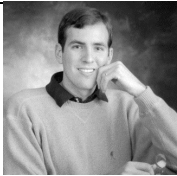


What's New In SQL Server 2000?

Brian Knight, Alltel

Introduction

Microsoft has added an enormous array of new features to this release of SQL Server. In an attempt to build up their market share, they have focused on performance, scalability, and reliability. They've also added a number of features to make developers' and DBAs' lives easier.



Brian Knight (bknight@sqlservercentral.com) has been working with SQL Server for about 5 years, during which he spent most of his time integrating SQL Server with web applications such as Cold Fusion and ASP. For the few past years he has been working with Alltel in Jacksonville, Florida, performance tuning enterprise servers and aiding in the creation of financial applications for the Web. He has written several columns with SQL Server Magazine, co-runs the SQL Server area on SWYNK.COM, and runs SQLServerCentral.com.

Performance

One of the best ways you can scale a server is to simply give it the ability to complete requests faster. Performance was a strong goal in the production of SQL Server 2000 (SS2K). TPC, an independent organization which has a benchmarking standard (TPC-C), placed SS2K as the second best performing database, below IBM's DB2. In December 1999, SQL Server 7.0 posted a modest a TPC-C benchmark result of 40,697 transactions/minute. In July this year, SS2K posted a TPC-C benchmark result of 262,243 transactions/minute, with a cost of only \$20.24/transaction, about a \$4 increase. While SQL Server's performance in the TPC-C has been nearly doubled, it remains one of the lowest cost/performance DBMS in the market.

Microsoft has enhanced a number of items to speed up SS2K's system. For instance, indexed views allow you to "materialize" a view. You essentially create a physical version of a view that is automatically persisted. This view can contain any other tables within the same database. This is ideal for a month-end type scenario where you need to report on profitability or where you need to load a data warehouse. The view can also be updated using a series of `Instead of` triggers, which we will explore later.

Although, on the surface, this is a nice feature, the way views are implemented means you must be careful when using them. Since an indexed view is "materialized" every time data changes in the table, it must also be applied to the view, causing added I/O. It is best to use indexed views where data is static, like in a data warehouse. Some of the restrictions on this feature are:

- `SCHEMABINDING` must be turned on
- You cannot reference other views
- All tables in the view must be in the same DB and have the same owner
- You must use two-part names for referenced tables

Many SQL Server actions that weren't using parallel CPUs before are now in this release. Maintenance procedures called DBCCs (Database Consistency Checker) now use multiple processors. DBCCs are occasionally still used to check databases for corruption if hardware fails or power outages occur – for large databases these procedures can take hours. This provides an almost linear performance increase, based on the number of processors installed. Keep in mind that there is also overhead on the operating system when you deal with more than one processor.

Index reorganizations can be done online now without potentially locking out other users. When you do this, you reorganize physical data pages based on logical order. Essentially, you defragment your database and make it run much more efficiently. This is similar to the performance gain you experience when fragmenting your hard drive. After you've rebuilt the indexes, SQL Server doesn't have to use as many I/O cycles to find the data since it's all in the same area of your hard drive.

One of the frustrations of SQL Server 7.0 is how long it takes to shrink a database and log after you issue the command. Logs were previously shrunk incrementally as data pages were freed. Log shrinking has been modified in SS2K. When you issue a shrink command in the latest release, data is moved rapidly to a temporary space and then back to its permanent location. Rather than freeing space as it became available, it is now instantaneous.

The text in row feature enables you to store small amounts of text on the same page that the row's data is stored. With this feature turned off, the text is stored off the row's page and only a pointer to the text page is stored in the row. You can turn on this feature at the table level by using the following syntax:

```
sp_tableoption N'TableName', 'text in row', 'ON'
```

Optionally, you can set a maximum number of bytes that can be stored in the table's row, from 24 to 7000 bytes. The default is 256 bytes if the text in row option is on. You can set the option by using the following syntax:

```
sp_tableoption N'TableName', 'text in row', '1024'
```

Merry-go-round scans are a performance benefit which has been added in SS2K that allow your query scans to "piggy-back" each other. As one table begins a scan another similar query can ride another query's scan until it finishes. It then completes the pieces of the scan that it missed.

