Making the Most of SQL Server 2005: Developing World Class Database Applications

Brian A. Randell
Senior Consultant
http://www.mcwtech.com

Kimberly L. Tripp
President and Founder
http://www.SQLskills.com

About Brian

- Senior Consultant with MCW Technologies, LLC
- Visual Basic MVP
- Manager and co-author of Microsoft SQL Server 2005 Hands-on Labs
- Co-author of Effective Visual Basic
- Contact me via my blog or e-mail:
  - http://www.mcwtech.com/cs/blogs/brianr/
  - brianr@mcwtech.com

About Kimberly

- Independent Consultant/Trainer/Speaker/Writer
- Founder, SYSolutions, Inc. www.SQLskills.com
  - e-mail: Kimberly@SQLskills.com
  - blog: http://www.SQLskills.com/blogs/Kimberly
- Microsoft Regional Director http://msdn.microsoft.com/isv/rd
- SQL Server MVP http://mvp.support.microsoft.com/
- Author for some SQL Server 2005 Whitepapers on MSDN (links from www.SQLskills.com)
- Coauthor MSPress: SQL Server 2000 High Availability, Presenter/Technical Manager for SQL Server 2000 High Availability DVD
- Writer/Editor for SQL Magazine www.sqlmag.com

Agenda

- Installation
- Optimal Database
- Database Design
- Developing on SQL Server 2005
- Caching
- Tuning, Monitoring, and Diagnosing System Problems

Installation

- Which version has the features you need?
- Server Installation
- Database Installation

SQL Server 2000
A Major Release

- XML support
  - SELECT … FOR XML
  - OpenXML
  - XML Views
  - XML Updategrams
  - XML View Mapper
  - XML Bulk Load
- URL and HTTP db access
- HTTP access to cubes
- Multi-instance support
- Integrated Data Mining
- Full-Text Search in formatted docs
- English Query for the Web
- C2 security rating (NSA)
- Installation disk imaging
- Active Directory integration
- Self-management and tuning
- Distributed Partitioned Views
- Log Shipping
- Parallel CREATE INDEX
- Parallel scan
- Parallel DBCC
- Failover clustering
- Failover cluster management
- 32 CPU SMP system support
- 64 GB RAM support
- VIA SAN support
- Indexed views
- ROLAP dimension storage
- Distributed Partitioned Cubes
- Online index reorganization
- Differential backup
- User-defined functions
- Server-less snapshot backup
- SQL Query Analyzer debugger
- New data types
- Column-level collations
- Virtual Cube Editor
- Linked cubes
- MDX Builder
- Dimensions
- Security in Analysis Services
- OLAP Actions
- Custom rollups
- Cascading referential integrity and actions
- INSTEAD OF triggers
- Indexes on computed columns
- Queued replication
- DTS enhancements
- Copy Database Wizard

SQL Server 2005

.NET Framework
- Common Language Runtime Integration
- User-defined Aggregates
- User-defined Data Types
- User-defined Functions
- SQL Server .NET Data Provider
- Extended Triggers

Data Types
- New XML Datatype
- n/varchar(max)

SQL Server Engine
- New Message Service Broker
- HTTP Support (Native HTTP)
- Database Tuning Advisor
- Enhanced Read ahead & scan
- Online Index Rebuild
- Multiple Active Result Sets
- Persisted Computed Columns
- Queuing Support
- Snapshot Isolation Level
- Non-blocking Read Committed
- Table and Index Partitioning
- NUMA support

Database Availability and Redundancy
- Fail-over Clustering (up to 8 node)
- Enhanced Multi-instance Support
- Database Mirroring
- Database Snapshots

XML
- XQUERY Support (Server & Mid Tier)
- XML Data Manipulation Language
- FOR XML Enhancements
- XML Schema (XSD) Support
- MSXML 6.0 (Native)
- .Net XML Framework Support for XML and CLR integration

Notification Services

Replication
- Auto-tuning Replication Agents
- Oracle Publication
- Improved Blob Change Tracking

OLAP and Data Mining
- Analysis Management Objects
- Windows Integrated Backup and Restore
- UDM, multi-fact table support
- DTS and DM Integration
- Eight new DM algorithms
- Auto Packaging and Deployment

