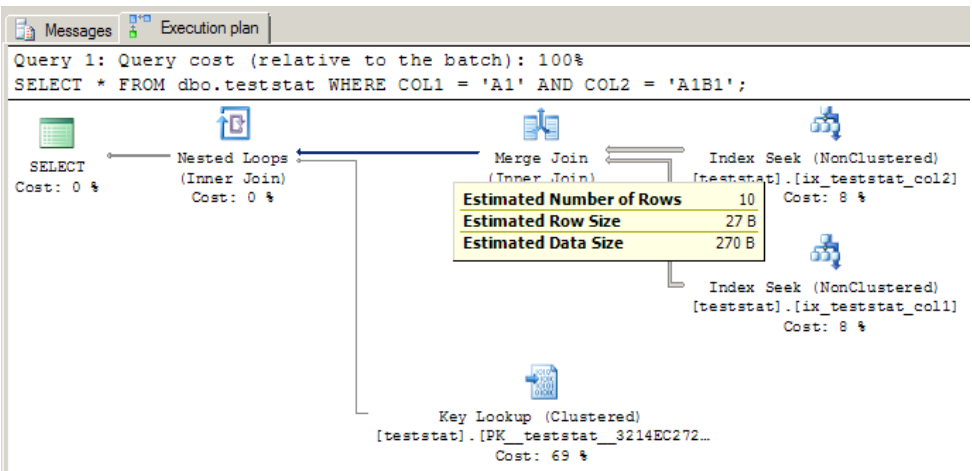
In addition the script creates indexes on COL1, COL2 and COL3 and fills the table with 2000 sample rows with the following distribution:

	COL1	COL2	COL3
Distinct Values	10	20	20
Rows per Value	200	100	~100 (random)

Let's first concentrate on COL1 and COL3. If we write a query and filter on COL1 SQL Server estimates correctly 200 rows and uses a clustered index scan as the index on COL1 is not selective enough. The same happens when we filter on COL3 only. Please note that the returned numbers can be varying as the values in COL3 were generated randomly. If we now filter on COL1 and COL3 SQL Server has to calculate the estimated row count based on the single column statistics on the 2 columns. Below you can see the query plan generated. SQL Server estimates to get 8,9 rows and uses the indexes on COL1 and COL3 to return the data because of the high selectivity. The actual rows returned are 6 in my case.



Now let's have a look what happens when we query COL1 and COL2. If you query COL1 and COL2 alone SQL makes the right estimation, but if we filter by COL1 = 'A1' AND COL2 = 'A1B1' SQL Server estimates to get only 10 rows back and therefore again decides to use indexes to return the data as you can see below. In reality in that case 100 rows are returned which would be a selectivity that would lead to a clustered index scan, if SQL Server would estimate correct.

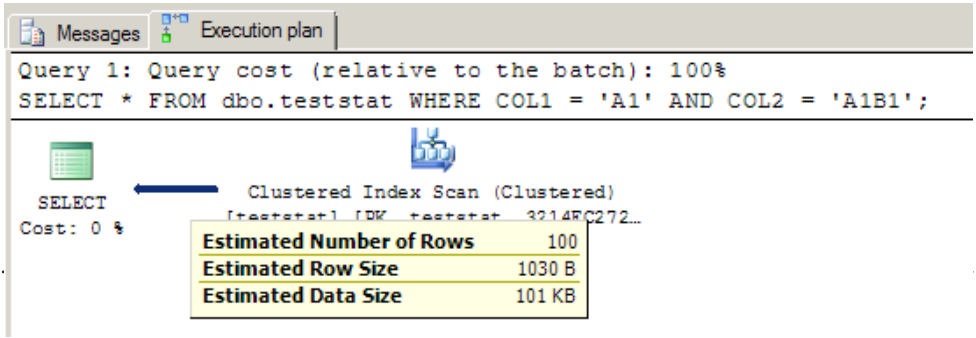


So what happened here? In the first case there is no correlation between the values in the 2 columns. In the second case there is a full correlation between the values in the columns. Actually every value within COL2 only exists in combination with a distinct value within COL1. And that is the problem. As SQL Server doesn't know about any correlation between the columns it has to guess something. In that case SQL Server always assumes that there is no correlation between the values in the different columns. This is perfect for the first case, but leads to a wrong plan in the second case.

To overcome this problem we can create a multicolumn statistic manually to help SQL Server to make the correct estimation.

```
CREATE STATISTICS teststat_col1_col2 ON dbo.
teststat(COL1,COL2);
```

After doing this we can see that SQL Server performs a clustered index scan as he estimates correctly 100 rows which are not selective enough to seek any of the available indexes as you can see below.



What we have seen is that the combination of different single columns statistics can be used to make good estimations for filters on multiple columns on a table, if there is no or only a small correlation between them. But if a correlation exists the estimations can lead to wrong plans. In that case multicolumn statistics can be considered if a composite index is not suitable.

Please note that the example used in that article is oversimplified to be able to concentrate on the statistics only. Especially the indexes used in that example would not be used in reality as none of them is selective enough to cover a query on even a single value. Therefore in reality it would not be a good idea to create them at all. In real live that cases happens mostly in cases where you have selective indexes on the columns. But for some queries the rows are filtered in ranges or based on multiple values using an IN operator, which produces a none selective query which will not be recognized as SQL Server doesn't realize that column values in the different filter columns are correlated and therefore underestimates the rows returned. And these are the cases where you should consider creating multicolumn statistics.

Herbert Albert

Herbert Albert je MCSE, MCTS, MCITP, MCDBA in MCT za SQL Server. Je SQL Server predavatelj, svetovalec in direktor Srednjeevropske enote SolidQ. Njegovo znanje pokriva širok spekter Microsoft tehnologij in mrežne arhitekture, predvsem pa se osredotoča na SQL Server. Je soavtor "SQL Server 2008 (R2)Upgrade Technical Reference Guide" in "SQL Server 2005 Step-by-Step Applied Techniques", Microsoft Press 2006.

Herbert Albert (tečaji so v angleščini)		
30.3.- 1.4.2011 **NOVO!** **Prvič v Sloveniji!**	Troubleshooting and Performance Tuning for SQL Server 2008	3 dni

Strategy and Implementation—A Powerful Duo to Optimize Ajax Applications

In general, performance is not an exact science; let alone it is in the realm of Web Ajax applications. If I have to indicate a golden rule of Ajax optimization, then I would say that saving your application some work is always a good way to improve its performance. This leads me to the fundamental question behind optimization: Is there anything you can take out of the code?

In computer science, there's an absolute way of measuring the goodness of an algorithm—computational complexity. Through computational complexity you just measure the number of fundamental operations a given algorithm performs on an input data set. Sometimes, however, you have no other choice than going with a given algorithm. Put another way, if you can't just improve on the algorithm you can apply shortcuts to its implementation and get better performance by avoiding some work. The general answer to this question is caching, but caching can be applied at various levels. In this article, I'll discuss two general patterns that help getting results quicker in Ajax applications. The first pattern is Predictive Fetch; the second is Memoization. As you can see, both revolve around the general theme of caching.

The Predictive Fetch Pattern

The idea behind the pattern is just predicting the next user actions and preloading some of the data that it will be necessary to use later—if the user will really perform the predicted action. Downloaded during idle time or, simply asynchronously in the background, data is cached on the client using some JavaScript object. Next, when the user triggers the action for which the previously cached data is required, some client-side code retrieves that and uses to refresh the user interface as appropriate. In the end, the Predictive Fetch pattern is a sort of context-sensitive, client-side cache that

requires a well-defined strategy about what to download and when to download it.

From a functional standpoint, predictive fetch is not such a hard pattern to implement. The real point with predictive fetch is all another—devising an effective strategy. You can't pre-fetch just any data the user can possibly request and from any stage of the user interface. You must be careful to cover the most likely actions and/or the most critical regardless of their likelihood. As you can see, the strategy is the most important aspect of predictive fetch and it is not something that can be hard-coded in a recurring pattern solution. It is part of the architecture and belongs to the overall solution. The main drawback of predictive fetch is loading the wrong data that will never be requested by the user. This creates unnecessary overhead and also may end up taking up some memory in the client PC. Another point to take into due account is the overall behavior of the application that may appear to be sort of random, if not inconsistent, to the end user. Imagine two similar features in a page—one that supports predictive fetch and one that doesn't. Clearly, when the user selects the one that doesn't have pre-fetch she faces a much longer response time. When she selects the other function, the response time is immediate. This may be confusing and may contribute to create a negative feeling around your application.

Imagine at some point your user may decide to click and download details about a customer. Next, he may, or may not, wish to drill down into the orders placed by that customer. Here's the tricky point. With predictive fetch enabled, at the end of the callback that updates the UI with the details of the customer you trigger another remote call to download orders. If later the user decides to look at the orders, data is already there. If not, you placed a useless extra request.

The Memoization Pattern

Caching is nothing new in software and is an effective way to save some work. At its core, caching consists in the software's ability of retrieving data from a location that is quicker to access compared to the original source. Caching is not limited to data loading; caching can be applied to calculated results. This is just the essence of the pattern.

In JavaScript, functions can contain properties as well as a behavior. You can use this feature to give your JavaScript functions—at least, the most sophisticated and performance critical of them—a private cache to store values of previous calculations. In this way, next time a previously calculated value is requested it is served immediately without having to execute the potentially long operation again. Here's an example:

```
var getDetails = function(id) {
   var thisFunc = arguments.callee,
       result;
   if (!thisFunc.cache[id]) {
      result = _longOperation(id);
      thisFunc.cache[id] = result;
   }
   return thisFunc.cache[id];
}
getDetails.cache = {};
```

The function getDetails has its own private cache where it stores computed results. Results are stored by parameter. If you need to store by multiple parameters you can save the list of parameters—as returned by arguments—to an array and then to a JSON string. When the following sequence of calls occur, the return value is calculated only the first time for each combination of input parameters.

```
var a1 = getDetails(2);
var a2 = getDetails(2);
var a3 = getDetails(1);
```

Saving work, especially if it is a hard and long work, definitely improves performance. In this

regard, a special mention is also worth for special cases. Whenever you realize that for a particular combination of parameters you can get return values more quickly than usual, by all means take the shortcut. Adding a bunch of IF statements around the body of a function may not seem elegant, but if it helps getting a better performance I usually do that.