Scaling

There are two ways to scale: up and out. Scaling up is when you add more processing power and RAM to your system. Scaling out is when you spread the I/O over a number of servers. Scaling up can be quite expensive and presents the dilemma of a single point of failure. Scaling out presents a scenario which is harder to manage, where you have to perform the same action on multiple servers. Scaling-out can become quite costly as you add more servers and human resources to manage the servers. However, scaling out is getting easier with Windows 2000. The best way is to do a combination of scaling up and out.

In the new release, the gap between Enterprise and Standard Editions has widened. There has also been a shuffle in names with the Desktop Edition being renamed the Personal Edition. SS2K focuses on performance and reliability improvements. To take full advantage of these improvements however, you may find that the Standard Edition just won't cut it.

Additional steps have been taken to guarantee that SS2K is more robust. The Enterprise Edition of SQL Server, in conjunction with Windows 2000 Data Center, can handle up to 32 CPUs and 64 GB of RAM. The Standard Edition can be used with 4 processors and 2 GB of RAM. Lastly, the Personal Edition can work with Windows 98, NT, and 2000, and is optimized for less than 5 concurrent users. MSDE has a new name as well. The light and free edition of SS2K is now called SQL Server 2000 Desktop Engine.

There is also an addition to the SQL Server family in this release. There's now a Windows CE Edition that will be programmatically compatible with SS2K but boasts a tiny footprint (about 700K). Although some functionality has been cut, some facets of replication including merge replication have been kept. SQL Server CE only supports Unicode data types such as `ntext`, `nvarchar`, and `nchar`. Smaller data types, such as `smalldatetime` and `smallmoney`, will be converted to their larger sister data types, such as `datetime` and `money`, respectively. The CE Edition will be released later this year. A beta version was made available in June 2000.

Other improvements include better caching for parameters and cursors. Parameter caching has resulted in a 6% boost in performance out of SAP SD workloads in Microsoft labs. Metadata is now being cached as well, which has resulted in a 10% reduction in query cost when selecting 50 columns or more. The Asynchronous I/O option previously introduced in version 7.0 has now been taken out in favor of a dynamic, more scalable configuration.

For example, you could segregate your customer table horizontally, keeping the northern clients on Server A and the southern clients on Server B. The results may be combined in a view using the `UNION ALL` clause so the two servers look like one. This method is nothing new to SS2K, but what is new is its ability to be updated and inserted into. This new feature is called Distributed Partitioned Views (DPVs). DBMSs like Oracle have had this ability for years and Enterprise DBAs have been calling out for it for just about as long. Now this feature is

available in SQL Server, but only in the Enterprise Edition. Some of the rules that apply to DPVs are that:

- They must have a uniform primary key (State, Country, Year, for example)
- They use check constraints
- The constraint must use part of the primary key
- They are built on OLE DB
- The Key T-SQL syntax is UNION ALL

Another concept that goes along with DPVs is the idea of federated databases. Federated databases allow a DPV to be broken onto several servers. The query optimizer has been improved to scan through OLE DB the check constraints of each server as a query is issued. It only scans servers that hold data that the query requested.