SQL Integration Services (SQLIS)
- New Architecture (DTR + DTP)
- Complex Control Flows
- Control Flow Debugging
- For Each Enumerations
- Property Mappings
- Full Data Flow Designer
- Full DTS Control Flow Designer
- Graphical Package Execution
- Immediate Mode and Project Mode
- Package (Advanced) Deployment Tools
- Custom Tasks and Transformations

Reporting Services
- Multiple Output Formats
- Parameters (Static, Dynamic, Hierarchical)
- Bulk Delivery of Personalized Content
- Support Multiple Data Sources
- STS (Web Parts, Doc Libraries)
- Visual Design Tool
- Charting, Sorting, Filtering, Drill-Through
- Scheduling, Caching
- Complete Scripting Engine
- Scale Out architecture
- Open XML Report Definition

Dynamic Management Views

Management Tools
- SSMS
- MDX Query Editor
- Version Control Support
- SQLCMD Command Line Tool
- SQL Service Manager

Database Maintenance
- Backup and Restore Enhancements
- Checksum Integrity Checks
- Dedicated Administrator Connection
- Dynamic AWE
- Fast Recovery
- Highly-available Upgrade
- Online Index Operations
- Online Restore
- Parallel DBCC\Index Operations

Performance Tuning
- Profiler Enhancements
- Profiler/Perfmon Integration
- Profiling Analysis Services
- Exportable XML Showplan
- Exportable Deadlock Traces

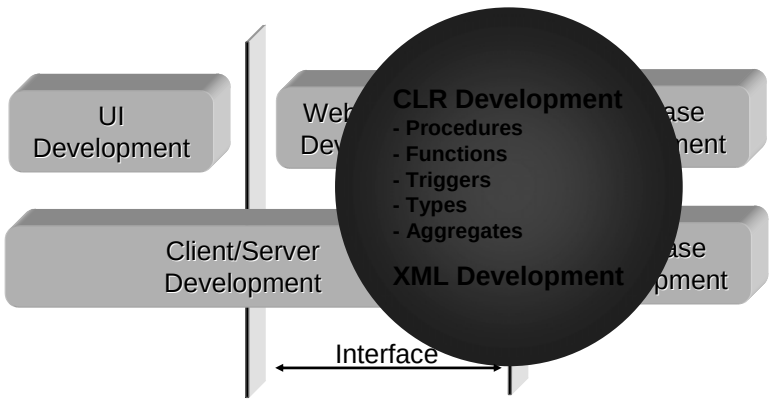
Full-text Search
- Indexing of XML Datatype
- Index files integrated with SQL backup

MDAC
- Side by Side installation
- Microsoft Installer base setup
- Support for Active Directory Deployment

SQL Client .NET Data Provider
- Server Cursor Support
- Bulk load support

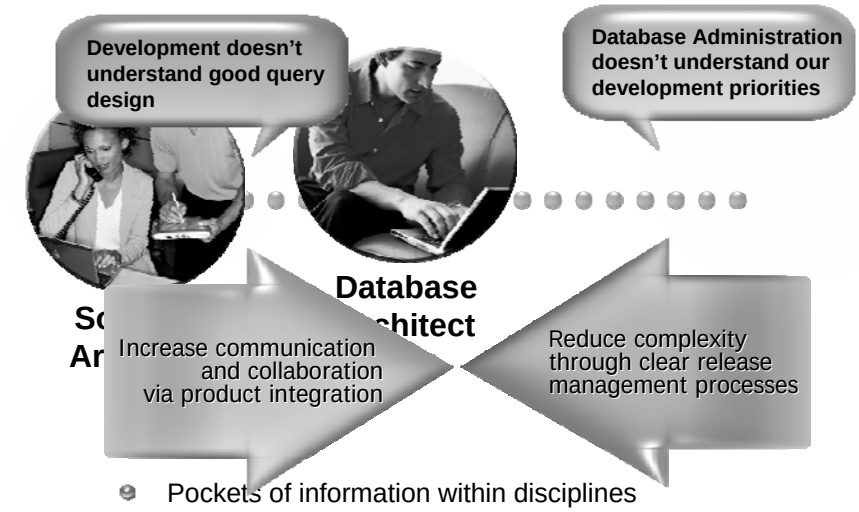
Security
- No default metadata access
- Fine Grained Administration Rights
- Separation of Users and Schema
- SSL Encryption at installation
- Data Encryption
- Code-signing