Summary

Performance is often a relative concept in the Web. Sometimes there's room for redesigning the code, changing the algorithm and achieve better and faster results. In some other times, you have to resort to giving users the perception of a better performance by optimizing the loading time and providing visual feedback during long operations. The two patterns examined touch on two different views—optimization by strategy and optimization by implementation.

Dino Esposito

Dino Esposito je avtor "Programiranje ASP.NET MVC" za Microsoft Press, kot tudi "Programming ASP.NET 4" in drugih knjižnih uspešnic, kot so "Microsoft ® NET: Architecting Applications for the Enterprise". Redno sodeluje z MSDN Magazine in DevProConnections Magazine in pogosto predava na tehničnih dogodkih po vsem svetu, vključno z Microsoft TechEd, DevConnections in vrhunskih evropskih dogodkih, kot so DevWeek in BASTA.

Dino Esposito (tečaji so v angleščini)		
27.-29. junij	.NET Software Architecture - Patterns of Application Architecture	3 dni
30.-1. julij	Workshop: Patterns and Practices of Architecting .NET Applications	2 dni
Posebna ugodnost: za kandidate, ki se prijavijo na oba dela tečaja, je kotizacija samo 1.500,00 € + DDV!		
3.-5. oktober	Programming ASP.NET MVC	3 dni
6.-7. oktober	Mastering Javascript and jQuery	2 dni
Posebna ugodnost: za kandidate, ki se prijavijo na oba dela tečaja, je kotizacija samo 1.500,00 € + DDV!		

Microsoft SQL Server: Evolution from SQL Server 7.0 to SQL Server 2008 R2

Excerpt

Microsoft SQL Server became a true enterprise-level database management system (DBMS) with SQL Server 7.0. But the evolution didn't stop there. SQL Server has come a long way in the past 10 years. This white paper provides an overview of SQL Server enhancements during this evolutionary journey, covering the SQL Server suite of services and tools for both the relational database engine and business intelligence (BI).

This white paper is organized historically. It covers the most important features from version to version, showing how the product has matured over time to satisfy ever growing business needs. The paper starts with SQL Server 7.0, moves to SQL Server 2000, 2005 and 2008, and then ends with a discussion of SQL Server 2008 R2, which is available from this year (2010). Although this paper focuses on the ongoing improvement process, it also mentions some unfortunate features—including some that have been or will be abandoned.

SQL Server 7.0: Sphinx

Let's start our historical look at SQL Server with Version 7.0, also known by the code name Sphinx. This version was revolutionary in many ways. Microsoft rewrote the storage engine from scratch, including a new locking mechanism and provided completely new tools. SQL Server 7.0 was the first version seriously targeting the enterprise market, providing both performance and scalability. And SQL Server 7.0 was the first version of the product that included OLAP functionality for free within the suite, which changed the OLAP market forever. Starting with SQL Server 7.0, OLAP became affordable for small and medium businesses, and not limited solely to large enterprises.

A New Database Engine

The code base for SQL Server originated in Sybase SQL Server. But with SQL Server 7.0, Microsoft moved away from Sybase code, rewriting the database engine from scratch. Microsoft developers did an excellent job; the SQL Server 7.0 storage engine remains the foundation for newer SQL Server versions' storage engine.

Microsoft developers increased the basic storage unit for data, the page, from 2K to 8K. The same page size is still used in SQL Server 2008. SQL Server 7.0 was also the first version that dynamically managed memory and files on disk. Although some IT professionals thought these management features would make the database administrator (DBA) role obsolete, they instead freed up DBAs to focus on more meaningful tasks, making the role more valuable than ever, especially for medium and large databases. However, with SQL Server 7.0's new self-management capabilities, small companies no longer needed extensive database management knowledge in-house to operate SQL Server.

SQL Server 7.0 was the first version of the product available on desktop computers through the free Microsoft Desktop Edition (MSDE). This edition—called SQL Server Express in later versions—was and still is useful together with merge replication to support disconnected users. Merge replication, also new in SQL Server 7.0, lets disconnected workers such as sales representatives travel around with their notebooks, insert orders, and update customer data, then merge their changes with changes on the central, production server when they reconnect. Merge replication worked in all editions of SQL Server 7.0, including MSDE. In addition to merge replication, SQL Server 7.0 also provided snapshot

and transactional replication to support the need for distributed, "scale out" data scenarios among multiple databases and locations.

Besides the rewritten storage engine, Microsoft developers also introduced an entirely new set of database engine tools for SQL Server 7.0. The main administrative tool was Enterprise Manager, a Microsoft Management Console (MMC) snap-in. For performance tuning and troubleshooting, SQL Server Profiler provided a rich user interface for setting up and analyzing server traces. Using server traces, you can capture the commands SQL Server is processing as well as SQL Server' s responses to those command. You can capture errors, warnings, execution times, disk I/O, memory and CPU consumption, and much more. SQL Server 7.0 also let you then use another tool—the Index Tuning Wizard (ITW)—to quickly get reasonable recommendations for indexes based on this real-time usage data from Profiler. The ITW worked well as long as the sample of captured queries was representative of your overall workload.

The main new SQL Server 7.0 tool for developers was SQL Server Query Analyzer. Although it had a small footprint, Query Analyzer was powerful and popular among database developers; many of us still miss this tool, which was replaced in SQL Server 2005 by functionality in the new SQL Server Management Studio (SSMS). Query Analyzer was handy for ad hoc queries and for troubleshooting; it was the first tool capable of graphically showing a query's execution plan, making the plan intuitive to understand. Figure 1 shows an example of an execution plan in SQL Server 7.0 Query Analyzer.

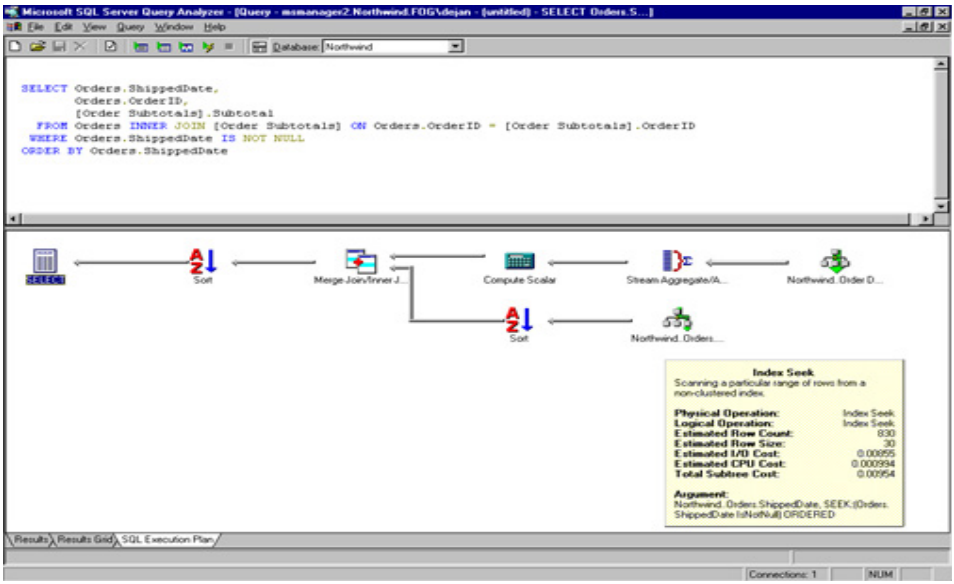


Figure 1: Graphical Execution Plan in Query Analyzer

Among the many database engine enhancements in SQL Server 7.0, developers especially appreciated the new locking schema. With the new version, the lowest level of locking granularity became the row level instead of the page level. Locking escalation was dynamic, so you didn't need to manually set the lock escalation threshold. In addition, accessing remote data sources from SQL Server was easier than ever before thanks to linked servers. A linked server, in short, is a named connection string that uses

an OLE DB provider. In SQL Server 7.0, you could refer to remote data from a Transact-SQL (T-SQL) query simply by providing the four-part object name: server, database, schema, and object. SQL Server 7.0 was also the first version that let you use full-text searches on database columns; this capability let you search for words or phrases against character data.

With Windows integrated security fully implemented in SQL Server 7.0, DBAs could stop worrying about brute-force attacks on SQL logins if they used only trusted connections and disabled the mixed authentication mode. However, with the mixed authentication mode, which means SQL logins, enabled SQL logins were still exposed to a brute-force attacks, because SQL Server did not implement any password or account policies, like account lockout policy, for SQL logins.

The dramatic improvements in SQL Server 7.0's storage engine and tools led to better and more controllable performance and provided appropriate scalability for most business needs in the late '90s. However, DBAs still had a hard time achieving high availability. For example, to correctly set up SQL Server in a cluster, you had to work through a 70-page "How to install SQL Server 7.0 Enterprise Edition on Microsoft Cluster Server" white paper, then check Microsoft's "Order of Installation for SQL Server 7.0 Clustering Setup" Knowledge Base article for updated information.

