Business Intelligence and Data Warehousing in SQL Server "Yukon"

Abstract: This paper provides and overview of the enhancments to the Business Intelligence platform for SQL Server "Yukon" Beta 1. This paper is not an implementation guide, but rather provides information about the enhancements to the Business Intelligence platform. The topics covered in this document are covered under the Microsoft NDA.

Author: Joy Mundy

Table of Contents

Introduction **1**
Where do I start with "Yukon" Beta 1? **3**
Relational data warehousing **4**
Reporting Server 4
Table partitions 4
Using table partitions for fast data loads 5
Using table partitions for fast data deletes 6
TSQL Improvements 6
New data types 6
New Analytic Functions 7
PIVOT and UNPIVOT operators 7
Recursive queries 7
Extract, Transformation, and Loading (ETL) **8**
Package development 8
Control Flow 10
Data Flow 11
Developing and Debugging 13
DTS "Yukon" for the DTS 2000 Developer 13
Analysis Services **13**
Create a custom database from an existing source 19
Data Sources and Data Source Views 20
Create dimensions and cubes 20
Build and deploy 21
Create a customizable database from a template 22
The Unified Dimensional Model 22
Attribute-based dimensions 23
Types of Dimensions 24
Measure Groups and Perspectives 24
Calculations and Analytics 25
MDX Scripts 26
Stored Procedures 27
Key Performance Indicators 27
Real Time Business Intelligence 28
Data Mining 30
Overview 30
Data mining algorithms 30
Build the mining model 32
Building the data mining application 33

Examples of DMX 34
Reporting Services 35
Why Use Reporting Services? 35
Ways to Use Reporting Services 36
Reporting Services Features 36
Authoring reports 37
Managing reports 37
Delivering reports 39
Summary 39
Appendix A: Code samples 40
Recursive query sample 40

Introduction

Microsoft SQL Server "Yukon" is a complete business intelligence (BI) platform that provides the features, tools, and functionality to build both classic and innovative kinds of analytical applications. This paper provides an introduction to the tools that you will use to build an analytical application, and highlights new functionality that makes it easier than ever to build and manage complex BI systems.

The following table presents an overview of the components of a business intelligence system, and the corresponding Microsoft SQL Server 2000 and SQL Server "Yukon" components.

Component	SQL Server 2000	SQL Server "Yukon"
Extract, transformation, and load	Data Transformation Services (DTS)	Data Transformation Services (DTS)
Relational data warehouse	SQL Server 2000 relational database	SQL Server "Yukon" relational database
Multidimensional database	SQL Server 2000 Analysis Services	SQL Server "Yukon" Analysis Services
Data mining	SQL Server 2000 Analysis Services	SQL Server "Yukon" Analysis Services
Managed reporting		SQL Server "Yukon" Reporting Services **(New!)**
Ad hoc query and analysis	Microsoft Office products (Excel, Office Web Components, Data Analyzer, Sharepoint Portal)	Microsoft Office products (Excel, Office Web Components, Data Analyzer, Sharepoint Portal)
Database development tools	SQL Server 2000 Enterprise Manager, Analysis Manager, Query Analyzer, various other	SQL Server "Yukon" BI Development "Workbench" **(New!)**
Database management tools	Enterprise Manager, Analysis Manager	SQL Server "Yukon" SQL "Workbench" **(New!)**

Three components are new for SQL Server "Yukon": Reporting Services, the BI Development "Workbench," and the SQL "Workbench". The other primary BI components – DTS, Analysis Services OLAP, Analysis Services Data Mining – are substantially different and improved in SQL Server "Yukon." The SQL Server "Yukon" relational database contains several significant new features. Although the Microsoft Office query and portal tools are not part of SQL Server, the current releases will continue to work against SQL Server "Yukon." The Office tools' BI functionality will evolve on the Office product release cycle.

The SQL Server "Yukon" Business Intelligence toolset delivers end-to-end BI application integration:

- **Design:** BI Development "Workbench" is the first integrated development environment designed for the business intelligence system developer. Built on Visual Studio .Net, the BI Development "Workbench" delivers a rich, integrated, professional development platform for BI system developers. Debugging, source control, and script and code development are available for all components of the BI application.
- **Synthesize:** Data Transformation Services has been rewritten to perform complex data integration, transformation, and synthesis at high speed for very large data volumes. The BI Development "Workbench" makes building and debugging packages positively fun. DTS, Analysis Services, and Reporting Services work together to present a seamless view of data from heterogeneous sources.
- **Store:** SQL Server "Yukon" blurs the lines between relational and multidimensional databases. You can store data in the relational database, in the multidimensional database, or use the new Proactive Cache feature to get the best of both worlds.
- **Analyze:** Microsoft's Data Mining has always been easy to use. Now it's even better with the addition of important new algorithms, including Association Rules, Time Series, Regression Trees, Sequence Clustering, Neural Nets, and Naïve Bayes. Important new analytical capabilities have been added to Analysis Services cubes as well: Key Performance Indicator framework, MDX scripts, and other built-in advanced business analytics. The Reporting Services report delivery and management framework enables easy distribution of complex analytics to the widest possible audience.
- **Deliver:** Reporting Services extends the Microsoft Business Intelligence platform to reach the business user who needs to consume the analysis. Reporting Services is an enterprise managed reporting environment, embedded and managed via web services. Reports can be personalized and delivered in a variety of formats, with a range of interactivity and printing options. Complex analyses can reach a broad audience through the distribution of reports as a data source for downstream business intelligence. Microsoft's and partners' ad hoc query and analysis tools will continue to be a popular choice for accessing data in Analysis Services and relational databases.
- **Manage:** The SQL "Workbench" integrates the management of all SQL Server "Yukon" components. Business Intelligence practitioners will benefit from Microsoft's extension of the server "abilities" you expect from the relational engine – scalability, reliability, availability, programmability, and so on – to the full set of BI platform components.

A key goal of the SQL Server "Yukon" Business Intelligence components is to support the development and use of business intelligence in enterprises of all sizes, and to all employees – not just management and analysts, but also operational and external constituents. To this end, SQL Server "Yukon" is complete, integrated, easy to use, publishes data as web services, delivers great performance on commodity hardware, and contains scores of new features that you can use to develop innovative analytical applications.

Where do I start with "Yukon" Beta 1?

The first thing you'll notice upon installing SQL Server "Yukon" is that the installation experience is integrated. No longer do you need to run the installation program separately for some features such as Analysis Services. If a feature such as Reporting Services is not available for installation, your computer does not meet the requirements for that feature's installation. Review the readme for a complete discussion of feature prerequisites. On most properly configured machines, you should accept all defaults during installation, to install all the major features:

- SQL Server relational database engine
- DTS
- Analysis Services
- Reporting Services
- SQL "Workbench" (database management toolset)
- BI "Workbench" (BI application development toolset)

Reporting Services requires that IIS be installed and correctly configured. We strongly recommend that you take the time to perform these configuration and installation steps, as Reporting Services is an integral part of the "Yukon" Business Intelligence feature set.

Customers who are familiar with Analysis Services may be mystified by the lack of an Analysis Services metadata repository. In SQL Server 2000 the Analysis Services repository was shipped as a Microsoft Access database. Analysis Services "Yukon" does not contain a metadata repository. Instead, the Analysis Services database metadata information are stored as XML files and managed by Analysis Services. These XML files can be placed under source control, if desired.

We recommend that you use the BI "Workbench" to develop, and the SQL "Workbench" to operate and maintain BI database objects. You could design DTS packages and Analysis Services cubes and mining models in the SQL "Workbench," but the BI "Workbench" offers a better experience for designing and debugging BI applications.

For Beta 1, you will probably learn more by beginning with a new application rather than upgrading existing DTS packages or Analysis Services databases. You may find it useful to "recreate" an existing package or database, if you have one available. After you have become more familiar with the new tools, features, and concepts, try upgrading existing objects.

Many customers will use the SQL Server tools to develop a system with the now-familiar business intelligence structure of one or more source systems using DTS to feed a dimensional relational data warehouse, which in turn is used to populate the Analysis Services database. However, SQL Server "Yukon" provides many options to deviate from this generic design, by eliminating or virtualizing different components.

Relational data warehousing

The SQL Server "Yukon" relational database engine contains several features of great interest for the design and maintenance of data warehouse style applications. These features include:

- Table partitions enable fast data load and simplified maintenance for very large tables.
- Easy creation of a reporting server
- Transact-SQL improvements including new data types and new analytic functions
- Online index operations
- Fine grained backup/restore operations
- Fast file initialization

Reporting Server

A common technique for offloading relational operational reporting from the transaction database is to maintain a reporting server. The reporting server maintains an image of the transaction database with some latency, most often as of the previous day. The reporting server is used for most reporting and data warehouse extracts.

Microsoft SQL Server "Yukon" adds two new features that make it very easy to create and maintain a reporting server. The SQL Server reporting server can have much lower latency than daily. As well, the reporting server is designed to act as a standby system for the transaction system.