Developer and DBA Convergence



Developers and DBA's have traditionally worked to an interface, defined by:
- Table or View definition
- Stored Procedure & result set definition

Communication Barriers



- Pockets of information within disciplines
- Unclear delineation of responsibilities
- Conflicting best practices and architectures
- Conflicting strategic goals and objectives

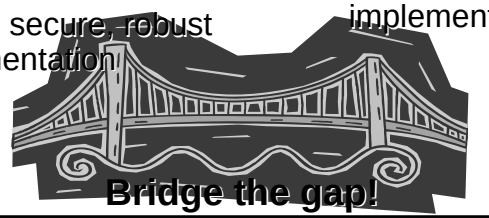
Developer and DBA Convergence

Developers:

- Get productive with the new tools
- Closer integration of DBAs into the development process
- Design application to be:
  - Supportable
  - Resilient to change
- Ensure secure, robust implementation

DBAs:

- Maintain role as data steward
- Understand & advise on appropriate technology choices
- Factor XML & CLR into schema design
- Ensure secure, robust implementation



Overview

- Installing SQL Server 2005
  - Clean install
  - Components
  - Security Settings during Install
    - Setting the startup Service Account
    - Choosing Authentication
- Verifying Initial Configuration
  - SQL Server Surface Area Configuration
    - Services
    - Features
  - SQL Server Configuration Manager
    - Service Account Changes
    - Encryption

Installing SQL Server 2005

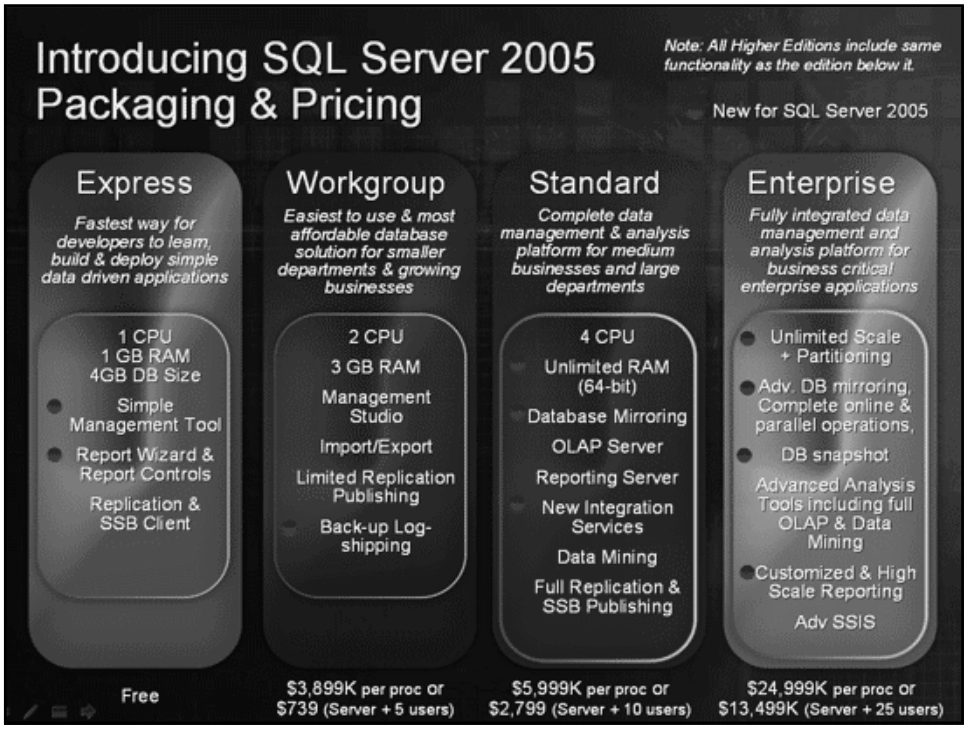
- Getting a current build
- Installation through Windows Installer
- Pre-installation checklist
  - Operating System Choices
  - Visual Studio – Beta?
- Versions from which to choose
  - 32-bit v 64-bit (Itanium and x64)
  - Developer, Enterprise, Express…
- Licensing and features
- SCC (System Configuration Checker)
- Installation