OLAP Revolution

Although the new SQL Server 7.0 database engine was significant, the IT community expected it. Customers demand that Microsoft continuously deliver database improvements, and the company must also try to stay ahead of or abreast of competitors, who are also always working to move their products forward. The real revolution in SQL Server 7.0 occurred in BI. Although many different Online Analytical Processing (OLAP) servers and client tools were on the market at the time, they all had a common failing: they were very expensive. The high prices were primarily related to the target buyers of OLAP solutions—decision makers, who also usually control the money. By including OLAP Services in the SQL Server 7.0 suite for free, Microsoft changed the OLAP market forever, bringing OLAP to the masses.

OLAP lets end users change the view of displayed data or the report they're looking at in real-time, without needing to ask for help or for a report update from IT support or developers. There are two main pillars that enable real-time analysis: simple, predictable schema, which enables building simple and intuitive client tools to navigate around, sort, and analyze the data; and OLAP server performance, which enables the needed speedy data access. OLAP uses the well-known star schema, which features two types of tables: fact tables and dimensions. Fact tables hold things you are measuring (i.e., measures), and dimensions consist of attributes you are using to break down your measures (i.e., the elements that let you drill down until you find interesting information). For example, a Sales fact table might hold such measures as sales quantity and sales amount, while the Customer dimension might have such attributes as country, region, city, and customer. You start analyzing from the top, checking the total sales, then drill down to individual countries, regions, cities, and customers. OLAP achieves lightning analysis speed through pre-computed aggregates. So, for example, from detailed data in your source system, the OLAP server calculates totals across cities, regions, and countries in advance instead of being processed for every user request at the time of the request.

SQL Server 7.0 provided several OLAP tools: OLAP Services, which was the OLAP server; Analysis Manager, a tool for developing OLAP databases and cubes for administering OLAP databases; and client tools. Microsoft extended Office Excel to serve as an OLAP Services client

tool, and third-party tools started appearing soon after. Probably the most popular client tool was ProClarity, a rich client from a company called Knosys; the company later changed its name to ProClarity, and in 2007, Microsoft acquired it. Figure 2 shows a typical star schema being built in OLAP Manager.

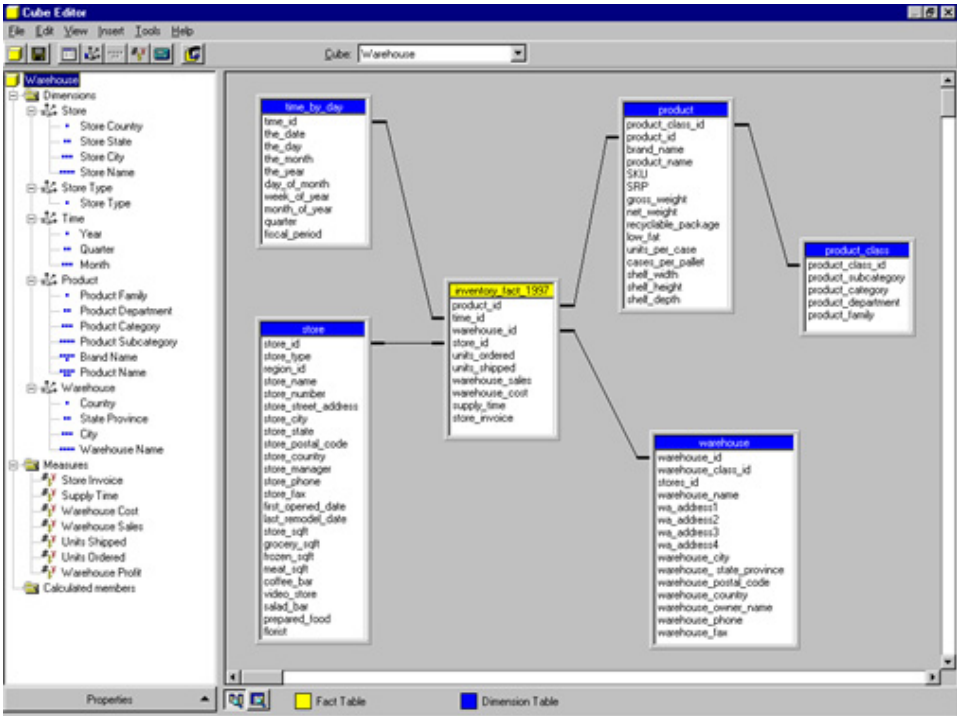


Figure 2: Star Schema in OLAP Manager

One star schema typically covers one business subject, such as sales, finance, or warehouse. You connect multiple star schemas through shared dimensions. And you almost always have a time dimension as well because nearly all organizations need to perform analysis over time. Before building any OLAP cubes, you need to consolidate data from multiple sources into multiple star schemas. This consolidation is typically performed in a relational database called a data warehouse. A data warehouse consists of one or more star schemas, with merged, cleansed, and historical data. From a data warehouse, you build your OLAP cubes. In SQL Server 7.0, you couldn't build an OLAP cube from the scratch; the schema was inherited from a relational data warehouse. In addition, a single cube was limited to a single fact table (i.e., a single star schema). Although it was possible to compare data from two fact tables-- for example, from actual sales and the sales plan--it wasn't simple.

To keep your data warehouse current, you must regularly extract data from sources, transform it into your data warehouse schema, and load it to the data warehouse. This regular process is called Extract-Transform-Load (ETL). You can write your own ETL application from scratch, but you can also use a tool that helps you build such an application. With SQL Server 7.0, Microsoft shipped a new tool called Data Transformation Services (DTS) that let you build ETL applications called packages. DTS packages were task-oriented. In one task, for example, you would

extract the data, do the transformations, and load the transformed data in a staging table. Then, you would use another task to re-read the same data for another purpose. Thus, DTS was actually more ELT than ETL tool.

With DTS, the only way to read data once, do multiple transformations in memory, and write the data, was to use the Data Pump task. Inside the Data Pump task, you could transform data by using an ActiveX scripting language such as VB Script. Nevertheless, with the new SQL Server database engine, OLAP Services, and DTS, SQL Server 7.0 provided a very powerful platform for satisfying business needs for data and information.

SQL Server 2000: Shiloh

Although not as revolutionary as SQL Server 7.0, SQL Server 2000, code-named Shiloh, nevertheless delivered substantial improvements that made the product even more useful in companies of all sizes. SQL Server 2000 database engine enhancements included expansion of the RDBMS to portable devices, supporting multiple instances of SQL Server on a single computer, new and expanded integrity checking mechanisms such as an expanded Foreign Key constraint and Instead Of triggers, better performance, easier replication management, and a new mass notifications functionality called Notification Services. And with its continued BI improvements, SQL Server 2000 battled for leadership in a market that it had entered only one version earlier.

RDBMS Everywhere: From Portable Devices to Datacenters

With the expansion of Web applications and different production systems automatically collecting more and more data, the size of databases grew dramatically in the late '90s. SQL Server 2000 was prepared for the challenges. You could install SQL Server 2000 on the most powerful Microsoft operating system of that time, Windows 2000 DataCenter Server, with 64-bit architecture support. You could scale out your database across multiple servers. And you could split a table's rows across multiple tables, which could reside in different databases on different servers, and then union all of the rows together through distributed partitioned views (DPVs). SQL Server 2000 also featured many other performance improvements, such as letting you materialize views and computed columns (i.e., store the result sets in the database) by indexing them.

As many companies started Web and application hosting, they needed many SQL Server instances. Installing them on dozens of computers and purchasing the associated licenses would have simply been too costly. But SQL Server 2000 let you install up to 16 instances on a single computer, with all the licenses covered by a single Enterprise Edition (EE) license. And on the lower end, SQL Server Compact Edition (CE) worked on CE devices and supported merge replication, extending disconnected applications to nearly any device.

Besides peak performance, the other indispensible requirement for mission-critical applications, especially Web applications, is high availability. As one of its high-availability features, SQL Server 2000 supported up to four-node clusters for failover clustering. The setup procedure was completely cluster aware—no more following a lengthy white paper to properly install SQL

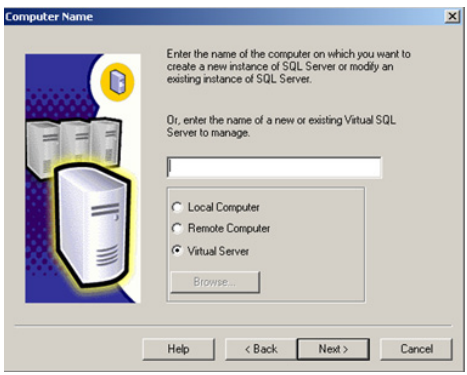


Figure 3: Installing SQL Server 2000 on a Virtual Server for Failover Clustering

Server in a cluster. Figure 3 shows SQL Server 2000's cluster-aware installation wizard.

For database developers, user-defined functions (UDFs) were a valuable addition to T-SQL in SQL Server 2000. In addition, the product recognized XML's growing popularity as a data-interchange format. For the many applications that needed data from SQL queries in XML format, Microsoft extended the SELECT statement in SQL Server 2000 with the FOR XML clause. Instead of a relational table, the result of such a statement was XML. Of course, you couldn't use this clause if you needed your data in relational format (e.g., in a view definition). However, being able to retrieve the data in XML format let you skip a step in transforming relational data from tabular format to XML format at the middle tier or client application, thus enabling faster development and better performing applications.

SQL Server 2000 improved data integrity from the referential integrity point of view by extending the Foreign Key constraint to cascading options. A foreign key prevents orphaned children in one-to-many relationships between entities. For example, a foreign key prevents orders without customers. SQL Server achieves this integrity through four rules: preventing inserts without a parent existing on the child side, updating the parent key on the child side to a value that does not exist on the child side, preventing deletion of a parent with children, and preventing update of the parent key on the parent side if the parent has children. Although a Foreign Key constraint in SQL Server 7.0 implemented these four rules, the implementation led to a lot of coding if you needed to delete a parent with children. For example; you had to delete the children first, probably in a stored procedure. SQL Server 2000 added two more possibilities for the last two referential rules on the parent side. With the new CASCADE option, you could specify deletion of the children if their parent was deleted or automatic propagation of the parent key change to all children.