To create a reporting server, first create a database mirror, a new SQL Server "Yukon" feature that provides an instant standby system for high availability. Read the Books Online topic "Database Mirroring Concepts" for more information. The database mirror cannot be queried directly, which is where the second new feature comes into play.

Create a database view on the mirror. A database view is a read-only copy of a database at a point in time. A database view is not a full copy of the database; it is extremely space efficient. Multiple database views can exist, although maintaining a database view does have some impact on the transaction database upon which the database view is based. Read the Books Online topic "Understanding Database Views" for more information.

By creating a database view on a database mirror, you can easily create a standby server for high system availability, which serves double duty as a reporting server.

Table partitions

Partitioned tables and indexes have their data divided into horizontal units, so that groups of rows are mapped into individual partitions. Operations performed on the data, such as queries, are executed as if the entire table or index is a single entity.

Partitioning can:

- Improve table and index manageability.
- Improve query performance on multiple-CPU machines.

In a relational data warehouse, fact tables are the obvious candidate for table partitioning, and partitioning by date range is the most common partitioning strategy.

There are three steps to defining a partitioned table, as described in the Books Online topic *Creating Partitioned Tables and Indexes*:

1. Create a partition function to specify how a table that uses the function can be partitioned.
2. Create a partition scheme to specify the placement of a partition function's partitions on filegroups.
3. Create a table or index using the partition scheme.

Multiple tables can use the same partition scheme.

This paper discusses only Range partitioning of fact tables, and is not intended to be a complete discussion or tutorial for table partitioning. Interested readers should see the whitepaper titled *SQL Server "Yukon" Beta 1 Partitioned Tables and Indexes*.

The most common partitioning scheme will partition the fact table by date range, for example year, quarter, month, or even day. In most scenarios, date partitioning of the large fact table(s) will provide the manageability benefits. In order to get improved query performance, the Time dimension table should be partitioned using the same partitioning scheme.

- A partitioned table behaves like an unpartitioned table.
- Queries to the table are resolved correctly.
- Direct inserts, updates, and deletes on the table are automatically resolved to the correct partition(s).

Using table partitions for fast data loads

Most data warehouse applications struggle to load increasingly large volumes of data in a small – and shrinking – load window. The typical process begins with extracts from several source systems, followed by steps to clean, transform, synthesize, and rationalize the data across these systems. The data management application is constrained to complete the entire extract, transformation, and loading process within the load window. Usually, the system's business users have a strong requirement that the system minimize the amount of time the data warehouse is unavailable for their queries. The "write" step of the data management application, in which new data are inserted into the existing data warehouse, must be designed to occur quickly and with minimum user impact.

In order to load data very fast, the database recovery model must be Bulk Logged or Simple, and the table must either be empty or contain data but no indexes. If these conditions are met, a non-logged load is possible. In SQL Server 2000, before partitioned tables existed, these conditions are typically met only in the initial historical data warehouse load. Some customers with large data warehouses have built a quasi-partitioned structure by constructing a UNION ALL view over separate physical tables; these tables were populated each load cycle using a non-logged technique. This approach was not entirely satisfactory, and SQL Server "Yukon" partitioned tables provide superior functionality.

In SQL Server "Yukon", you cannot perform a non-logged load directly into a partition. However, you can load into a separate table that we will call the pseudo-partition. Under certain conditions, you can switch the pseudo-partition into the partitioned table as a metadata operation that occurs extremely quickly. This technique meets our two requirements of

- minimizing overall load time: the pseudo-partition load is performed without logging, and
- minimizing end user impact and ensuring data warehouse integrity: the pseudo-partitions can be loaded while users are querying the data warehouse. The data management application can wait until all fact tables are loaded and ready before performing the partition switch. The partition switch operation occurs very quickly, on the order of sub-second response.

In addition, the pseudo partition can be backed up as a separate table, improving system manageability.

Using table partitions for fast data deletes

Many data warehouses keep a sliding window of detailed data active in the data warehouse. For example, the fact table may contain data for three, five, or ten years. Periodically, the oldest data are removed from the table. The primary reasons for keeping data pruned are to improve query performance and minimize storage costs.

SQL Server "Yukon" partitions make it very easy to prune old data from a large partitioned fact table. Simply create an empty pseudo-partition as described above, and then switch it into the partitioned table. The partitioned table has an empty partition where it once had a populated partition; the pseudo-partition has data where once it was empty. The pseudo-partition can be backed up, truncated, or dropped, as appropriate.

Optionally, you may choose to redefine the partition function to merge all of the left-hand (empty) partitions together into one.

Transact-SQL Improvements

New data types

There are several important new types that will be useful for data warehousing:

- **UtcDateTime** is a datetime datatype that is aware of time zone. UtcDateTime will be used in source transaction databases that span global operations. UtcDateTime will be used in data warehouses as an attribute, and in the data staging process to correctly align by date and time.
- **Date** is the date part of datetime, with precision of a day, ranging from 1/1/0001 to 12/31/9999. Most data warehouses include a so-called "Time" or "Date" dimension with granularity to date. The Date datatype will be useful in the definition of that important dimension.

- **Time** is the time part of datetime, with precision of 100 NS, ranging from 0:00:00 to 23:59:59.9999999.
- **Varchar(max)**, nvarchar(max), and varbinary(max) hold up to 2 GB of data, and will be useful alternatives to text, ntext, and image datatypes. These extended character types may be useful to hold extended metadata and other descriptive information in a data warehouse.

New Analytic Functions

Several new analytic functions provide basic analytic capabilities within Transact-SQL. These functions will be useful in data warehouses that allow user queries into the relational database rather than exclusively through Analysis Services. Also, these complex calculations are commonly used during data staging to develop valuable data attributes.

ROW_NUMBER. Returns the sequential row number of the result set.

RANK. Returns the rank of rows in a result set. RANK would be identical to ROW_NUMBER on an ordered set, but for the treatment of rows that tie. All rows with the same ordered value receive the same rank. The next rank matches up with the ROW_NUMBER again. In other words, if there is a two-way tie for 1st place, rows one and two receive RANK=1, and row three receives RANK=3. No rows receive RANK=2.

DENSE_RANK. Returns the rank of rows in a result set. The DENSE_RANK function is similar to RANK, but squeezes out the gaps left by the RANK function. In the sample above, rows one and two receive RANK=1, and row three receives RANK=2.

NTILE. Divides an ordered set into the specified number of groups of approximately equal size.

These analytic functions are not available for SQL Server "Yukon" Beta 1.

PIVOT and UNPIVOT operators

The PIVOT operator allows you to generate a cross-tab report, by pivoting a result set along a break value in a query. For example, if the table contains data for "Actuals" and "Budgets" in separate rows, the PIVOT operator can be used to generate a cross-tab report with columns named [Actuals] and [Budgets].

Similarly, the UNPIVOT operator can be used to break one row into several. In this example, a rowset with [Actuals] and [Budgets] could be transformed into multiple rows tagged with these values.

In earlier versions of SQL Server it was possible to write complex Transact-SQL SELECT statements to rotate data. The PIVOT and UNPIVOT operators provide a straightforward mechanism for rotating data.

Recursive queries

There are several scenarios where a "recursive query" is extremely useful. New functionality in SQL Server "Yukon" makes recursive queries possible, if not exactly easy.

A recursive query is a query on a table with a self-join. The two most common examples are tables holding information about employees and their managers, and bill-of-materials tables. A self-joined table is illustrated in the AdventureWorks database, Employee table.

It has always been easy to query a self-joined table for direct relationships, for example for the count of employees who directly report to a manager. However, answering a question "How many employees are in a manager's organization?" is more difficult.

The SQL Server "Yukon" relational database feature that addresses this problem is called "Recursive common table expressions." The Appendix contains an example of a recursive query that answers the question defined above. More information is available in the Books Online topic "WITH <common_table_expression>".

Extract, Transformation, and Loading (ETL)

Data Transformation Services (DTS) is completely new for SQL Server "Yukon." DTS is a popular feature of SQL Server 2000, but DTS "Yukon" has been redesigned to be an enterprise ETL platform. DTS provides the breadth of features, and very high scale performance, necessary to build enterprise-class ETL applications. DTS is fully programmable, embeddable, and extensible – characteristics that make it an ideal ETL platform.

The DTS "Yukon" features are summarized below. For a more complete discussion of the applicability of DTS to the development of ETL systems, see the whitepaper *A Practical Introduction to What's New in "Yukon" Data Transformation Services*.