To create a DPV, first create a table on each server with the check constraints to separate the data.

```
-- ServerA:
CREATE TABLE Customer_1
  (CustomerID  INTEGER PRIMARY KEY
   CHECK (CustomerID BETWEEN 1 AND 499999))

-- On ServerB:
CREATE TABLE Customer_2
  (CustomerID  INTEGER PRIMARY KEY
   CHECK (CustomerID BETWEEN 500000 AND 999999))

Next create a view on each server as shown below.

--On Server A
CREATE VIEW EnterpriseCustomers AS
  SELECT * FROM Database.Owner.Customers_1
UNION ALL
  SELECT * FROM ServerB.Database.Owner.Customers_2
```

On Server B:

```
CREATE VIEW EnterpriseCustomers AS
  SELECT * FROM Database.Owner.Customers_2
UNION ALL
  SELECT * FROM ServerA.Database.Owner.Customers_1
```

DPVs are just the down payment in SS2K. A later release that is still in the planning stages (called "Yukon") hopes to allow you to manage all the servers as if they were one server, reducing the need for dual administration.

Log shipping is now also built into the SS2K Enterprise Edition. Previously, you had to load the Resource Kit to have this ability and it was quite difficult to configure. Log shipping allows you to capture a set amount of the transaction log and apply it to a standby server. This option is perfect for those who want to have a warm standby server in case of system failure. It is not made for high-availability type systems. In such instances, you would want to use a clustering environment.

Security

SS2K is Kerberos aware. This means it is compatible with Windows 2000 security and its Active Directory structure. SQL Server can encrypt data with SSL before transmitting the data. It now can be configured to have C2-level security, although this feature is not by default. C2-level security is a designation given by the National Security Agency (NSA) for applications that follow strict security standards. In SQL Server, this means that all access to objects will be logged, whether successful or unsuccessful. To enable this option, use the syntax below in Query Analyzer:

```
sp_configure "c2 audit mode", 1
go
reconfigure
go
```

You must be a member of the sysadmin role to turn on auditing and show advanced options must be turned on (also through `sp_configure`). The auditing process is done through SQL Profiler, which has the ability to roll over logs as they reach 200 MB. This process is repeated until the SQL Server service is stopped. As you can imagine, for busy systems hard drive space could become a premium as the logs begin to fill. If you run out of hard drive space, you can start the instance with the `-F` flag to bypass C2 auditing and free up space. If a hard drive fills or a disk quota for a directory is met, the SQL Server instance is stopped.

Auditing objects does cause a significant performance impact. Only turn on this type of auditing if needed.

Multiple Instances

SQL Server now has the ability to have multiple instances on one server. Previously, in SQL Server 7.0, you had to have the SQL Server name the same as the NT server name. This architecture didn't allow you to have multiple instances of SQL Server running on the same machine. Microsoft has dropped this requirement in SS2K which means you can now have SQL Server 6.5 and 2000 live on the same machine, for instance. This particular combination is not supported, however. Running versions 7.0 and 2000 on the same machine is supported. In this latter case, SQL Server 7.0 would be your default instance. A sample instance name looks like `BKNIGHT\BEDROCK` (where `BKNIGHT` is the server name and `BEDROCK` is the instance name). Keep in mind that, as you add instances, each instance is a completely new install of SQL Server and uses equal amounts of RAM and CPU.

Performance Monitor and Event Viewer are now instance aware. Instances are ideal for development environments, or for application hosting companies who need individual billing based on CPU. Individual billing can be done through logging average CPU in Performance Monitor. There is always one default instance plus up to 16 additional instances. Note that you should only run two versions when really necessary – dedicated servers are almost always the way to go.

Programmatically, you can find what instance you're currently on with the following T-SQL:

```
SELECT ServerProperty('InstanceName')
```