SQL Server 2000 also introduced Instead Of triggers. Before this version, triggers in SQL Server could fire only after the transaction. With Instead Of triggers, you can intercept the original data modification command, check for data integrity issues, and reapply a correct command. In addition, you can intercept the data modification commands directed to a non-updateable view and send modification commands to underlying tables, thus making the view updateable.

Replication in the new version added support for automatic management of identity ranges in merge replication scenarios. In SQL Server 7.0, snapshot and transactional replication already allowed updates on the replica database (i.e., on the subscribers). However, these updates were executed through a distributed transaction synchronously on the source (i.e., on the publisher) and on the subscriber. SQL Server 2000 added support for asynchronously updating subscribers by using queues. You could implement queues through the Microsoft Message Queue (MSMQ) service or through SQL Server tables.

A couple of years after SQL Server 2000's initial release, Microsoft added SQL Server Notification Services (SSNS) to the version as a free, downloaded component for Standard or Enterprise customers. SSNS was more a development tool than an application, letting developers quickly build applications that needed to perform mass notifications of subscribers.

Battle for BI Leadership

On the BI side, SQL Server 2000 became a serious player. The version considerably improved OLAP cubes, and you could now define security settings on the lowest possible granularity level–on members of a dimension and even on cells of a cube. SQL Server 2000 added an im-

portant aggregate function, the Distinct Count function. Distinct count is important for market basket analysis, for example, which is based on the theory that if a customer buys a certain group of items, he or she is more (or less) likely to buy another group of items. In the new RDBMS version, dimensions could support hierarchical relations if you used the adjacency model in your relational tables. In the adjacency model, you use an entity key, a parent entity key, and a foreign key between the entity and parent entity keys to model a hierarchy. Examples of hierarchies are employees and their managers, bills of material, and so on. To use such a hierarchy for a report, you have to expand the adjacency model in a tree. In SQL Server 2000, this could be done automatically by so-called parent-child dimensions. You could use parent-child dimensions for ragged and unbalanced dimensions as well. Figure 4 shows a hierarchy modeled in an adjacency model.

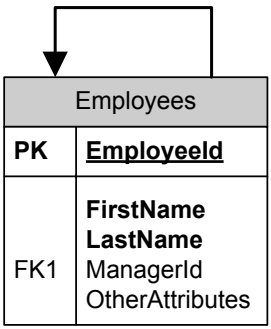


Figure 4: The Adjacency Model

OLAP analysis isn't the only kind of analysis organizations need to perform. With OLAP, for example, you have to define the model in advance inside OLAP cubes through star schema. Then, you can do model-driven analysis, drilling down through one attribute at a time. To create a view of fact data over a specific combination of dimension attributes to gain more insight than a view over a single attribute, you have to work with a client tool for a while, especially to find the most meaningful combination of dimension attributes. Wouldn't it be nice to have a tool that could find the most meaningful combination for you automatically? That functionality is what a technology called data mining provides. With data mining, you analyze your data by using advanced mathematical methods to find meaningful patterns and rules (i.e., a meaningful model). Thus, data mining provides data-driven analysis.

SQL Server 2000 added support for two of the most popular data mining algorithms: the Decision Trees and Clustering algorithms. And because the OLAP service wasn't just for OLAP analysis anymore, Microsoft renamed it Analysis Services and renamed OLAP Manager to Analysis Manager.

Data mining wasn't mature in Analysis Services 2000. It was Microsoft's first foray into the data mining world and couldn't compare with older products. However, it paved the road for the future. Competitive tools were very expensive. And you could imagine the same revolution coming in the data mining market as happened in the OLAP market with SQL Server 7.0—data mining would sooner or later be available for the masses.

As noted earlier, ETL applications can be complex, and the better tools you have, the easier it is to develop such applications. In the ETL space, SQL Server 2000 brought significant enhancements to DTS. Standing out among the many new DTS tasks was the Dynamic Properties task, which let you read connection settings, table names, and just about anything else for your DTS package from an .ini file, a global variable, an environment variable, a query, a constant, or a data file. The Data Pump task became a Multiphase Data Pump, letting you customize the data pump at various phases of its operation and add functionality such as:

- row-level restartability
- individual handling of insert or transformation errors, such as constraint violation errors

- customization of initialization or termination steps

Using the same release process as for SSNS, a couple of years after SQL Server 2000's initial release, Microsoft added reporting functionality to the product through a free downloadable component called SQL Server Reporting Services (SSRS). As with SSNS, SSRS was free for users with a Standard or Enterprise Edition license. SSRS, still a valuable component in SQL Server 2008, is a platform for creating and managing reports for relational as well as BI needs. Not all users need advanced BI capabilities such as OLAP and data mining, and SSRS adds another way to fulfill business needs for gathering useful information from a sea of data. However, many users didn't appreciate that the only development tool for SSRS in SQL Server 2000 was Visual Studio (VS)—and that VS wasn't shipped with SSRS. If you didn't have a valid VS license in place, SSRS wasn't exactly "free" for you.

A less fortunate part of the BI suite in SQL Server 2000 was English Query. With English Query, you could write applications that accepted users' input entered as English language questions and then transformed the questions into database queries. English Query's life span was very short; it existed only in SQL Server 2000, never making it to the next version of the product.

SQL Server 2005: Yukon

SQL Server 2005, code-named Yukon, was the next release of the database system, and it was a major one. Microsoft rewrote most of the BI suite from the scratch, for example, which is the main reason there are five years between Shiloh and Yukon. Among the many new features and enhancements in SQL Server 2005 are Common Language Runtime (CLR) and XML support in the database engine, new transaction-isolation levels, rewritten tools, improved performance and reliability, and security that meets modern needs. From the BI perspective, the new version delivered a rewritten Analysis Services with mature data mining capabilities, a completely new ETL tool called Integration Services, and improvements in Reporting Services.

Overcoming T-SQL Limitations

Many DBAs' first reaction when they started to explore the novelties in Yukon was fear. SQL Server 2005 is the first version of the product not limited solely to the T-SQL language. Yukon adds support for CLR languages and for XQuery. Using CLR languages such as Visual Basic .NET and Visual C#, you can write stored procedures, UDFs, triggers, and two new objects that you cannot write in T-SQL–user-defined aggregate functions and user-defined data types.

DBAs were especially afraid of the CLR language support, airing their concerns about security and performance. T-SQL is a limited language—limited to data operations inside SQL Server only. With CLR code, developers can do nearly anything from inside SQL Server, thus introducing a whole new set of security issues. In addition, DBAs were concerned that with CLR code, developers would move from efficient set-based operations in SQL Server to row-based processing and thus introduce a lot of new performance issues in database applications.

However, Microsoft had not forgotten about security. DBAs were quickly comforted by the fact that they do not have to deal with security in detail. For CLR code, SQL Server 2005 introduced three permission sets: SAFE, EXTERNAL_ACCESS, and UNSAFE. The SAFE set, the recommended permission set, limits CLR code to operations that T-SQL can do. With the EXTERNAL_ACCESS set, code from SQL Server can access external resources such as the file system and network resources. With the UNSAFE set, CLR code can do nearly anything, including directly writing to memory, so it can endanger SQL Server itself. Of course, you should have a good reason to use any permission set except SAFE and be especially careful with the UNSA-

FE set. In addition, CLR is disabled by default; a DBA must enable it for an instance by using the sp_configure system stored procedure. Concerns about poor performance of CLR code also disappeared over time; DBAs realized that a developer can write good or bad code in any language.

Besides adding the capability of creating CLR data types, SQL Server 2005 introduced some new system data types. Probably the most important among them is the XML data type. You can finally store XML documents where they should have been from the start: in a relational database. Because XML is a large data type, letting you store up to 2GB of data, customers need a way to retrieve and update just part of it. This is where XQuery comes in. XQuery is a query language for traversing XML nodes, finding elements and attributes, and even looping through nodes at a selected level. SQL Server 2005 implements XQuery support through XML data type methods. The XML data type has methods to retrieve and update parts of XML data, and XQuery expressions are parameters of those methods. The methods are:

- query(), which returns part of the XML data in XML format
- value(), which returns a scalar value of an element or an attribute of an element
- exist(), which returns 1 if an element or an attribute exists in an XML instance, 0 if an element or an attribute doesn't exist, and null if the XML data type instance contains null
- modify(), which lets you insert an element or an attribute, delete an element or an attribute, or update the value of an element or an attribute
- nodes(), which lets you shred an XML data type instance into relational data

Microsoft wasn't just concerned about CLR code security in SQL Server 2005. Yukon contains many other security enhancements, including properly implementing schemas—as namespaces for database objects. You no longer have to use object owner to group objects anymore. All we can say is—finally! Schemas should have been introduced years earlier. You can now also set permissions on a schema, and the permissions are inherited on objects inside the schema, making the job of setting up and maintaining permissions much easier.

Other security enhancements include SQL logins that are finally protected; SQL Server 2005 enforces Windows account and password policies for SQL logins. In addition, triggers are no longer limited to data modification operations; a DBA or developer can write data definition language (DDL) triggers to, for example, prevent table modifications in a production database. Data encryption is also now possible with new encryption functions and complete key and certificate support. However, using encryption in SQL Server 2005 means upgrading your applications. Yukon adds many new detailed permissions as well, and you can overcome problems with broken ownership chains by specifying the execution context for programmable objects. Figure 5 show the complete key and encryption hierarchy.