Build Naming Conventions

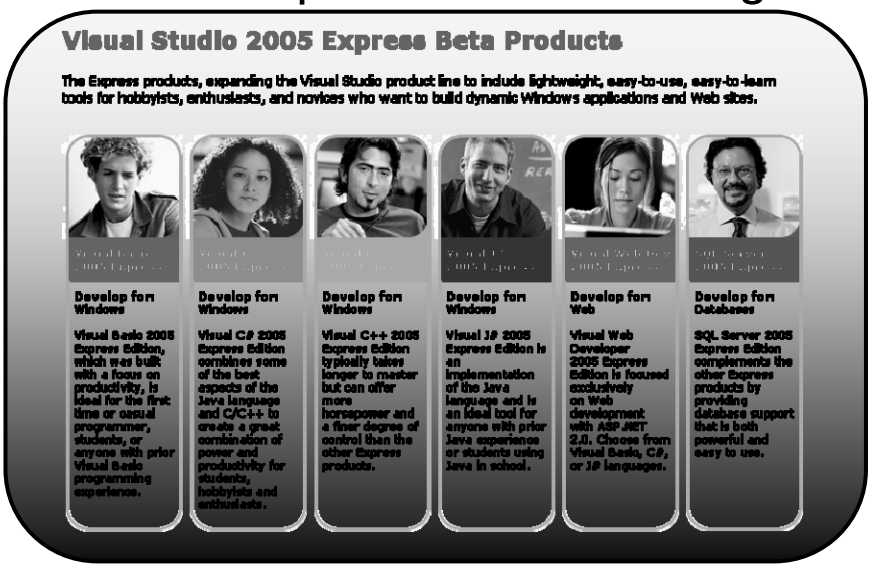
- Stability Improves
- Interim builds/Internal Releases
- IDW – Internal Developer Workstation
- CTP – Community Tech Preview
- Beta – Wider Release
- RC – Release Candidate
- RTM – Release to Manufacturing (“golden”)
- RTM + Hotfix – Only to specific customers
- RTM + SP Beta – Service Pack Beta
- RTM + SP – Service Pack Release

Choosing a Product to Install

- SQL Server 2005 Developer Edition
  - 32-bit and 64-bit
    - 64-bit (Itanium and x64 for AMD Athlon and Intel EMT64)
- SQL Server 2005 Standard Edition
  - 32-bit and 64-bit
- SQL Server 2005 Workgroup Edition
  - 32-bit only
- SQL Server 2005 Express Edition
  - 32-bit only
  - Tools in separate MSI (XM Download)
- Can run multiple versions side-by-side



Microsoft Express Product Range



Brian's article: *Get a Lean, Mean Dev Machine with the Express Editions of Visual Basic and SQL Server 2005*
http://msdn.microsoft.com/msdnmag/issues/04/09/ExpressEditions/toc.asp?frame=true

Installation as an “Application”

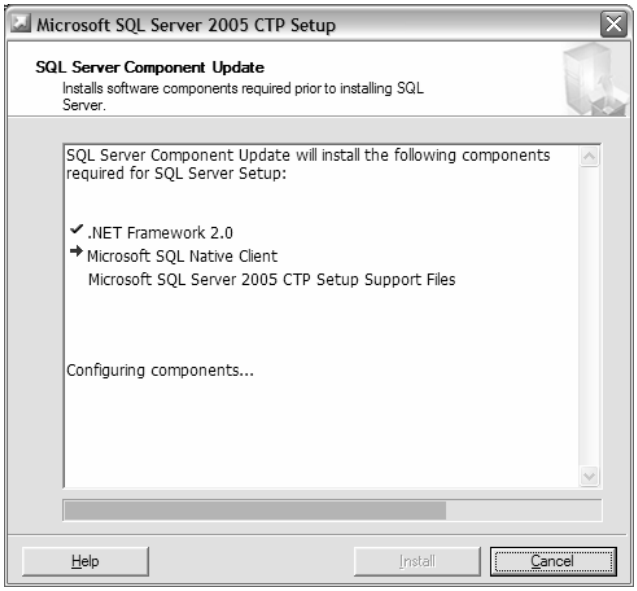
- Not using Install Shield
  - Third party product
  - Hard to “control”
  - Only stopped installation before file copy: “Installation has enough information to start copying files…”
  - Limited rollback capabilities, uninstalling partial and failed install was messy
- Uses Windows Installer
  - Microsoft product
  - Can download/review Windows Installer SDK
    - MSDN, Win32 and COM Development, Administration and Management, Setup, Windows Installer
  - Components have an MSI file
  - Installation can be rolled back until the very end