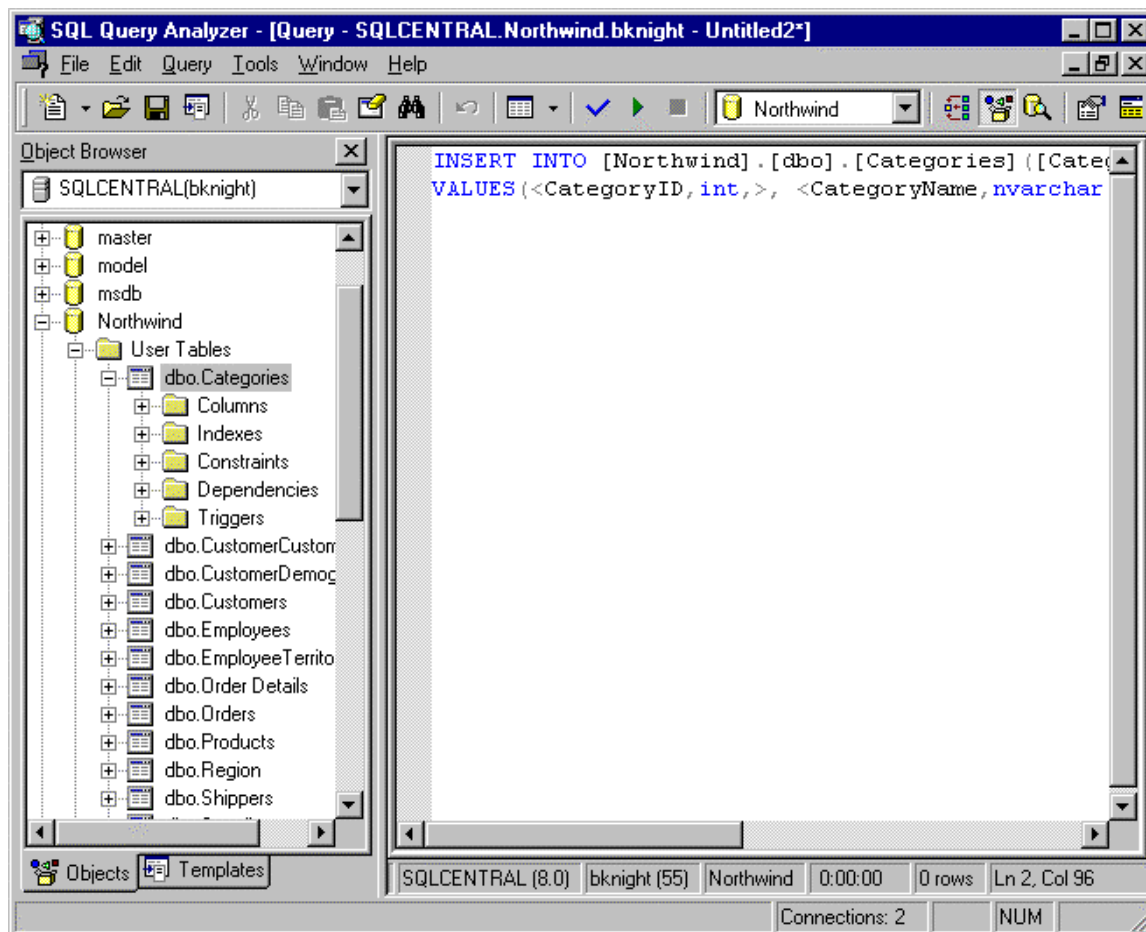
Tool Enhancements

All the tools, including Data Transformation Services (DTS), have been improved in this release of SQL Server. The focus on the tool enhancements was to allow a DBA to perform a task with

far fewer steps, as well as allow programmers to easily create queries and debug them after they're created.

Some of the Query Analyzer enhancements include:

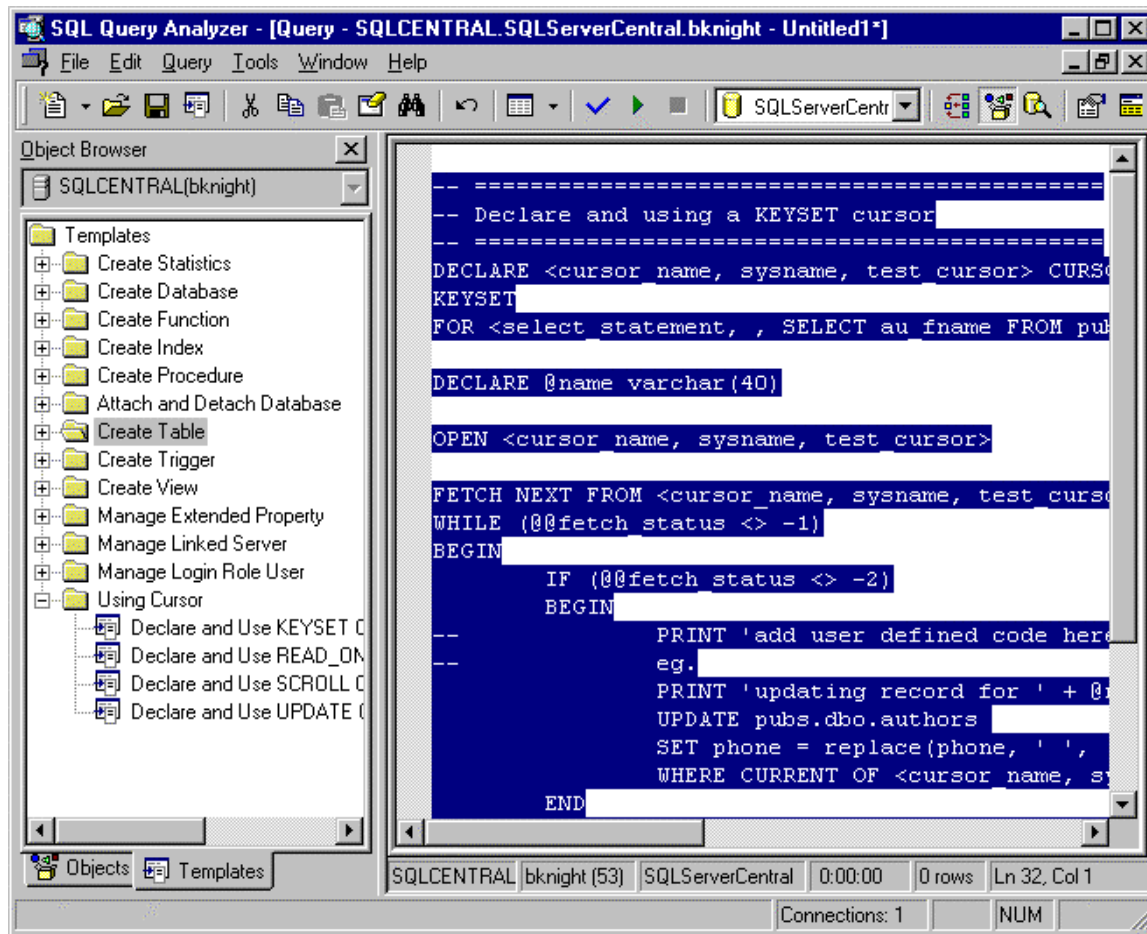
- Database object/element browser and search
- Query templates that can be customized
- Improved tracing and Showplans
- Stored Procedure debugger
- Drag-and-drop queries
- Multiple result sets in a grid
- Bookmark your code
- Shortcut queries