From the manageability point of view, the difference between Yukon and previous versions is immense. With new catalog and dynamic management views and functions, a DBA has detailed insight into SQL Server operations, state, and resource usage in real time, with T-SQL queries. In addition, a DBA can use a dedicated admin connection, which runs in a higher thread priority, to quickly kill a runaway session. You can now gather and save execution plans in XML format, which is simple to consume within an application. This ability simplifies the viewing of a large number of plans and lets you easily use the plans as input for support.

The SQL Server community also needed some time to really appreciate another Yukon feature: completely rewritten tools. The primary DBA tool is now SQL Server Management Studio

(SSMS), a centralized tool for managing the database engine, SQL Server Analysis Services (SSAS), SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSIS), and mobile edition servers. SSMS also lets you write queries. SSMS replaces Enterprise Manager, Analysis Manager, Query Analyzer, and other tools. Because SSMS is based on VS, you can easily add versioning support to it by using Team System Server or Visual SourceSafe.

Besides CLR and XQuery support, developers also received new T-SQL elements in SQL Server 2005. New operators—including PIVOT, UNPIVOT, and APPLY—can help abbreviate your code. And the version boasts new ranking functions, including ROW_NUMBER(), which have topped wish lists for years. Yukon introduces Common Table Expressions (CTEs) as another way of using subqueries; with them, you can introduce subqueries in advance, in the WITH clause, before the outer or main SELECT statement. In addition, with CTEs, you can expand hierarchies modeled as the adjacency model.

Asynchronous applications are simpler and more reliable than ever with SQL Server Service Broker, a new message queuing system inside SQL Server 2005. An application can subscribe to query notifications; with query notifications, SQL Server informs an application that the cached data is outdated. And to enable applications in which readers do not block writer and to mitigate moving applications from Oracle, Yukon adds two new transaction-isolation levels: Snapshot and Read Committed Snapshot. Thus, SQL Server is no longer limited to the pessimistic locking approach only; these two levels implement an optimistic approach, storing old versions of rows before updates in the tempdb system database, then driving SELECTs through these old versions.

Yukon delivers new performance features as well. You can now physically partition a table horizontally while still referring to it logically in queries by using a single table name. This capability makes partitioning inside a database much easier than with partitioned views. Another new ability in SQL Server 2005 is related to database snapshots, which are read-only snapshots of a production database at a specific time point. You can now drive reading through database snapshots and, with them, have another way of implementing reading that doesn't block writing. And the Index Tuning Wizard (ITW) has matured into the Database Tuning Advisor, which gives better optimization recommendations, including suggestions for using table partitioning.

SQL Server 2005 also provides new and improved high availability technologies. You can implement up to eight nodes in a cluster configuration. You can also now perform indexing online, meaning that applications can use a table while it is being indexed. The online indexing feature uses the same row versioning framework used to implement snapshot transaction isolation levels. Last but not least in the area of new high availability features in Yukon is database mirroring, which is essentially an inexpensive alternative to clustering. Database mirroring maintains two copies of a single database that must reside on different instances of SQL Server, which can reside on computers in different locations. One server instance, the principal server, is the production server, while the other server instance, the mirror server, acts as a hot or warm standby server. Automatic failover is possible if a third server instance, the witness server, is present.

Rewritten BI Suite

SQL Server 2000 gained a well established position among major players in the BI market. However, Microsoft decided to rewrite almost all products in the SQL Server 2005 BI suite, including the development tools. The new Business Intelligence Development Studio (BIDS) is just another name for VS. The tool, which includes templates for all BI projects, now ships

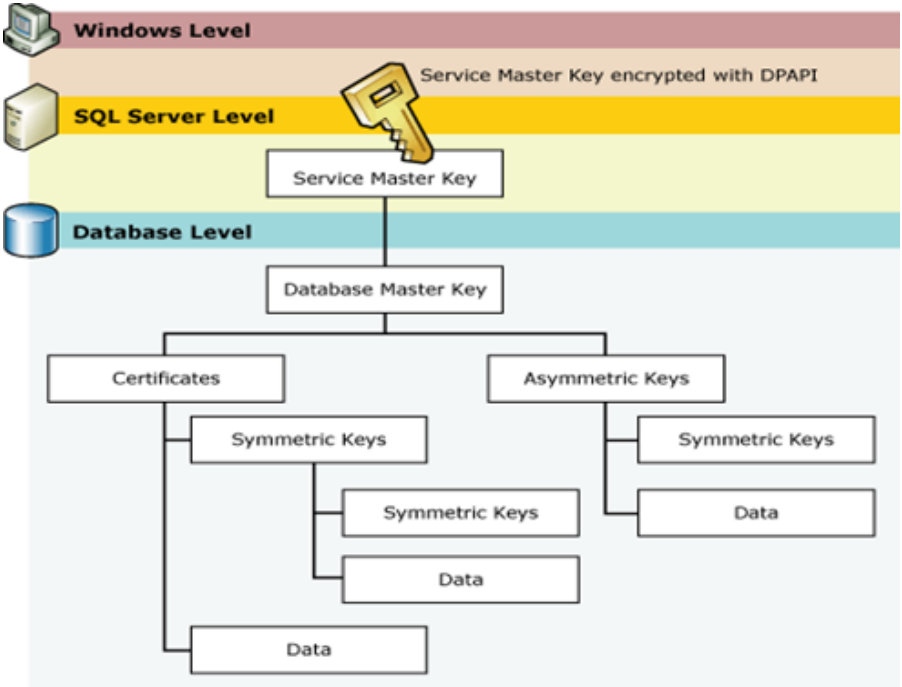


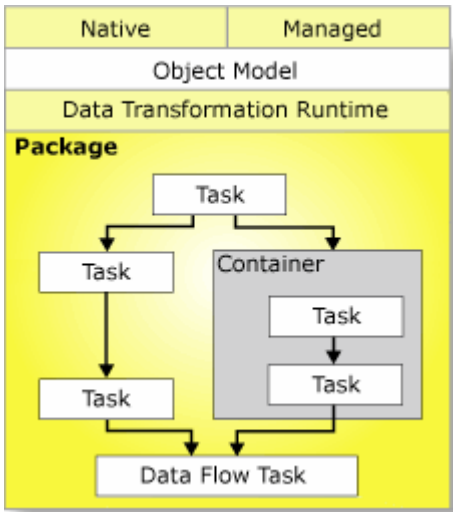
Figure 5: Key and Encryption Hierchy in SQL Server 2005

with SQL Server, providing a separate tool from SSMS for developing BI applications. Version management of BI projects is simple, using the same version management as any VS project. Customers that do not have a valid VS license no longer have to purchase it just for SSRS projects, as they had to in SQL Server 2000. In addition, SQL Server 2005 extends Profiler, letting you trace SSAS as well as the database engine.

Although as one of the newer BI additions, SSRS didn't need major changes, the reporting component lacked a simple tool that let advanced end users author reports. End users can't use VS, and they don't know how to write database queries. So Microsoft offered a solution. In SSRS 2005, DBAs can create report models, semantic descriptions of database metadata that serve as an intermediate layer between report author and data. Report authors can then create reports by using business entities and attributes, and SSRS uses the report models to transform the report requests into database queries. End users create ad hoc reports by using Report Builder, a new, lightweight and simple-to-use report authoring tool. In addition to the report the end user creates, SSRS automatically generates click-through reports, letting consumers follow navigation paths based on foreign keys that exist within the report model.

As noted earlier, in SQL Server 2005, SQL Server Integration Services (SSIS) replaces DTS. The difference is enormous. SSIS is an actual ETL tool. Besides control flow and tasks, it features a separate data flow engine. Inside a data flow, which you create by using the Data Flow task in the control flow, you can connect to different sources, read the data, store it in memory in tabular format, work on this data through multiple transformations, and finally load it to one or more destinations. The Data Flow task is actually Data Pump on steroids. SSIS includes both basic and advanced transformations that

you can use to add a lot of intelligence to your ETL package. The Fuzzy Lookup transformation, for example, can match rows from different sources based on similarity of character columns, and the Fuzzy Grouping transformation is handy for de-duplicating rows based on similarity of strings. The Data Mining transformation lets you add predictions from a data mining model to your data flow and later filter the data based on those predictions. And Term Extraction and Term Lookup transformations are useful together with SSAS data mining for text mining. The Slowly Changing Dimension transformation isn't actually a transformation; it is a wizard that generates all the transformations you need for maintaining changes in dimension data over time. Inside a data flow, a package need not fail on erroneous rows when executed; instead, you can define separate error flows and either correct or log erroneous rows. Figure 6 shows the relationship between control flow and data flow.



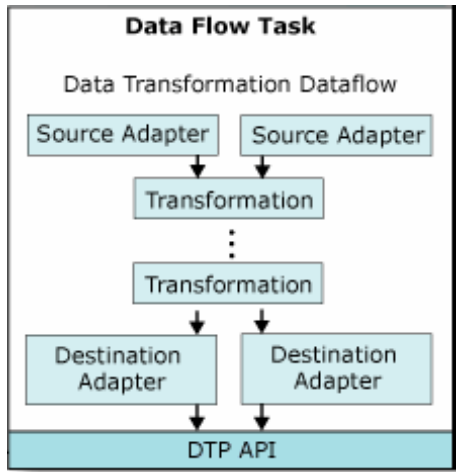


Figure 6: SSIS Control Flow and Data Flow

Besides tasks and precedence constraints, control flow also includes containers. Two containers, For Loop and For Each Loop, allow looping of tasks. You can base precedence constraints not only on success, failure, or completion of a task, but you can now include custom logical expressions to define the flow exactly as you want it. These features together with package variables, event handlers, extensive logging, restartability, external package configurations, and improved security make SSIS a high performance enterprise ETL tool.