Package development

Yukon DTS features	Enterprise ETL development	ETL Platform
Use the **BI "Workbench" graphical user interface** to design DTS packages for data management applications. DTS packages are designed, developed, and debugged in the BI "Workbench," by dragging tasks from the toolbox, setting their properties, and connecting tasks with precedence constraints.	✓	
Use **SQL "Workbench" wizards** to develop simple DTS packages that perform common tasks such as Copy Database. (Not available for Beta 1.)	✓	
Software vendors will **embed DTS functionality** in their products, building wizards to generate custom packages as required.		✓
Separation of control flow from data flow. Most DTS packages contain multiple control flow tasks, and loops or sequences of tasks, which are laid out in the control flow pane. One control task, the Pipeline task, is the workhorse of the package, and has its own design surface for laying out data flows. The separation of control and data flow makes packages easier to read.	✓	

Yukon DTS features *(Continued)*	Enterprise ETL development	ETL Platform
Package variables are defined and visible. Variables are scoped, for example to a package, a loop, or a task.	✓	
Complex ETLM systems can be implemented by building package networks, with one package that calls other packages. Subpackages can elegantly re-use logic, variables, and context. There is less need to nest packages in this way than in DTS 2000.	✓	
The **package configuration framework** is an extensible system for customizing the way a package runs in different circumstances.	✓	
DTS packages are stored as XML in the file system or in SQL Server. DTS XML files can be managed under source control.	✓	✓
DTS 2000 package migration wizard will help migrate packages to DTS "Yukon," and provide warnings where upgrade is problematic.	✓	
DTS 2000 runtime is included with SQL Server "Yukon," to run DTS 2000 packages without upgrading them.	✓	
Packages' operations and results are **logged** in a variety of formats, to a wide variety of providers.	✓	✓
Event handler logic can be defined once and used many times.	✓	✓
Integration with WMI means that packages can respond to external events (such as completion of a file copy), or throw WMI events that other processes can consume. (Not available in Beta 1.)	✓	✓
Package restartability with checkpoints for failures, helps administrators manage complex packages moving large data volumes.	✓	

Control Flow

Yukon DTS features	Enterprise ETL development	ETL Platform
Precedence constraints: You can design a package to have control pass to different tasks on success, on failure, or on completion.	✓	
Looping tasks include For, ForEach, and Sequence loops. Package developers can easily perform a set of actions on all (or a set of) tables in a database, files in directory, or partitions in an Analysis Services cube.	✓	
Analysis Services integration is seamless, with control tasks that automatically execute Analysis Services DDL, process Analysis Services objects, or execute a data mining query. As discussed below, the DTS pipeline also integrates with Analysis Services.	✓	
VB.NET scripting is available with the Script Task. A second scripting task called the ActiveX Script Task is used primarily for backward compatibility to DTS 2000.	✓	
Communication tasks include: • Message queue • Send mail	✓	
Other control flow tasks include: • Bulk insert • Execute package • Execute process • Execute SQL • File system • FTP	✓	
Additional tasks can easily be developed using the DTS object model.		✓

Data Flow

Yukon DTS features	Enterprise ETL development	ETL Platform
Multiple sources, transformations, and destinations in the data flow pipeline. Data are read, combined, and manipulated, then written only when the transformations are complete. The need for multiple writes to staging tables is reduced or eliminated; transformation performance is greatly increased.	✓	
The DTS Pipeline task consumes data from **multiple, heterogenous data sources** and locales. The extensible Data Source architecture already supports data from flat files, OLEDB sources (including DB2 and Oracle), and raw files. Additional sources, including sources that will consume specifically structured data, are planned.	✓	✓
Additional data sources can easily be developed by Microsoft or its partners.		✓
Data from multiple sources can be joined with the **Join, Lookup, and Union** operators. These operations are performed in memory, without requiring a write operation to a database or file.	✓	
A data stream can be split using the **Conditional Split and Multicast** transforms. The compiler-like DTS engine determines which **data streams can be operated on in parallel**.	✓	
A wide variety of row-based data transformations are provided by the **Character Map, Copy Map, Data Conversion, and Derived Column** transforms. These operators are more like wizards than simple transforms, and provide the vast majority of required data transformations.	✓	

Yukon DTS features *(Continued)*	**Enterprise ETL development**	**ETL Platform**
Some data transformation tasks require comparing data from multiple rows. The **Sort and Aggregate** transforms perform these operations with extremely high performance, in the data flow, at a scale far beyond the reach of database aggregates.	✓	
Some data transformation tasks require complex logic, such as **Fuzzy Matching, Fuzzy Grouping, Time Dimension Generation, and Pivoting or Unpivoting**. Other common tasks, such as **Dimension Key Management**, require multiple steps. Special technologies and wizards make these complex operations available to all users.	✓	
Transformed data can be written to **heterogenous targets**, including SQL Server tables, OLEDB database tables, flat and raw files.	✓	✓
Transformed data can be **integrated with other components of the Microsoft BI solution**, including Analysis Services databases and data mining models.	✓	
Error flows from a transformation step can be managed in multiple ways: • In-process transformations can "fix" the data and resubmit to the main flow. • Error rows can be logged to tables or files for offline investigation and resubmission.	✓	✓
Additional transforms and destinations can easily be developed by Microsoft or its partners.		✓

Developing and Debugging

Yukon DTS features	Enterprise ETL development	ETL Platform
The package developer can define control flow **Breakpoints** at each control flow task. The breakpoint can be defined before, after, or at several points during the task's execution in the debugging process.	✓	
The package developer can attach a **Data Viewer** to each transform in the data flow. During debugging, the data viewer displays the contents of the transformed data stream at that point.	✓	
The BI "Workbench" is **hosted in Visual Studio**. Scripting and other programming tasks take advantage of that enterprise development environment.	✓	✓
Package deployment automatically bundles together all package components, including custom scripts and executables, for distribution to Test, Production, or other customer systems.		✓

DTS "Yukon" for the DTS 2000 Developer

Users of DTS 2000 have developed a set of tricks for performing complex operations. These tricks – particularly writing self-modifying packages – are not necessary with DTS "Yukon." Use the variables and configurations infrastructures to write dynamic packages; do not attempt to write self-modifying packages.

The elegant variables and configurations infrastructures also reduce the need to create complex systems of subpackages. With good design, a single package may be able to meet a variety of needs; for example, a single package could be reused with different configurations to load many dimension tables in a dimensional data warehouse. In DTS 2000 a complex network of DTS packages may include 50-100 packages; in DTS "Yukon" a complex network may include only ten packages.

Analysis Services

SQL Server 2000 Analysis Services consisted of two major, and complementary, pieces of functionality: On-Line Analytical Processing (OLAP) and Data Mining. These two components still exist in Analysis Services "Yukon" as the keystones of analytical applications.

The improvements in Analysis Services "Yukon" OLAP functionality can be broadly grouped into two categories:

- Enabling new kinds of analytical applications, either by adding completely new functionality, or by making it much easier for you to build complex functionality.
- Increasing the enterprise-readiness of analytical applications

New feature or improved functionality	Design and development	Management and operations
Unified Dimensional Model combines the best characteristics of relational and OLAP data models. The Unified Dimensional Model is discussed in greater detail below.	✓	
Proactive caching lets you operate low-latency applications with near-zero management cost. Proactive caching is discussed in greater detail below.	✓	✓
The **Key Performance Indicator (KPI)** framework provides a simple, server-defined mechanism for defining corporate measures. A KPI consists of expressions for value, goal, current status, and trend, and are displayed using simple graphics such as gauges and stoplights.	✓	
Translations provide a simple, centrally-managed mechanism for storing and presenting analytic data to users in their preferred languages. One analytic database can be presented in multiple languages.	✓	
MDX Scripts are the new mechanism for defining Calculated Members, Named Sets, and Cell Calculations. • MDX Script syntax is simplified and improved. MDX Scripts can be debugged step-by-step. • MDX Script calculations can be cached and persisted, which provide excellent query performance even with complex calculations. • MDX Script calculations can maintain real-time dynamic calculation behaviors. MDX Scripts are discussed in greater detail below.	✓	✓
Analysis Services **stored procedures** allow you to create external routines in common language runtime programming languages such as C++, VB or C. Stored procedures extend the capabilities provided by Analysis Services 2000 user defined functions (UDFs). Analysis Services stored procedures are discussed in greater detail below.	✓	

New feature or improved functionality *(Continued)*	**Design and development**	**Management and operations**
Enhancements to **data writeback** include a ten-fold performance improvement. The analytic application can write data back to an aggregate cell, and optionally perform allocation of the aggregate data to its underlying leaf data.	✓	✓
Built-in **business rules, tools, and wizards** make hard designs easy: • Semi-additive measures • Time Intelligence • Account intelligence • Financial Aggregations • Currency Conversion • Time dimension generation	✓	
Data Source Views provide a mechanism to both simplify and extend the relational database that underlies the analytic application. Data Source Views are discussed in greater detail below.	✓	
Analysis Services' **data definition language** is XML. The Analysis Services metadata repository is gone, replaced by XML files that are stored and managed by the Analysis Services server.	✓	✓
Web services: **XML for Analysis (XML/A)** is the native, standards-based protocol for communicating with the Analysis Services server. New kinds of applications are enabled and easy to develop – applications that integrate analytics with operations in real time. • With XML/A as the native protocol, Analysis Services clients will be zero-footprint, and each server is automatically a web service. • A light-footprint Win32 layer is available for backward compatibility with tools that work with Analysis Services 2000 on OLE DB for OLAP, ADOMD, and ADOMD.Net. Many customers and developers will continue to use the ADOMD.Net object model to build custom applications on Analysis Services.	✓	✓