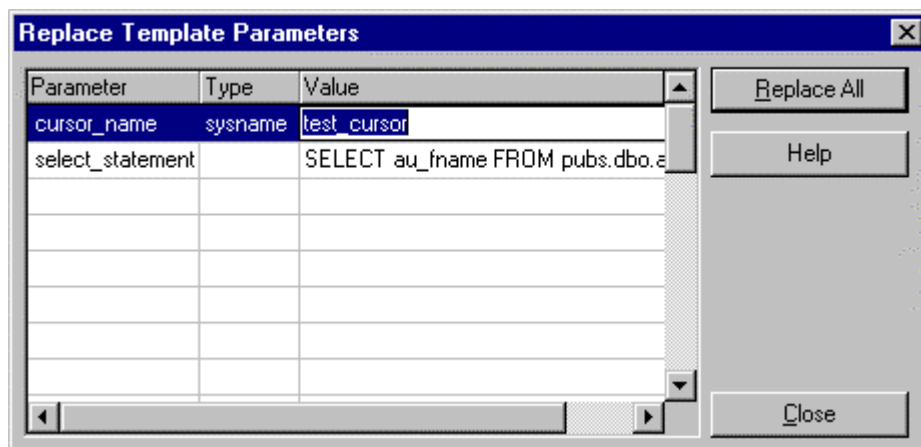
You can enable the drag-and-drop query and template features with your *F8* key. After they're enabled, drill down into the table you'd like to perform an action on. Click your right mouse button on any table and select any database action to perform on the table. Highlight your newly created query, and select *Replace Template Parameters* under the *Edit* menu. Here you can change the template that is produced by Query Analyzer to use your own values.

Templates hold the boilerplate code necessary to quickly create queries. It allows you to drag-and-drop templates onto the Query Analyzer pane and edit the parameters. The first step to creating your own custom template is to create a directory for your templates. By default, the templates are located in the *C:\Program Files\Microsoft SQL*

Server\80\Tools\Templates\SQL Query Analyzer folder. Anything you place in that directory will show up in the template tree on your next Query Analyzer load.



Parameters are marked with "<" and ">", for example, <cursor_name, sysname, test_cursor>. In this example, cursor_name is the name of the parameter, sysname is the type of value, and test_cursor is the default value. You can highlight your query and select Edit Template Parameters to change the parameters to the actual values as shown below.



Another tool enhancement is the Copy Database Wizard (CDW). CDW gives you the ability to copy or move database files and their accompanying logs to a new server. For a detailed look at this new feature, please see my "Fundamentals of DTS" presentation.

User Defined Functions

User Defined Functions (UDF) can be scalar-valued or inline table-valued. Scalar-valued functions allow you to make a calculation based on a received value. An inline table-valued function returns a table. Both types are written in T-SQL. Future releases of SQL Server (Yukon) will offer the ability to write such functions in VBScript or JavaScript.

The following in-line function code returns a "parameterized view" that you can treat as a view but also something that you can pass parameters into.

```
create function collaborator_inline (@au_id char(11))
returns table as
RETURN (
    select l.au_id as "author",
           l.title_id as "title",
           r.au_id as "co-author"
    from titleauthor l inner join titleauthor r
    on l.title_id = r.title_id AND l.au_id <> r.au_id
    where l.au_id = COALESCE(@au_id, l.au_id)
    -- order by l.title_id, l.au_id)
go
select * from dbo.collaborator_inline('724-80-9391')
select * from dbo.collaborator_inline(NULL)
```

"Instead of" Triggers

"Instead of" triggers fire instead of the normal triggering action. They can be placed on views or tables. The primary benefit to instead of triggers is that they allow you to insert or update a view that subsequently inserts or updates into any number of tables. The code below creates a view of two servers using the UNION ALL syntax, and then shows you how to insert into the view.

```
CREATE VIEW AllOrders AS
    Select OrderID, Amount, CustomerID, Region
    FROM OrdersEast
UNION ALL
    Select OrderID, Amount, CustomerID, Region
    FROM OrdersWest
----
CREATE TRIGGER IO_Trig_INS_AllOrders ON AllOrders
INSTEAD OF INSERT AS
BEGIN
    INSERT INTO OrdersEast
    Select OrderID, Amount, CustomerID, Region
    FROM inserted WHERE Region = 'East'
    INSERT INTO OrdersWest
    Select OrderID, Amount, CustomerID, Region
    FROM inserted WHERE Region = 'West'
END
```