In SSAS 2005, the term OLAP is replaced with Unified Dimensional Model (UDM). UDM isn't just a marketing term; metadata—the schema of OLAP cubes—is enhanced so that it is richer than relational schema in a data warehouse. Therefore, you do not need to first create a data warehouse, and then inherit star schemas in OLAP cubes; you can start building your dimensional model from either the SSAS or relational engine side. Thus, SQL Server 2005 delivers true unified dimensional modeling. Also in the new version, a single OLAP cube can include multiple fact tables. Because a cube can now represent a complete data warehouse, SSAS 2005 in-

cludes additional objects called Perspectives. Perspectives are UDM views that let users see only the part of the cubes they are interested in. In the new version, you can pivot data and drill down using any attribute of a dimension, not only attributes that are part of a hierarchy. For example, you aren't limited to drilling down only through the path country-region-city-customer; you can also drill down from country to customer account manager, then to size of company, and so on. Inside the UDM, you can define translations for data and metadata. Besides storing calculations centrally in your UDM, you can also store Key Performance Indicators (KPIs). By centralizing KPIs and attribute properties, such as format strings and presentation information, you can maintain a single version of truth in the enterprise. No matter where your UDM data is retrieved from and with which client tool, it is always presented in a consistent way, thus unifying the perception to end users. Note that you can also install multiple instances of SSAS in SQL Server 2005. And the number of supported instances on a single box has risen from 16 for SQL Server to 50 for SQL Server and 50 for SSAS. Finally, you aren't limited to a scheduled refresh of OLAP data if you store it in an UDM; UDM supports proactive caching.

SSAS 2005's data mining features underwent an even bigger overhaul. Data mining in version 2005 isn't just a teaser; it is a mature product, integrated with other SQL Server components. SSAS ships with all the most popular data mining algorithms:

- Association Rules
- Decision Trees, which includes Regression Trees
- Naïve Bayes
- Neural Networks
- Clustering
- Sequence Clustering
- Linear Regression
- Logistic Regression
- Time Series, a forecasting Auto-Regression Trees (ART) algorithm

Although not part of the SQL Server suite, Microsoft Office Excel 2007 now serves as a first-class OLAP client—and with freely downloadable data mining add-ins, as a data mining client as well. In addition, Excel 2007 with data mining add-ins can use SSAS data mining algorithms to analyze spreadsheet data.

SQL Server 2008: Katmai

The SQL Server story continues in 2008 with SQL Server 2008, code-named Katmai. While perhaps not as revolutionary as SQL Server 2005, Katmai nevertheless is an important step forward in satisfying ever growing data-management needs. New system data types support spatial applications and sparse data and improve date and time handling. The MERGE statement, change tracking, and change data capture (CDC) simplify data warehouse maintenance. Policy-based management, resource control, and performance data collecting enable DBAs to work more efficiently. SSIS 2008's new Data Profiling task lets you check data quality before you actually start your ETL process. And the entire BI suite—including SSIS, SSAS, and SSRS—gets a performance boost.

Supreme Database Engine

Having CLR support in the database engine enabled Microsoft to use it to ship additional data types in SQL Server 2008. Katmai supports spatial data and spatial applications with Geometry and Geography data types, and maintaining hierarchies is simplified with the HierarchyId data type. For large objects, you no longer have two mutually exclusive options: storing them in the database or in the file system; you can now use the new Filestream storage attribute to store unstructured data, such as documents and images, on the file system while working with the attribute as you do with regular large object database data types. The benefit of using

Filestream data is that you don't inflate your database files with large objects, but you can administer large objects together with other database objects, including performing backups and setting up security. In SQL Server 2008, full--text indexes are integrated in the database as well, making them easier to maintain and providing better performance for full-text searches.

SQL Server 2008 also delivers one of the most requested features for years: new Date and Time data types that let you store data and time separately. Additionally, the new Datetime2 type has a larger date range and a larger default fractional precision than the old Datetime type as well as optional user-specified precision. The new Datetimeoffset type offers another way to handle date and time, defining a date combined with a time of a day that has time zone awareness and is based on a 24-hour clock. When you create a table, you can also define some columns as sparse columns. Sparse columns are ordinary columns that have optimized storage for null values. Sparse columns reduce the need for separate subtype tables for the not-applicable attribute.

Pure database developers will be happy with the extended T-SQL language features in SQL Server 2008. Stored procedures and UDFs can accept table-valued parameters. The new MERGE statement, also known as UPSERT, can do multiple data modifications at a time. Based on a source table, you can update matched rows in the destination table, insert rows that do not exist in the destination table yet, and delete rows from the destination table that do not exist in the source table anymore. This is especially useful for maintaining a data warehouse. Two other options target data warehouse scenarios as well: Change Tracking (CT) and Change Data Capture (CDC). With CT, you can track whether there were changes in source tables, and then efficiently use MERGE statements to implement the changes in your data warehouse. However, what happens if you don't have any attribute useful for row matching in the source and destination tables? This could occur if the primary key changed in the source table. In that case, you could use CDC, which captures all the changes in additional system tables. Because the capturing feature uses data from the transaction log asynchronously, it is has much lower performance impact on the source server than if you implemented this feature using data modification language (DML) triggers.

For application developers, Katmai supports the Entity Framework and Language Integrated Query (LINQ). Both application and database developers will appreciate the version's CLR enhancements, especially large user-defined data types, which aren't limited to 8K anymore, and multi-parameter user-defined aggregate functions. Database encryption is transparent for applications in Katmai, meaning you should be able to use it without application modifications.

On the DBA side, there are also plenty of reasons to look forward to Katmai. Take a look at Figure 7, which shows some interesting new folders in SSMS. The new Policy Management folder and subfolders are where you implement the Policy-Based Management (PBM); the Data Collection folder lets you set up performance data collection; and the Resource Governor folder and subfolders let you control resources used by SQL Server operations.

The PBM is a policy-based management framework for the SQL Server database engine. It ensures compliance with policies for system configuration, enables preventing and monitoring changes to the system, and reduces total cost of ownership by simplifying administration tasks. Resource Governor is a new technology that lets you manage workload and resources by specifying limits on resource consumption by incoming requests. This feature is useful, for

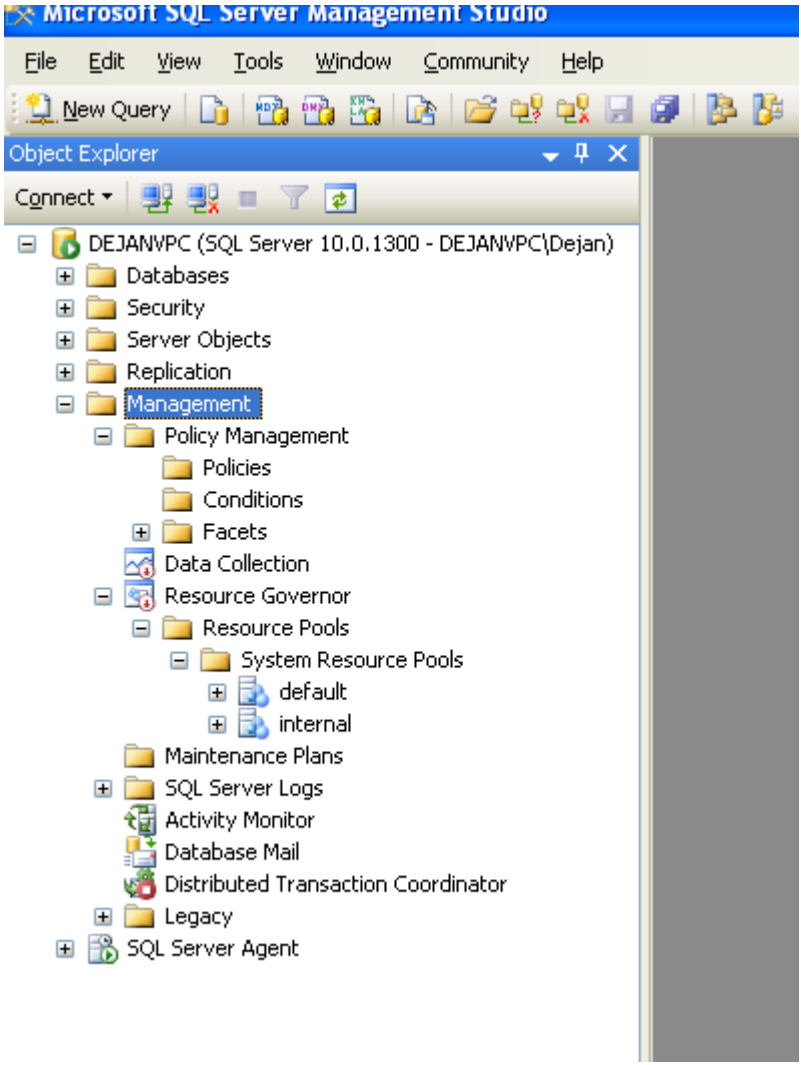


Figure 7: New Management Folders in SSMS

example, if you need to run a huge query during peak hours and you do not want the query to consume all the resources, such as memory and CPU. DBAs will also find Katmai's automatic performance and other management data collection features valuable. You can collect this data in a Management Data Warehouse (MDW), which you can create—together with the required SSIS packages and SSRS reports for analyzing the data—with a couple of mouse clicks in SSMS. All these features together comprise the new Performance Studio.

In addition, SQL Server 2008 lets you optimize database file space consumption with data and backup compression. And new filtered indexes

let you index only interesting subsets of rows, thus making indexes smaller and more effective.

Note that SQL Server Notification Services (SSNS) doesn't exist in SQL Server 2008. Apparently Microsoft realized that not many customers were using the component. However, users of existing SSNS implementations aren't going to be happy with this decision.

Mature BI

The BI suite in SQL Server 2008 continues to mature, with SSIS, SSAS, and SSRS seeing advanced features, usability enhancements, and performance improvements.