New feature or improved functionality *(Continued)*	**Design and development**	**Management and operations**
Calculations are centralized on the server. Analysis Services "Yukon," by contrast with Analysis Services 2000, performs all calculations on the server. The advantages are significant: • Clients have a zero footprint; the client-side cache is eliminated. • Query performance for complex calculations is greatly improved. The cost of these improvements is minor query performance degradation for the very simplest queries, which in Analysis Services 2000 were resolved from the client cache.	✓	✓
The **development and management tools** (BI Developer "Workbench" and SQL "Workbench") are the first complete development environment for business intelligence applications. The new tools help you to capture and model all of your data, and enable fast application development.	✓	✓
Analysis Services "Yukon" has improved its **permissions model**. Different roles and permissions include: • Server Administrator • Database Administrator • Process object • View object structure (grantable by object) • Alter object structure		✓

New feature or improved functionality *(Continued)*	**Design and development**	**Management and operations**
Analysis Services "Yukon" contains over 150 **security design** changes. Improvements in the security model include: • Analysis Services is "secure by default" with multiple lines of defense. • Administrative permissions are fine-grained; with separable permissions for different database objects, and for making design changes versus processing. • Local cubes can be encrypted. • Analysis Services runs with the lowest possible level of permissions. • Client/server communications can be encrypted and signed to secure against packet sniffing, spoofing, tampering and repudiation • Encryption is enforceable on server, and the server can reject clients not using encryption		✓
The Analysis Services "Yukon" server will generate **server trace events** that can be monitored using tools such as SQL Server Profiler, which have long been available for SQL Server relational databases. • Audit access and use of the application. • Audit application and server events to improve server manageability. • Audit application errors, and work with Microsoft Support to more easily resolve problems.		✓
Improved performance for calculations comes from several features: • The server calculation cache is shared among users • Query optimizer will "rewrite" a query to an equivalent statement with improved performance. • Improved NonEmpty performance • Distinct Count measures are improved		✓

New feature or improved functionality *(Continued)*	**Design and development**	**Management and operations**
Analysis Services "Yukon" includes broad support for **middle tier architectures**. A light object model footprint delivers a scalable middle tier – scalable to thousands of concurrent users. Performance of deployments across a wide area network, although never recommended, is improved over SQL Server 2000.		✓
Analysis Services "Yukon" supports **unlimited dimension sizes**. Dimensions are no longer required to be cached in memory.	✓	✓
Analysis Services "Yukon" supports **parallel processing of partitions** within the standard management toolset.		✓
The **SQL "Workbench"** will be used to manage all SQL Server databases. It offers integrated management of the relational database with Analysis Services, including integrated tools for: • Server console management (replacing Enterprise Manager and Analysis Manager), • Query analysis (SQL and MDX), • Profiling events from the relational engine and Analysis Services, • The "Flight Recorder" and "Capture and Replay" features automatically capture server events, which can greatly help you (or Microsoft Services) diagnose a problem.		✓
A new object model, **Analysis Management Objects (AMO)**, replaces DSO. DSO is provided for backward compatibility, but AMO provides significant new features, most notably the ability to script an object's creation or modification from the management and development tools.	✓	✓

There are two main paths for building an analytic database:

- **Fully custom:** Starting from a source, usually a relational source, define dimensions, cube(s), KPIs, calculations, and data mining models. This path is appropriate for customers with an existing data warehouse or subject matter mart. In the first screen of the cube wizard, this choice is labeled "Use existing DB/Data Warehouse."
- **Customizable template:** Starting from a template, define and generate an entire application, including relational database(s), DTS packages, and an Analysis Services OLAP database. These components are designed and generated to work seamlessly together as a complete application. This approach is appropriate for customers who are installing a complete business intelligence solution from a template. In the first screen of the cube wizard, this choice is labeled "Design BI model without data source." For Beta 1, there are two templates available: Budget, and Sales and Inventory.

With either approach, the basic system design assumes the now-familiar business intelligence structure of one or more source systems feeding a dimensional relational data warehouse, which in turn is used to populate the Analysis Services database. However, SQL Server "Yukon" provides many options to deviate from this generic design, by eliminating or virtualizing different components. Some alternative systems are discussed in the section below on the Unified Dimensional Model.

Create a custom database from an existing source

The first method of creating an Analysis Services database is most familiar to users of SQL Server 2000. Start from a source database of any structure:

- A dimensional database structured as fact and dimension tables, or
- Any other database structure, including normalized transaction systems.

The ability to source from a normalized database is a significant departure from Analysis Services 2000, which required a dimensional structure, either star, snowflake, or flattened. This feature makes it easier for you to develop very low latency BI applications.

Many users' requirements can be met most simply and cost-effectively by building the Analysis Services database directly on the transaction database, without first building a formal data warehouse. If your data needs minimal transformation, cleansing, and integration before it is useful, consider using an Analysis Services database to supplement or replace existing relational reporting. You can provide the power and interactivity of Analysis Services and better manage the load on your transaction systems.

Although it's possible to build and maintain an Analysis Services database directly from transaction systems, many enterprise analytic requirements are best met by first building a relational data warehouse. Complex data integration and data change management problems are best solved through the classic data warehouse architecture, with the Analysis Services database acting as the query and analysis engine.

Data Sources and Data Source Views

The first step toward building a new analytic application is to create a new Analysis Services project in the BI "Workbench". Once the blank project is created, you should create a Data Source to connect to the source database, which may be in any supported relational database management system. For Beta 1, you are advised to start with SQL Server 2000 or SQL Server "Yukon" relational databases as sources.

The Data Source stores the information for connecting to the source data. Data Source Views contain information about the relevant subset of tables in a source database. This information is not limited to the tables' physical structures in the source database; you can add information such as relationships, friendly name for tables and columns, calculated columns, and named queries.

Data Source Views can be shared between BI projects and DTS projects. Data Source Views are always useful, but they are particularly valuable when:

- The source database contains thousands of tables, of which only a relatively small number are useful in any single BI application.
- The BI system developer does not have system administration privileges on the source database, and is not permitted to create physical views or modify the source database.
- The BI system developer needs to work in "offline" mode, disconnected from the source database. Design and development tasks occur against the Data Source View, which is decoupled from the source database.

The investment you make in setting up good names and relationships in the Data Source View, pays off in ease of development of the analytical application.

Create dimensions and cubes

After you have created the Data Source View, you can create a cube by right-clicking on the "Cubes" icon in the Solution Explorer pane, and choosing "New cube." You are offered the option of enabling IntelliCube© detection and suggestions. If you choose to use IntelliCube, you must decide whether to build a cube that's optimized for pivoting or for reporting. The IntelliCube technology examines the database and data cardinality relationships in the Data Source View, and intelligently characterizes tables as fact tables, dimension tables, or dimension-to-fact bridge tables that resolve a many-to-many relationship. For Beta 1, it makes little difference whether you choose to optimize cubes and dimensions for pivoting or reporting. The only difference is whether IntelliCube attempts to create hierarchical relationships among the attributes in a dimension. Since hierarchies are very easy to create and destroy, this is not a choice to spend much time worrying about.

It is instructive to hit the "Finish" button immediately after this initial screen of the Cube Wizard. An Analysis Services database, dimensions, hierarchies, attributes, and cubes are all defined. You could edit this design, but it is generally more effective to go through the wizard a bit more slowly and make intelligent choices along the way.

After you have experimented with the Cube Wizard for a little while, you will likely find that you prefer to use the Dimension Wizard – which you launch by right-clicking on "Dimensions" in the Solution Explorer pane – to create complex dimensions one at a time. Once the big dimensions, such as Product, Customer, and Time, have been carefully defined, launch the Cube Wizard and be sure to include these predefined dimensions where appropriate.

Build and deploy

The steps to date have simply created dimension and cube definitions and structures as XML files on your development machine. The BI "Workbench" and the Configuration Manager let you manage the process of building and deploying your project on a target server. By default, the "development" target server is your local server. You can create alternative configurations for deployment to different environments. Key properties of the project, such as the name of the target server and the data source connection strings, can vary between configurations.

To preview and test the cubes and dimensions during the development cycle, build and deploy the project to the designated target server by choosing Deploy from the main BI "Workbench" menu. Alternatively, hit F5, or choose Debug… Start from the main BI "Workbench" menu. One of several debugging and browsing tools is launched, depending on what you doing when you choose Deploy. Depending on this context, the Deploy process will launch the cube browser, the MDX script debugger, or the KPI browser.

You may want to view a prototype system after defining its dimensions, measures, and cubes. Process against a development database with a relatively small amount of data, and verify the data and structures behave as expected.