Cascading DRI

Cascading Declarative Referential Integrity (DRI) is a long overdue feature which will allow you to create a trigger on a table to cascade updates and deletes to all Foreign Key tables from the Primary Key table. The following code will create two tables. If the region name is updated in the region table, it will also cascade that update to the employee table. The NO ACTION keyword allows you to keep the original constraint behavior.

```

CREATE TABLE region (
    Region_Name Varchar(50) NOT NULL PRIMARY KEY)

CREATE TABLE employee (
    EmployeeID int,
    Territory Varchar(50) NOT NULL REFERENCES region
    ON UPDATE CASCADE
    ON DELETE NO ACTION

```

One restriction on this is that a cascading trigger cannot be placed on a table's column that has a timestamp field as a Primary or Foreign Key.

XML

The new XML feature in SS2K allows you to output data into XML easily. You are available to update data through XML UpdateGrams as well, which is available through a web release. T-SQL has been extended to allow you to select data directly too. See the "Using XML and SQL Server 2000 Integration" presentation by Rob Vieira for more details on this feature.

SQL Server XML has support for the following types of data:

- Structured
- Semi-structured
- Unstructured

Other Relational Engine Improvements

New Data Types

Three new data types have been added to this release of SQL Server.

The `bigint` data type is like an `int` (4 byte integer) but stores 8 bytes of data. Only a few of the functions have been changed to support this new data type. One such function is `COUNT_BIG`, which performs the same action as `COUNT`, but returns a `bigint`. Only use the `bigint` data type in cases where you feel you may exceed the 4 byte restriction. `bigint` holds integers between `+ - 9,223,372,036,854,775,807`.

The `sql_variant` data type is a flexible data type that can hold any type of data except `timestamp`, `text`, `ntext`, and `image` data types. It operates much like the `variant` data type in Visual Basic. It is especially handy when you're trying to store metadata in a column where you don't know that type of data you'll be receiving.

The `table` data type is used to store result sets temporarily for later processing. You can create them while declaring a variable in a query or a stored procedure. Later in the procedure or query, you can select from the variable as if it were a table. The `table` data type is a perfect, easy substitute for temp tables. When comparing the execution plans between the `table` data type and temp tables, the `table` data type skips the step of creating a table and its table scan takes much less time. `table` variables in stored procedures use less recompilations than temp tables also. Below is an example of how to use the data type.

```

DECLARE @VariableName TABLE
    (columna int PRIMARY KEY,
     columnb varchar(20))

INSERT INTO @VariableName VALUES (1, '123 Main St')

```

```
INSERT INTO @VariableName VALUES (2, '55 Riverside Av')

SELECT * FROM @VariableName
GO
```

Extended Properties

These allow you to store metadata for database objects like views or columns – you can document your data model as you create your data structure. Extended properties are stored as `sql_variants` and can be up to 7500 characters. Applications can read these properties to display a caption about the field they are entering.

Column Level Collations

A great feature for international companies. Collation is the combination of language and sort order. In SQL Server 7.0, you could only define the collation at the server level and, once it was set, you could not change it without rebuilding the master database. A limitation resulting from this was that you couldn't backup a database from one collation and restore it on a different collation server. Column level collation allows you to set at the database or column level collation. It is ideal for those who only want the column name `firstname` to be represented in accent insensitive sort order.

Summary

In the latest release of SQL Server a number of primary goals were addressed by Microsoft: scalability, usability, and performance. You may scale up with added processor and RAM support, and have the ability to scale out to multiple servers. Performance has been enhanced mainly by using all the processors a server has. Finally, the tools in SQL Server 2000 have improved dramatically to provide a programmer an easier way to write queries in Query Analyzer and transform data through DTS. Through these changes, Microsoft aim to gain access to the largest segment of the database market.

SQL Server 2000 Resources

Web

<http://www.microsoft.com/sql>
<http://msdn.microsoft.com/sqlserver/>
<http://microsoft.com/technet/sql/>
<http://www.swynk.com>

Newsgroups

microsoft.public.sqlserver.server
microsoft.public.sqlserver.tools