Probably the most important new feature in SSIS 2008 is the Data Profiling task. Before you transfer source data to your data warehouse, you should always perform some quality checks. You can use the Data Profiling task to measure the following:

- Column length distribution, which reports all the distinct lengths of string values in the selected column and the percentage of rows in the table that each length represents
- Column null ratio, which reports the percentage of null values in the selected column
- Column pattern, which reports a set of regular expressions that cover the specified percentage of values in a string column
- Column statistics, which reports statistics, such as minimum, maximum, average, and standard deviation for numeric columns and minimum and maximum for datetime columns
- Column value distribution, which reports all the distinct values in the selected column and the percentage of rows in the table that each value represents; can also report values that represent more than a specified percentage of rows in the table
- Candidate keys, an option that reports whether a column or set of columns is a key, or an approximate key, for the selected table (i.e., measures uniqueness of a column or set of columns)
- Functional dependency, which reports the extent to which the values in one column (the dependent column) depend on the values in another column or set of columns (the determinant column)
- Value inclusion, which computes the overlap in the values between two columns or sets of columns; this profile can determine whether a column or set of columns is appropriate to serve as a foreign key between the selected tables, or if all keys in a subtype are included in the pertaining supertype

The main focus of improvements in SSAS 2008 is performance. Performance is the key in all phases of the SSAS database life cycle, starting with design. OLAP models can be complex. And best practices and performance tips aren't generally well known, so designing the models correctly isn't easy. In SSAS 2008, performance best practices are embedded in the design process. When you design a UDM database in BIDS, you get more than 40 best practices warnings integrated into real-time designer checks. You see blue squiggly lines and receive build-time warnings if your model isn't in accordance with performance best practices. In addition, the Dimension Designer has a new Attribute Relationship Designer. The Cube Wizard doesn't create all possible attribute hierarchies by default, so you don't automatically get aggregates over attributes that you never use for drilling down, such as customer address and phone. The Attribute Relationship Designer also gives you more control over the design of aggregates. When a UDM is deployed, a DBA can monitor performance through new Dynamic Management Views (DMVs). SSAS also exposes server resource information as a cube that you can use to perform resource analysis.

SSAS 2008 also boasts performance improvements for cubes in production. Calculating expressions on sparse cubes is optimized so

much that you can feel the magnitude of improvement compared to SSAS 2005—and many OLAP cubes are sparsely populated. If you write back to cubes—for example, for a planning and budgeting application—in previous versions, write-back was done in the relational data warehouse; reading relational data was magnitudes slower than reading OLAP data. In SQL Server 2008, multidimensional OLAP storage (MOLAP), a variant where you store everything (including metadata, aggregates, and data) in an SSAS database, isn't read-only anymore. You use MOLAP write-back, thus gaining a lot in performance. SSAS 2008 also supports scalable read-only shared databases; a database can use SAN storage and is shared between multiple Analysis Services instances. Backup speed is also improved.

Katmai brings a couple of data mining improvements as well. Time Series analysis now supports the Auto-Regressive Integrated Moving Average (ARIMA) algorithm in addition to the ART algorithm. ART is better in short-term forecasting, and ARIMA performs better for long-term forecasts. You can use a mix of ART and ARIMA forecasts in a single mining model. Microsoft will also upgrade Office 2007 Data Mining Add-Ins will to support SSAS 2008l. In Figure 8, you can see Excel 2007 as a data mining client tool for SSAS 2008.

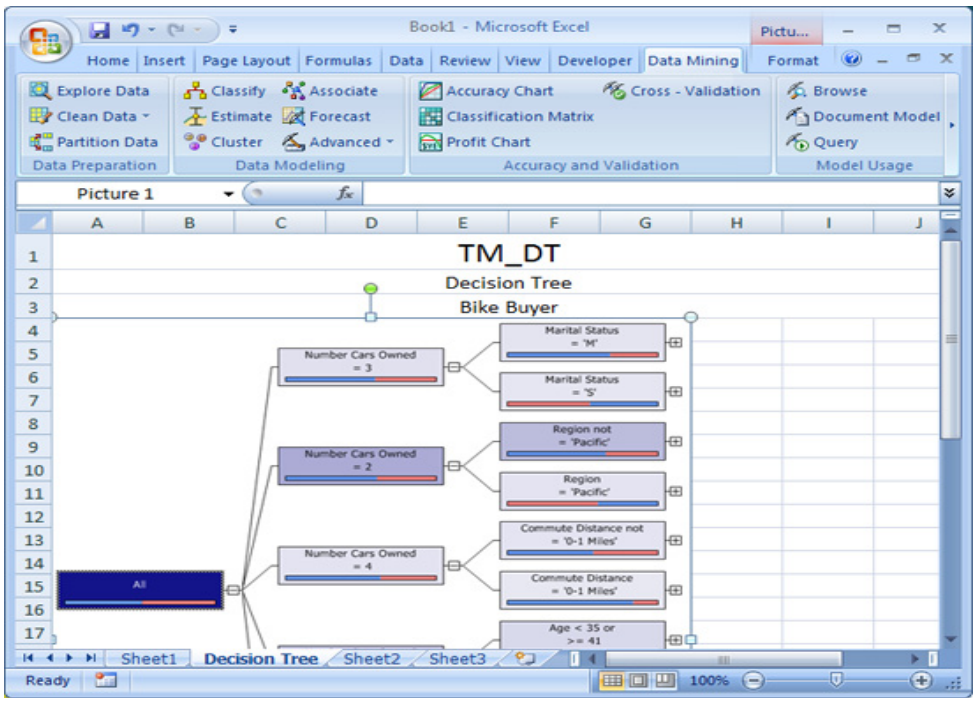


Figure 8: Excel 2007 as SSAS 2008 Data Mining Client

In SSRS 2008, I want to point out two important features in particular. The first one is the Tablix data region. A Tablix data region combines features of both a table and a matrix, letting you organize data both by column and row groups. Some features enabled with the Tablix data region include the possibility of having stepped columns with crosstab reports—for example, customer country and region can share the same column; using side-by-side crosstab sections—for example, product category and color;

and having independent group aggregates with crosstab reports. The other exciting SSRS 2008 feature is the new authoring tool. Report Designer is now a standalone application with an Office 2007 look and feel. It is easier to use than VS, yet it doesn't lack any functionality.

SQL Server 2008 R2: Kilimanjaro

The 2008 R2 version, released in 2010, is not treated as a complete new version; it is rather an enhancement of SQL Server 2008. Most of new features are focused on BI part. Microsoft introduced couple of completely new products in the BI suite. Some minor enhancements can be found in the database engine as well. Besides new features, Kilimanjaro has also refurbished editions. The 2008 R2 editions include:

- Premium editions
  - Datacenter: 256 logical CPUs, unlimited virtualization,...
  - Parallel Data Warehouse: data warehouse appliance, MPP technology
- Core editions
  - Enterprise: OS max CPUs, compression, partitioning …
  - Standard: up to 4 CPUs, backup compression (new) …
- Specialized editions
  - Developer: all Datacenter features, license for development only
  - Web: 4 CPUs, unlimited RAM
  - Workgroup: 2 CPUs, 4GB RAM
  - Express: 1 CPU, 1GB RAM, 10GB db size
  - Compact: mobile systems
  - Evaluation: all Enterprise features, evaluation license only

Small Delights in the Database Engine

Compression in pre-SQL 2005 edition was limited to variable-length data types. SQL Server 2005 added vardecimal type, i.e. decimal type stored in variable-length format. SQL Server 2008 added row compression and page compression. SQL 2008 R2 brings Unicode compression. It works on nchar(n) and nvarchar(n) data types. There is no special command to turn on the Unicode compression; it is turned on automatically with row or page compression. The actual savings depends on language; can be up to 50% in English or German and only 15% in Japanese. Nevertheless, it makes sense in nearly any language, as the performance penalty for using it is very low.

Data Tier Application (DAC) is a single unit of deployment that contains all of the database's schema, dependent objects, and deployment requirements in a single zipped XML file. You author SQL Server data-tier application project in Visual Studio 2010, or extract it from an existing database with the Data-Tier Application Wizard in SSMS. Deployment of data applications is simple, and the collaboration between data-tier developers and DBAs is significantly improved with DAC. However, not all objects are supported in DAC in this version. Also, because of security, SQL logins' passwords are not saved.

Finally, there is SQL Server Utility – and utility to centrally monitor and manage database applications, SQL Server instances, database files, and volumes. It has a management interface – Utility Control Point in SSMS. It collects configuration and performance information every 15 minutes. In SSMS, you can visualize the information collected through SQL Server Utility dashboard and viewpoints. They provide health summary of SQL Server resources through policy evaluation and historical analysis. The data collected is stored in Utility Management Data Warehouse (UMDW) , named sysutility_mdw.

BI: Enhancements

From existing BI suite, SQL Server in version 2008 R2 brings additional useful features only in Reporting Services. Associations between

datasets from different sources are not limited on report models, i.e. on data source views, anymore. Regular reports still do not use data source views, and datasets are still limited on a single rowset; however, SSRS has new built-in functions you can use in expressions to denote associations between datasets.

From version 2005 we can use shared data sources. However, up to 2008 R2, this is the only thing besides images we can share. In 2008 R2, you can share also datasets and report parts, like Tablix or Chart controls. This way, you can speed up and simplify report development be reusing these shared report parts.

From visualization perspective, Maps are the most important enhancement. Data sources for maps include:

- SQL Server spatial data type
  - Point: Exact location defined as (X,Y) coordinate
  - Line: Connected sequence of points
  - Polygon: Two-dimensional shape having same start and end point
- Bing map tiles
  - Provides road, aerial, or hybrid view as background for map layers
  - Requires report server configuration to support Bing Maps Web Services
- ESRI shape file
  - Complies with Environmental Systems Research Institute spatial data format
  - Contains data in a pair of files defining geometric or geographic shape and attributes for those shapes.