As part of a prototype, you will want to design some of the more complex components of the Analysis Services database, Key Performance Indicators, Actions, and Calculations. If your database will be used by different user communities who are interested in different views of the data, explore Perspectives and alternative security plans. If you plan to deploy the database internationally to users with different languages, you can present localized object names with the Translations feature. Finally, a prototype should evaluate alternative physical configurations, such as Partitions and different Proactive Caching options.

Once the Analysis Service database has been developed, the database objects are deployed to a final testing, staging, or production server. The output of the project, produced during the build stage, can be used as input to the Analysis Services deployment utility. This utility helps deploy and process the database.

Create a customizable database from a template

We have just described the basic steps of creating a custom Analysis Services database from a known source. This path through the Cube Wizard and Dimension Wizard is analogous to the standard method of creating an Analysis Services 2000 database.

The alternative method of creating a "Yukon" analytical application is to choose the option in the second screen of the Cube Wizard to "Design BI model without data source." This path through the wizard is analogous to the design experience of the SQL Server 2000 Accelerator for Business Intelligence. This design experience generates a complete customizable application from a template: rich dimensional structures and analytic features, optionally including a relational data warehouse and DTS packages. Templates may be supplied by Microsoft, integrators, or independent software vendors.

You could design identical Analysis Services databases by taking either of the two paths through the wizards – create from a source database or create from a template. The first option assumes you will create a completely custom system. Object names and structures are completely customizable, with the initial design driven from the names and structures in the source database. The template option also creates a customizable database, but the initial design is driven from an expert's subject area template.

Many users will combine the two approaches. A very common scenario will be to create most of an Analysis Services database from an existing source, but to generate the Time dimension from a template.

The Unified Dimensional Model

Analysis Services "Yukon" blurs the line between relational and multi-dimensional OLAP databases. OLAP databases have always offered tremendous advantages for analytic applications, primarily:

- Superb query performance,
- Analytical richness, and
- Ease of use by business analysts.

Until now, these advantages have come at a cost. OLAP databases, including Analysis Services 2000, have found it difficult to deliver:

- Complex schemas including many-to-many relationships,
- Detailed reporting on a wide set of attributes, and
- Low-latency data.

By combining the best aspects of traditional OLAP analysis and relational reporting, Analysis Services "Yukon" provides a single, unified dimensional model that covers both sets of needs. A set of cubes and dimensions defined in Yukon is referred to as a Unified Dimensional Model, or UDM. The advantages and flexibility of the UDM drives a sea change in design. In the past, the BI architect chose either relational or OLAP, weighing the benefits and costs of the alternative infrastructures. Now, the architect designs a Unified Dimensional Model, and then determines which point between those traditional extremes to place the logical design and physical configuration of the Analysis Services system.

Attribute-based dimensions

Analysis Services "Yukon" structures a cube around dimensional *attributes* rather than dimensional *hierarchies*. In Analysis Services 2000, dimensional designs are dominated by hierarchies such as {Year, Month, Day} or {Country, Region, City}. These hierarchies demanded strong data relationships between levels. Attributes, which were exposed as member properties and virtual dimensions, were second-class citizens. While it was possible to make attributes into physical dimensions, performance considerations discouraged widespread use of this technique. Users familiar with relational structures have been perplexed by the strong focus on hierarchies in OLAP databases.

The Analysis Services "Yukon" structure is more akin to a relational dimensional structure. A dimension contains many attributes, each of which can be used for slicing and filtering queries, and each of which can be combined into hierarchies regardless of data relationships.

Users who come from an OLAP background understand the value of strong hierarchies, wherein you can be certain that Cities roll up cleanly to Regions and Countries. These natural hierarchies still exist, and should be defined where appropriate: query performance benefits obtain from their use.

As an example, consider a Customer dimension. The relational source table has eight columns:

- CustomerKey
- CustomerName
- Age
- Gender
- Email
- City
- Region
- Country

The corresponding Analysis Services dimension should have seven attributes:

- Customer (integer key, CustomerName as the name)
- Age, Gender, Email, City, Region, Country

There is a natural hierarchy in the data, on {Country, Region, City, Customer}. For purposes of navigation, the application developer may choose to create a second hierarchy on {Age, Gender}. Business users don't see any difference between the way the two hierarchies behave, but the natural hierarchy benefits from indexing structures – hidden from the user – that understand the hierarchical relationships.

The most important advantages of the new dimensional structure are:

- Dimensions are not required to be loaded into memory. As a result, dimensions can be extremely large (hundreds of millions of members have been tested for Beta 1).
- Attribute hierarchies can be added and removed without reprocessing the dimension. The attribute hierarchy index structure is lightweight, and is calculated in the background while the cube remains available for querying.

- Duplicate dimensional information is eliminated; dimensions are smaller.
- Dimension processing performance is improved as the engine exploits opportunities for processing parallelism.

Types of Dimensions

Analysis Services 2000 has two types of dimensions: Regular hierarchical, and Parent-child. Analysis Services "Yukon" adds important new dimensional structures. Some of these structures have odd names, but these names are common in the BI literature.

- **Role Playing:** The dimension plays several roles depending on context. For example, the [Time] dimension may be reused for [Order Date] and [Ship Date]. In "Yukon," the role playing dimension is stored once and used multiple times. Disk space and processing times are minimized.
- **Fact:** A fact or "degenerate" dimension has a one-to-one relationship with the facts, such as a transaction number. The degenerate dimension is not used for analysis per se, but rather for identification – to locate a specific transaction, or to identify the transactions that make up an aggregate cell.
- **Reference:** The dimension is not directly related to the fact table, but instead is indirectly related through another dimension. The archetypal example is a [Geography] reference dimension that is related to both a [Customer] dimension and a [Sales Force] dimension. A reference dimension might be sourced from a data provider, and included in cubes without modifying the fact data.
- **Data Mining:** The data mining dimension provides support for dimensions that result from data mining models, including clusters, decision trees, and association rules.
- **Many to Many:** These dimensions are sometimes called multiple-valued dimensions. In most dimensions, a fact joins to one and only one dimension member. Many to many dimensions resolve to multiple dimension members. For example, a consumer banking customer has many accounts (Checking, Savings); an account has many customers (Mary Smith; John Smith). The [Customer] dimension has multiple members that relate to a single account transaction. "Yukon" many to many dimensions support complex analytics when dimensions are not directly related to fact table, and expand the dimensional model beyond the classic star schema.

Measure Groups and Perspectives

Analysis Services "Yukon" introduces Measure Groups and Perspectives to simplify analytical database design and deployment. In Analysis Services 2000, users were encouraged to build multiple physical cubes. Each cube corresponded to a specific dimensionality, and usually also to a specific relational fact table. Virtual cubes combined multiple fact tables in a manner that was transparent to the business user, but unnecessarily complex for the developer to define.

In "Yukon," the most common scenario will have a single physical cube that contains one or more Measure Groups. The fact data in a measure group has a specific grain, defined by the intersection of the dimension hierarchies. Queries are automatically directed to different measure groups as appropriate. On a physical level, partitions – analogous to Analysis Services 2000 partitions – are defined on Measure Groups.

A large application can present to the user a large number of dimensions, measure groups, and measures, and may be challenging to navigate. A Perspective, defined in the Perspectives tab of the Cube Editor, creates a subset "view" of a cube. To provide a degree of personalization, a security role can be associated with the set of perspectives applicable to that role.

We expect that most Analysis Services "Yukon" databases will contain a single cube with multiple measure groups and multiple perspectives.

Other important improvements to cube fact structures and query performance include:

- Measures can be nullable; in SQL 2000, "null" measures were treated as zero.
- Query performance for Distinct Count measures is improved by up to several orders of magnitude, for appropriately partitioned cubes.
- Access to alternative database management systems is provided by an extensible cartridge infrastructure. A cartridge for an RDBMS specifies how to optimize the SQL statements for relational querying and writing. Cartridges for additional relational systems can easily be added; a cartridge is implemented as an XSL file.

Calculations and Analytics

One of the greatest arguments for using an analytical server such as Analysis Services is the ability to define complex calculations centrally. Analysis Services has always delivered rich analytics, but some complex concepts have been difficult to implement.

One such concept is that of a semi-additive measure. Most common measures, such as [Sales], aggregate cleanly along all dimensions: [Total Sales] for all time is the sales for all products, for all customers, and for all time. A semi-additive measure, by contrast, may be additive in some dimensions but not in others. The most common scenario is a balance, such as the number of items in a warehouse. The aggregate balance for yesterday plus today is not, of course, the sum of yesterday's balance plus today's balance. Instead it's probably the ending balance, although in some scenarios it is the beginning balance. In Analysis Services 2000 you would have to define a complex MDX calculation to deliver the correct measure. With Analysis Services "Yukon," beginning and end balances are native aggregation types.

Distinct count measures are also much improved in "Yukon." A distinct count measure can now be defined on string data, and queries can be defined that will perform a Distinct Count over an arbitrary set. Analysis Services 2000 performed distinct counts only on the predefined hierarchical structure.

The "Time Intelligence" wizard will create a time calculations dimension containing calculations for this period versus last period, moving averages, and other common time calculation constructions.

MDX Scripts

MultiDimensional Expressions (MDX) is an extremely powerful language which was used to define Analysis Services 2000 calculations and security rules. But MDX is as complex as it is powerful. Analysis Services "Yukon" defines a new calculation model with MDX Scripts that use simplified constructs and syntax.

MDX is also the query language into Analysis Services systems. Query tools such as Excel Pivot Tables generate MDX queries based on users' "drag and drop" behaviors. This use of MDX is unrelated to MDX Scripts; MDX Scripts are used for server-defined objects such as calculated members and cell calculations, not for user queries.

When an Analysis Services "Yukon" cube is defined, it consists of structure only without data. The MDX Script is part of the cube structure. One default MDX Script command is always defined, to compute the default aggregations. The default MDX Script command consists of a single statement:

```
Calculate;
```

When the cube is fully processed but before this default MDX Script is applied, the cube contains leaf level data but no aggregations. When the single-statement default MDX Script is applied, the aggregations are computed and stored.

The MDX Script statement contains the following commands, separated by semicolons:

- Scope statements to limit statement scope
- Formula and value assignments
- Calculated member definitions
- Named set definitions

In the cube design user interface of the SQL "Workbench" or the BI Developer "Workbench," MDX Scripts – including calculated members and named sets – are constructed in the Calculations view. The MDX Script can be viewed in the default Calculations Form view, which provides guidance on syntax, or in the Calculations Script view, which presents the MDX Script as a set of commands terminated by semicolons. You can switch back and forth between the views, although the Form view currently requires correct syntax in the entire script in order to display.

There are several key features of MDX Scripts:

- Scripts follow a procedural model: statements are applied in order. The MDX Script developer no longer needs to worry about concepts such as pass order and infinite recursion.
- A calculation can be scoped: the SCOPE statement lets you define one or more calculations against a specific area of the cube. For example:

```
SCOPE ([Customers].[Country].[Country].[USA]);
   [Measures].[Sales] = 100;
END SCOPE;
```

- Scopes can be nested.
- A calculation can be cached: The CACHE keyword indicates that the result of the script calculation should be stored on disk, rather than computed at run-time. Cached calculations enable very high performance for queries on large cubes that contain a lot of complex calculations. When an input to a cached calculation changes, that calculation is dropped and rebuilt.
- MDX Scripts can be debugged. You can step through the MDX Script line by line, browsing the cube's results at each step.

Stored Procedures

Analysis Services "Yukon" introduces stored procedures to extend the capabilities provided by user defined functions (UDFs). A stored procedure can be written in any common language runtime programming language such as C++, VB or C. Stored procedures simplify database development and implementation by allowing common code to be developed once, stored in a single location, and reused in other stored procedures, calculations, and user queries.

There are three types of stored procedures:

- MDX function stored procedures are like any other MDX function, and provide a mechanism for easily extending the MDX language.
- Custom stored procedures perform tasks specific to an implementation such as cube processing or updating cells in a portion of the cube.
- Event handler stored procedures respond to user or system generated events.

A stored procedure can be used to perform any task that a client application can perform.

Key Performance Indicators

Analysis Services "Yukon" introduces a Key Performance Indicator (KPI) framework, for server-side definition of calculations to measure your business. These KPIs will be displayed in reports, portals, and dashboards, through data access APIs and Microsoft and 3rd party tools. For Beta 1, there are no client tools available that display KPIs.

Different commentators and vendors use the phrase "KPI" to refer to different concepts. For Microsoft SQL Server Analysis Services "Yukon," a KPI is precisely defined in four steps:

- Value to be measured: a physical measure such as Sales, a calculated measure such as Profit, or a calculation that is defined within the KPI,
- Goal for the value: a value (or MDX expression that resolves to a value) that defines the target for the measure,
- Status: an MDX expression to evaluate the current status of the value, as a normalized value in the range -1 (very bad) to +1 (very good),
- Trend: an MDX expression to evaluate the current trend of the value. Is the value getting better or worse relative to its goal?

An example of several KPIs displayed on a webpage follows:

Real Time Business Intelligence

Data warehouses and Business Intelligence applications have traditionally used "aged" or high latency data – data refreshed monthly, weekly, or daily. Traditionalists will assert that real time BI is an oxymoron – strategic decisions should not need data that is more timely than daily at best. What these commentators are missing is that business intelligence should be pervasive throughout the enterprise, and not just deployed to a few analysts and executives for purely strategic and tactical decision-making. Operational business intelligence requires low latency data.

Analysis Services "Yukon" provides new processing options for operational business intelligence. In Analysis Services 2000, cubes – no matter their storage mode or partitioning strategy – were processed with a "Pull" model. An Analysis Services process was launched that looked for new information in the source database, processed and optionally storeed the detailed data, and computed and stored the aggregations.

The Pull model is still supported in Analysis Services "Yukon," but it is joined by additional options that are particularly useful for low latency business intelligence:

- **Push data from the DTS pipeline** or a custom application. The data can flow directly into an Analysis Services partition from the DTS package pipeline, without intermediate storage. This scenario can be used to reduce the latency (and storage cost) of analytical data.
- **Manage the cube as a proactive cache,** without administrative intervention, to maintain a cache with prescribed latency and performance characteristics.

The query performance characteristics of an Analysis Services multidimensional store dominate relational storage. In short, queries perform best against multidimensional (MOLAP) storage. The downside is the latency: a multidimensional store is downstream from its relational source. The trick with the proactive caching technology is to maximize query performance while minimizing data latency and administrative costs.

The proactive caching features simplify the process of managing data obsolescence. If a transaction occurs on the source database, such as a new dimension member or new fact transaction, the existing "cache" is obsolete. Proactive caching provides a tunable mechanism for determining how often the multidimensional cache is rebuilt; for specifying how queries are answered while the cache is rebuilt; and for launching the process without any administrator intervention.

Proactive caching lets you set up the cube to automatically refresh its multidimensional cache when transactions occur. Although Analysis Services processes data extremely quickly, processing does take some time. Optionally, your proactive cache configuration can automatically redirect queries to the relational store if the multidimensional cache reprocessing is not complete.

When you are designing your proactive caching configuration, it's important to remember that proactive caching is set for each multidimensional partition. If a partition includes data for a short time period such as an hour, the cache refresh process can occur extremely quickly. The most complex proactive caching configurations depend on a notification from the relational database to Analysis Services that an update has occurred. The Microsoft SQL Server relational database supports such notifications.

The proactive cache parameters are:

- **Quiet period:** Amount of time the relational source must be free of transactions before the server starts to process the new information. The parameter is usually set to a value of less than ten seconds. Waiting for a quiet period protects against repeatedly dropping and rebuilding the cache if there are many sequential updates on the relational source.
- **Latency:** How long users are allowed to access data that is stale. If latency is set to zero, user queries are redirected to the relational source as soon as a notification is received. If the latency is set to 600 seconds, users access data that is no more than ten minutes old. A setting of -1 means users will continue to access stale data until the proactive cache processing has completed.
- **Silence override interval:** The maximum duration between a change notification and the start of the proactive cache processing. If the source database is updated continuously, this parameter will override the setting for "Quiet period."
- **Force rebuild interval:** This parameter is used to provide simple proactive caching functionality on systems where the source database system cannot supply an update notification. If the source data are in the SQL Server RDBMS, this parameter should be set to zero.

Data Mining

Overview

Microsoft SQL Server "Yukon" Data Mining is the business intelligence technology that helps you build complex analytical models, and integrate those models with your business operations. Data mining models answer questions such as

- What's the credit risk of this customer?
- What are my customers' characteristics?
- What products do people tend to buy together?
- How much product do I expect to sell next month?

A data mining application integrates the data mining model into the daily operation of your business. The goal of many data mining projects is to build an analytical application that your business users, partners, and customers can use every day, without having any notion of the complex computations that underlie the application. There are two main steps to reaching this goal: build the data mining model, and build the application. SQL Server "Yukon" Data Mining makes both of these steps easier than ever before.

Microsoft's goal for the data mining functionality in "Yukon" is to build tools that:

- Are easy to use
- Provide a complete set of features
- Are easily embedded in production applications
- Integrate closely with the other SQL Server BI technologies, and
- Extend the market for data mining applications.

Almost certainly, every reader of this whitepaper has "used" a data mining application before. If you have purchased a book or music online and received a recommendation that "other customers like you have enjoyed," or if your credit card company has asked you to verify a suspicious transaction, or your grocery store prints personalized coupons on your receipt, then you have used or benefited from a data mining application. To date, the development of such applications has been focused on the biggest problems at the biggest companies – those who can afford scarce analytical talent and high development costs that have traditionally been required to build a data mining application. Just as Microsoft's OLAP technologies have helped the OLAP market to grow, we are looking to extend data mining technologies to enterprises and corporate departments that have not been able to develop such applications in the past.