Figure 9 shows in a quite condensed way some of the options you can use to represent spatial data with maps in SSRS 2008 R2.

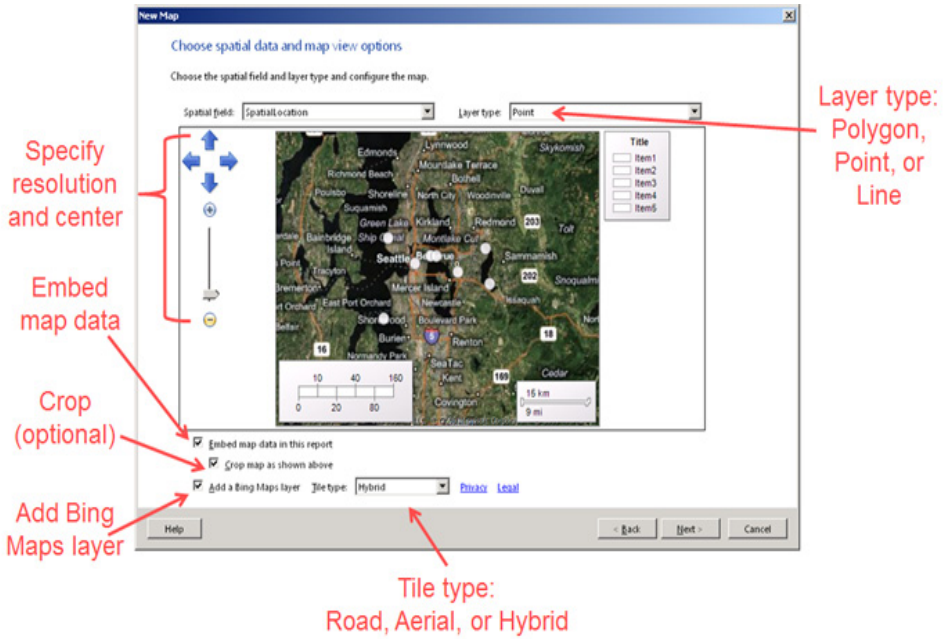


Figure 9: Options for Spatial Data in Map Wizard

Small delights in visualization include sparklines, data bars and indicators. A new rendering

extension supports ATOM data feeds. Report Builder 2.0 is upgraded to 3.0, which is even simpler to use. Still, the target for this authoring tool is an advanced end user. However, because of usage of shared report parts and new wizards, authoring reports is faster and simpler than in version 2.0.

BI: New Products

In an average BI project, we spend most of the time in data cleansing. This is especially important for dimensions, which give context to transactional data. Master data is generally a quite broad term; for BI purposes, we can make an approximation and say that the dimensions data is our master data. For extract – transform – load (ETL) processes, SSIS is the tool. However, I want to mention also Master Data Services (MDS) in this place. Although this is not an ETL tool, MDS can help with data quality, and might lower the need for data cleansing and merging.

MDS is a product that coordinates process of managing master data, involving policies, processes, procedures and people. Goals of master data management in general include:

- Unifying (harmonizing) master data between different systems
- Maintaining multiple versions of master data
- Integrating master data for analytical and CRM systems
- Maintaining history for analytical systems
- Capturing information about hierarchies in master data
- Supporting compliance with government prescriptions (e.g., Sarbanes-Oxley) through auditing and versioning
- Having a clear CRUD process through prescribed workflow
- Maximizing ROI through re-usage of master data.

There are multiple approaches to master data management:

- No central master data management
  - Not really MDM
- Central metadata storage
  - Probably in an unstructured form – in documents, worksheets, or even on paper only
- Central metadata storage with identity mapping
  - Useful during upgrading, testing and initial usage of a new ERP system
- Central metadata storage and central data that is continuously merged
  - DW and CRM apps can be an authoritative source of master data
- Central MDM, single copy
  - Hardly possible to make an union of all needs
- Central MDM, multiple copies
  - Realistic, although involves continuous merge.

Master Data Services is a centralized master data management solution. It is an authoritative source for master data. Other applications, including BI applications, can subscribe to this data through subscription views or programmable APIs. MDS is a central storage and also provides master data management services. It brings limited data integration features through four well-known, publically available algorithms for evaluating similarities of strings.

I want to expose in this place the SSIS Fuzzy Lookup transformation. It is available in SSIS from version 2005. It enables merging data based on string similarities using proprietary Microsoft algorithm. The important part here is the quality of algorithm. Fuzzy Lookup is magnitudes better than any public algorithm, including all algorithms provided by Master Data Services. The same algorithm uses Fuzzy Grouping trans-

formation, which is very useful for de-duplication of data.

The next product I have to mention is StreamInsight. StreamInsight is a set of .NET classes that allow real-time analysis of in-memory data streams. For analysis, it uses LINQ language, which is quite similar to SQL. With StreamInsight, it is possible to identify patterns and relationships from seemingly unrelated events on-line, in real time. StreamInsight is appropriate for monitoring and processing streams of events that pass by synchronously, for example from some machines.

Fast Track Data Warehouse (FTDW) is a pre-built and tuned hardware and SQL Server. Everything is optimized for star schemas and sequential IO. As the RDBMS is SQL Server, this is a logical next step if SQL Server Enterprise is not enough for your Enterprise Data Warehouse (EDW).

Parallel Data Warehouse (PDW) is a data warehouse appliance that offers massive scalability. PDW is built on the Massive Parallel Processing (MPP) technology acquired with Datallegro, and is provided as an appliance running SQL Server on Windows Server, with standard hardware components from a number of hardware vendors – HP, IBM, Dell, Bull, and EMC. This provides a balanced solution consisting of software and hardware working together. PDW supports data warehouses that store 100's of terabytes, and beyond into the petabyte range. PDW uses multiple compute nodes with database servers and storage nodes, control and management nodes, backup nodes and landing zone nodes for loads from ETL processes.

PDW is s hub and spoke solution. PDW comprises a centralized EDW and a set of loosely coupled data marts. With PDW, you can create a diverse range of types of spoke, from SQL Server Parallel Data Warehouse MPP appliances for user groups with extreme scalability requirements, FTDW implementations, SQL Server 2008 Enterprise data warehouses, and even SQL Server 2008 Analysis Services OLAP databases. However, everything has its own price. PDW SQL language is somehow limited. Figure 10 shows the PDW architecture.

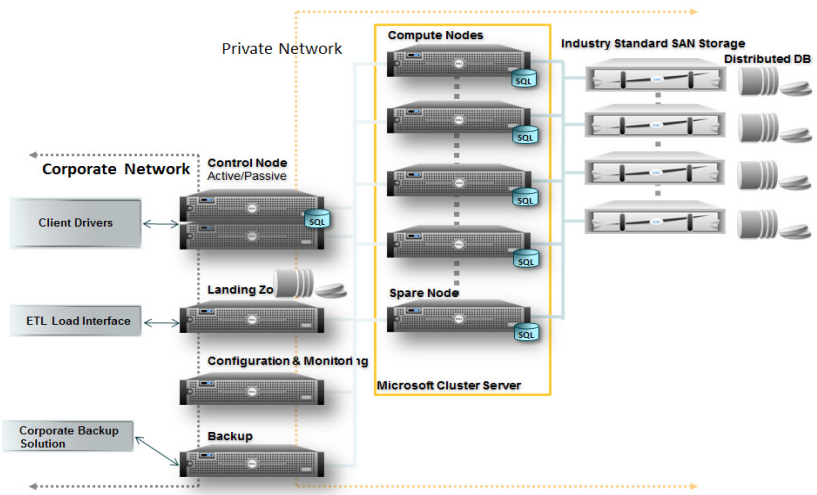


Figure 10: PDW Architecture

PowerPivot is a column-oriented in-memory database. It can live on client side as an Excel add-in, or in a SharePoint Enterprise installation through Excel Services. It gives SSAS strength to Excel, and can be used as SSAS on advanced end user (Excel) or departmental level (SPS). Howe-

ver, metadata is still far from the real SSAS. In addition, PowerPivot introduces new expression language, Data Analysis Expressions (DAX).

Work in Progress

From Version 7.0 to Version 2008 R2, SQL Server has provided and continues to provide a reliable, scalable, and secure database platform that satisfies growing business needs. However, it is much more than a relational database management system. With all its database engine and BI components, SQL Server delivers everything you need to store, maintain, integrate, and analyze your data. It is a mature, market-leading product. However, the work is never finished, and there is still room for improvement. In future releases, the first one expected in 2011, I would like to see support for temporal applications built into the database engine. And the OVER clause of the T-SQL SELECT statement would be even more useful if it fully implemented all ANSI-standard subclauses. From the tools perspective, I really miss a good database modeling tool and would like to see one included in the SQL Server suite. In the BI suite, we need more tools for checking data quality and getting an overview of the data. I would like to see more statistical algorithms implemented in SSAS, maybe through language enhancements or through system stored procedures. Based on the enhancements that Microsoft has delivered in the last four SQL Server versions, I am confident that future versions will give us even more advanced, efficient, and easy-to-use ways to meet our business needs for storing, managing, and analyzing data.

Kolofon

Izdaja

Kompas Xnet
Stegne 7
1000 Ljubljana

Telefon: 01 5136 990
Fax: 01 5136 999
Email: info@kompas-xnet.si
Web: http://www.kompas-xnet.si

Direktorica

Branka Slinkar

Urednik in oblikovalec

Gašper Kamenšek

Člani uredništva

Dejan Sarka, Herbert Albert, Dino Esposito

Osvojili smo prvo mesto! (januar 2011)
Microsoft je potrdil naš uspeh! (november 2010)
Prejeli smo posebno priznanje (oktober 2010)