Use the SQL Server "Yukon" Data Mining tools to explore a set of data for patterns, and then optionally perform prediction on those patterns. That's all data mining is: exploration, pattern discovery, and pattern prediction.

Data mining algorithms

All data mining tools, including Microsoft SQL Server "Yukon" Analysis Services, use multiple algorithms. Analysis Services, of course, is extensible; third party ISVs can develop algorithms that snap in seamlessly to the Analysis Services data mining framework. Depending on the data and the goals, different algorithms are preferred, and each algorithm can be used for multiple problems.

Data mining tools are good at solving many kinds of problems. A rough categorization of business problems is included in the table below:

Analytical problem	Examples	Microsoft algorithms
Classification: Assign cases to predefined classes such as "Good" vs "Bad"	• Credit risk analysis • Churn analysis • Customer retention	• Decision Trees • Naïve Bayes • Neural Nets
Segmentation: Develop a taxonomy for grouping similar cases	• Customer profile analysis • Mailing campaign	• Clustering • Sequence Clustering
Association: Advanced counting for correlations	• Market basket analysis • Advanced data exploration	• Decision Trees • Association Rules
Time Series Forecasting: Predict the future	• Forecast sales • Predict stock prices	• Time Series
Prediction: Predict a value for a new case (such as a new customer) based on values for similar cases (such as existing customers)	• Quote insurance rates • Predict customer income • Predict temperature	• All
Deviation analysis: Discover how a case or segment differs from others	• Credit card fraud detection • Network infusion analysis	• All

SQL Server "Yukon" ships with the most popular data mining algorithms.

- **Microsoft Decision Trees** is often the starting point for data exploration. It is primarily a classification algorithm, and works well for predictive modeling of both discrete and continuous attributes. As the algorithm builds the model, it looks at how each input attribute in a dataset affects the result of the predicted attribute. The goal is to find a combination of input attributes and their states which allows you to predict the outcome of the predicted attribute.
- **Microsoft Naïve Bayes** quickly builds mining models that can be used for classification and prediction. It calculates probabilities for each possible state of the input attribute given each state of the predictable attribute. The algorithm supports only discrete (non-continuous) attributes and considers all the input attributes to be independent given the predictable attribute. Because the Naïve Bayes algorithm computes very quickly, it's a popular choice during the initial data exploration phase, and for classification and prediction problems.
- **Microsoft Clustering** uses iterative techniques to group records from the dataset into clusters containing similar characteristics. Using these clusters, you can explore the data to find relationships. You can also create predictions from the clustering model.

- **Microsoft Association** is based on the *a priori* algorithm, and provides an efficient method for finding N-way correlations within large datasets. The Association algorithm cycles through the transactions in the database to find which items are most likely to appear together in the transactions of a single user. Associated items are grouped together into itemsets, and rules are generated that can be used for prediction. Microsoft Association is used most often for market basket analysis. Any relational or OLAP analysis that is performing a lot of "distinct counting" is a good candidate for an Association analysis. The Microsoft Association algorithm is sensitive to the choice of algorithm parameters, so for small problems Microsoft Decision Trees may be a better algorithm to use for market basket analysis.
- **Microsoft Sequence Clustering** combines sequence analysis and clustering for data exploration and prediction. The sequence clustering model is sensitive to the order in which events occur. In addition, the clustering algorithm takes account of other attributes in its clustering of records, enabling you to develop a model that correlates sequential and non-sequential information. The Sequence Clustering algorithm will be used to perform clickstream analysis to analyze the flow of Web site traffic, to identify which pages are most closely related to the sale of a particular product, and to predict which pages are going to be visited next.
- **Microsoft Time Series** creates models that can be used to predict one or more continuous variables such as a stock price. The Time Series algorithm bases its prediction solely on the trends derived from the training data during the creation of the model. Microsoft Time Series uses an AutoRegression Trees technique, is very easy to use, and generates highly accurate models. There is an entire discipline of statistical analysis devoted to time series. Most other data mining products provide many techniques such as ARMA, ARIMA, and Box-Jenkins, among which the statistician must determine the model's best fit. Microsoft has chosen an approach which makes time series analysis accessible to a broad audience, with excellent and accurate results.
- **Microsoft Neural Net**, like Decision Trees and Naïve Bayes, is used primarily for data exploration, classification and prediction. Neural Net is an artificial intelligence technique that explores all possible data relationships. Because it is such a thorough technique, it is the slowest of the three classification algorithms. Microsoft Neural Net is not available for Beta 1.

Build the mining model

The model building, training, and testing process is the hardest part of creating the application. Actually developing the application is simple programming, as we discuss below. Before you start building a data mining model, you should have collected and cleaned your data, most likely in a data warehouse. "Yukon" Data Mining can access data from either the relational database or from Analysis Services cubes.

The best person to develop the data mining model is someone with skills in both business and technology. The developer of the model will benefit from a background in statistics, understand the key business problems the enterprise faces, have a deep curiosity about data and relationships, and also be able to work with the SQL Server "Yukon" tools to manipulate and store data. A member of an existing data warehouse team is most likely to meet these criteria.

As a novice to data mining, plan to spend several weeks exploring the data, tools, and alternative algorithms while you build a prototype model. Use a development server on which you have database administration privileges. The initial stages of model building are exploratory: you will probably want to restructure data and experiment with different approaches. You will almost certainly want to work with a small subset of data to start with, expanding the data set as you develop a clearer vision of the model's design. In the prototype phase, don't worry about building a "production ready" application. Use DTS or whatever tools are most comfortable to perform any necessary data manipulations. Keep a high-level log of any necessary transformations, but don't expect that anything you do will become part of the permanent application.

You should prepare two sets of data: one for developing the models, and one for testing the models' accuracy in order to choose the best model for your business problem. When you're thinking about how to subset the data, make sure you're not introducing bias. For example, choose every tenth customer, or discriminate based on the first character of the last name, or some other arbitrary attribute.

Developing a data mining model consists of making choices about:

- Input data set,
- Input fields,
- Data mining algorithm, and
- Parameters used during the computation of that algorithm.

If you don't know what kind of algorithm to use for your business problem, you should start with Decision Trees or Naïve Bayes to explore the data. If you don't know which attributes to include, pick them all. Use the dependency network view to provide a view that will help you simplify a complex model.

During the prototype development phase, you will want to build related models in order to evaluate the best algorithm and model. Use the Mining Accuracy chart to evaluate which model does the best job at prediction. You may also want to build related models in order to perform separate types of analysis on the same data. These models will process more quickly as related models than as independently defined models.

Once you have built and tested a prototype, you can build and test the real data mining model. If you need to transform the data before inputting it into the data mining engine, you should develop a production-ready operational process for doing so. In some cases you may choose to populate a mining model directly from the DTS pipeline. If the prototype was developed on a small subset of data, you will need to re-evaluate alternative models on the full set of training data.

Building the data mining application

An enterprise can get significant value from developing and exploring a data mining model within the BI "Workbench". You can browse the model, learn about the relationships in the data and your business, and use that information to drive strategic decisions. However, the greatest value will come from data mining applications that influence the daily operations of your company: for example, a data mining application that recommends products to customers, scores customers' credit risk, or places an order based on predicted inventory shortfall. To develop an operational data mining application you will need to step outside the SQL Server "Workbenches," and write some code – using Microsoft Visual Studio or the development environment of your choice.

Most enterprise customers will implement a customer-facing data mining application as a web-based or Win32 application, for example an ASP page. The data mining model has already been built, and the application performs a prediction for a customer based on something they have chosen or entered in a web commerce application. This could be a very simple application; the only unusual part is issuing the prediction query.

The data mining application developer does not have to be the same person who developed the mining model. The application developer should have classic development skills, but needs relatively little business or statistical knowledge.

Microsoft's data mining technologies make it very easy to build the automated data mining application. There are two steps:

- Develop the data mining prediction query, whose DMX syntax is defined in the OLE DB for Data Mining specification. Rather than writing the DMX by hand, click the Mining Model Prediction icon on the left bar of the BI "Workbench" editor. The graphical Prediction Query Builder tool helps you develop the prediction query.
- Use the prediction query in your data mining application. If your application simply uses DMX to perform a prediction, your project will need to include the ADO, ADO.Net, or ADOMD.Net class reference (ADOMD.Net is recommended for post-Beta 1 development). If you are building a more complex application – for example to display to your users the mining model viewers such as the Decision Tree Viewer – you will need to include the Microsoft.AnalysisServices and Microsoft.AnalysisServices.Viewers classes.

Some customers, primarily independent software vendors, will want to create an application that generates a data mining model. Such an application would substitute for developing a mining model in the BI "Workbench," perhaps for a specific domain such as web analytics. In this case your development project will need to include Microsoft.DataWarehouse.Interfaces to gain access to AMO (Analysis Management Objects).

Examples of DMX

The three steps of a data mining – creating a data mining model, training the model, and predicting behavior from the model – are all available from a simple, SQL-like programming language called DMX. Example syntax is shown below; a complete treatment of DMX is available in Books Online.

Create a data mining model:

```
CREATE MINING MODEL CreditRisk
(CustID           LONG KEY,
Gender            TEXT DISCRETE,
Income            LONG CONTINUOUS,
Profession  TEXT DISCRETE,
Risk        TEXT DISCRETE PREDICT)
USING Microsoft_Decision_Trees
```

Train a data mining model:

```
INSERT INTO CreditRisk
(CustId, Gender, Income, Profession, Risk)
SELECT CustomerID, Gender, Income, Profession, Risk
From Customers
```

Predict behavior from a data mining model:

```
SELECT NewCustomers.CustomerID, CreditRisk.Risk,
PredictProbability(CreditRisk)
FROM CreditRisk PREDICTION JOIN NewCustomers
ON CreditRisk.Gender=NewCustomer.Gender
AND CreditRisk.Income=NewCustomer.Income
AND CreditRisk.Profession=NewCustomer.Profession
```

Reporting Services

With the Microsoft® SQL Server™ "Yukon" release, Microsoft extends a major new component of its integrated Business Intelligence platform. SQL Server Reporting Services™ expands the Microsoft BI vision by making it easy to get the right information to the right people – in any business environment.

Reporting Services is a complete, server-based platform for creating, managing, and delivering traditional and interactive reports. It includes everything you need "out of the box" to create, distribute and manage reports. At the same time, the product's modular design and extensive Application Programming Interfaces (APIs) enable software developers, data providers, and enterprises to integrate reporting with legacy systems or third party applications.

Reporting Services ships with SQL Server "Yukon" and includes:

- A complete set of tools for creating, managing, and viewing reports
- An engine for hosting and processing reports
- An extensible architecture and open interfaces for embedding reports or integrating the solution in diverse IT environments.

Why Use Reporting Services?

No one questions the value of getting the right information to the right people, at the right time. For many businesses this is a challenge – people who need access to information may be distributed throughout and outside the traditional organization, with a wide range of skills and expertise.

Reporting Services helps makes it easy to create both traditional and interactive reports and deliver them to a wide range of people, using flexible subscription and delivery mechanisms. It also provides the security and manageability to handle complex and demanding business environments.

Reporting Services offers a unique combination of attributes:

- *A complete, server-based platform for reporting*: Reporting Services supports the full reporting lifecycle, from authoring to delivery and ongoing management of reports.
- *Flexible and extensible reporting*: Reporting Services supports both traditional and interactive reports of numerous formats, with extensible delivery options. It can be integrated easily into any environment or solution using open APIs and interfaces.
- *Scalability*: The product's modular, web-based design scales easily to support high volume environments. You could create a reporting server farm with multiple report servers accessing the same core reports, serving thousands of web-based clients.
- *Integration with Microsoft products and tools*: Reporting Services ships with SQL Server and integrates easily with familiar Microsoft tools, such as Office and SharePoint Portal Server, without requiring programming and customization.

Ways to Use Reporting Services

Because Reporting Services combines a single, complete reporting platform with a scalable and extensible architecture, it meets a wide variety of reporting needs.

- *Enterprise reporting:* Enterprises can use Reporting Services for internal reporting and BI applications. Many companies create data marts or warehouses to aggregate operational data. Using Reporting Services, corporate IT staff can design a variety of reports and deploy them to individuals throughout the enterprise, using a combination of email distribution and publishing on a corporate portal. For the enterprise, Reporting Service provides significant value as a comprehensive reporting solution integrated with the Microsoft BI platform.
- *Embedded reports:* Independent Software Vendors (ISVs) can use Reporting Services to deliver pre-defined reports as part of a packaged application that runs with Microsoft SQL Server. The customer's IT organization can access these reports as-is, or use Reporting Services to customize reports or create new ones for specific business needs. For the ISV, Reporting Services offers a simplified way to embed flexible, interactive reports in an application.
- *Web-based reporting for partners/customers:* Organizations can deploy traditional or interactive web-based reports to interact with customers or partners over extranets. Reporting Services isolates report consumers from the complexity of the underlying data sources, while providing personalization and interactivity.

Reporting Services Features

Reporting Services combines the benefits of a centrally-managed reporting system with the flexibility and on-demand nature of desktop and web-based applications. A complete reporting platform, Reporting Services supports the entire report lifecycle, from authoring through deployment.

Authoring reports

Reporting Services includes everything you need to start creating traditional or interactive reports, including a graphical report designer tool with report design wizards.

Report Authoring Feature	Details
Wide range of supported data sources	Microsoft SQL Server Microsoft Analysis Services Any OLE DB-compliant data source Any ODBC-compliant data source
Flexible authoring tools	Report Designer (uses Visual Studio .NET) XML-based Report Definition Language (RDL) Third party tools generating RDL
Flexible reporting formats	Freeform Table Matrix Charts Parameterized reports using run-time filtering Sorting and grouping Drillthrough Linked reports
Modular report execution	Rendering is a separate process from querying; the same report may be rendered in different formats. Execution can be scheduled or on-demand.

Managing reports

Reporting Services includes a web-based tool for managing reports, the Report Server Web Application. Administrators can use this interface to define role-based security for reports, schedule report execution and delivery, and track reporting history. Or, an enterprise or ISV can use the Reporting Services Web Services APIs to write customized management tools.

Because the report definitions, folders and resources are stored in a SQL Server database, you can use other tools such as SQL Server "Workbench" to manage metadata, or use third party applications that take advantage of published APIs.

Reporting Services implements a flexible, role-based security model to protect reports and reporting resources. This can be tailored to meet a wide variety of security needs. The product includes extensible interfaces for integrating other security models if desired.

Report Management Feature	Details
Report metadata	• Name • Description
Data Source management	• Connections • Credentials
Parameter Management	• Defaults • Prompts
Report scheduling	Integrated with SQL Server Agent
Execution properties	Live, cached or snapshot. A Reporting Services snapshot is a stored copy of the report dataset – the results of the report's source query when the report snapshot was run.
History of prior report executions	Cataloged list of snapshots retained to be used again as needed
Report security	• Users, Groups, and Roles
Report Server Web Application	Web-based management tool to: • Define security • Schedule report execution and delivery • Track reporting history
Flexible management APIs	Web Service API

Delivering reports

You can post reports to a portal, email them to users, or allow users to use the web-based report server to access reports from a folder hierarchy. Navigation, search, and subscription features help users locate and run the reports they need. Personalized subscriptions let them select the rendering format they prefer.

Report Delivery Feature	Details
Range of report rendering options	• Web formats (HTML) • Print formats (PDF, TIFF) • Data (Excel, XML, CSV) • Others through open API
Flexible delivery options	• Scheduled • Event-driven • Personalized subscriptions • Rendered report or link delivery • Data-driven subscriptions • Other applications through integration

Summary

Microsoft SQL Server "Yukon" is a complete BI platform that provides the infrastructure and server components to build:

- Large, complex, data warehouses that are easy to query and inexpensive to maintain;
- Small reporting and analysis systems that smaller enterprises, or departments of larger enterprises, can easily build and manage;
- Low latency systems that deliver analytic data to operational users;
- Closed loop analytic and data mining systems; and
- Embedded systems that extend the reach of business intelligence.

The familiar tools – the SQL Server relational database, DTS, and Analysis Services OLAP and Data Mining – are greatly improved. New functionality, such as Reporting Services and the development and management "Workbenches" , further extend the Microsoft BI platform. Each tool is innovative and designed to help you do more with less: build, deploy, and manage great business intelligence applications with less hardware, a smaller team, and faster and better than ever before.

Appendix A: Code samples

Recursive query sample

```
USE AdventureWorks
GO
/*
This query brings back a list of managers, and the count of employees who
report to them directly or indirectly)
*/
WITH reps_cte (emp, mgr, recursion_level)
AS
(
/*Get the initial list of employees.*/
SELECT EmployeeID, ManagerID, 0
FROM Employee AS E
/*Get a Union of the anchor and the recursive term.*/
UNION ALL
SELECT reps_cte.emp, E.ManagerID, recursion_level+1

FROM Employee E, reps_cte           -- Join with Employee
WHERE reps_cte.mgr=E.EmployeeID     -- This employee's manager
AND recursion_level<=20       -- up to 20 levels of mgmt
)           -- End of common table expression
/*Now query the recursive common table expression reps_cte*/
SELECT r.mgr,  E.[LastName]+', ' + E.[FirstName] AS MgrName, count(*)
CntEmployees
FROM reps_cte r INNER JOIN [Employee] E ON (r.mgr=E.EmployeeId)
GROUP BY mgr, E.[LastName]+', ' + E.[FirstName]
HAVING count(*) > 1                 -- Means they manage at least one
person
ORDER BY 3 DESC               -- Sort by count of employees
GO
```