Options for Installation

- Local Installation – using GUI
- Local “unattended” and/or “silent” installation
  - Even for clusters!
- Remote installation

Component Update

- .NET Framework 2.0
- SQL Native Client (SNAC)
- Support Files



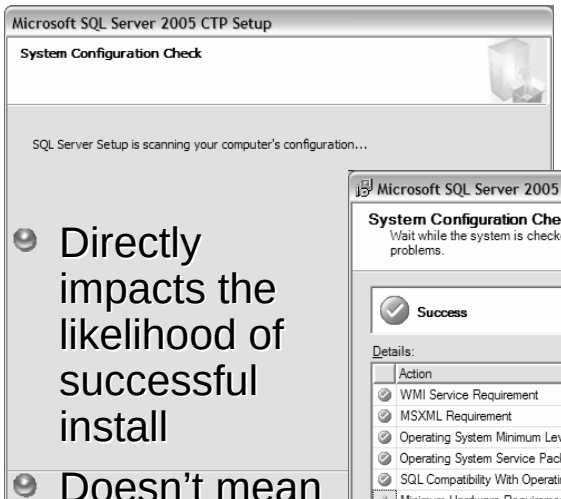
.NET Framework 2.0

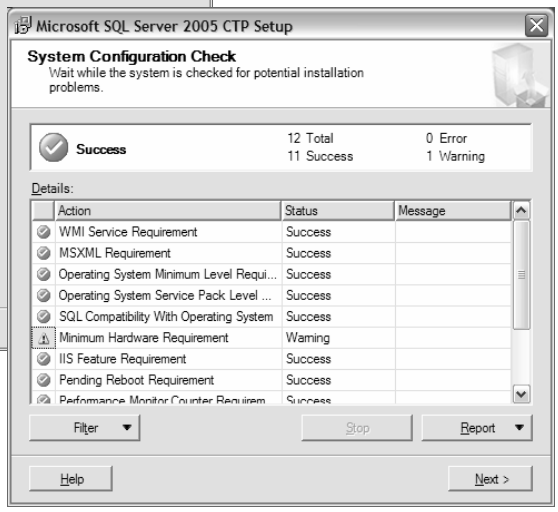
- Required for:
  - SQL Server Management Studio
  - Express Manager (used for SQL Express)
  - Engine (used by SQLCLR)
- How to Identify .NET Framework Version on your Machine
  - %WINDIR%\Microsoft.NET\Framework\version Right-click mscorlib.dll, click Properties, and then click Version.
  - Administrative Tools, .NET Framework 2.0 Configuration. At the top of the right pane, .NET Framework version displays.

SQL Native Client

- Why SNAC?
  - Separates “infrastructure” from “drivers”
  - SNAC is owned by SQL and only updated by SQL (during install/service packs)
  - SNAC is on the “drivers” side
  - Infrastructure owned by OS and only updated by OS
  - Enables side-by-side driver installation with minimal downtime
  - MDAC doesn’t allow side-by-side therefore requires stopping the app
  - MDAC to be removed on 64-bit as well

System Consistency Checker





- Directly impacts the likelihood of successful install
- Doesn’t mean you’ll have success with upgrade…

Hardware Recommendations

- File System—NTFS, no compression for installation drive or primary database file or log file
- More smaller/faster disks over fewer/larger disks
- Be sure to update all drivers and firmware!
- OS should be on redundant disks
- SQL installation should be on redundant disk
- Transaction logs should be on fast, redundant and preferably isolated disks
- TempDB should be isolated and if on multi-proc use multiple files (Q328551)
- Databases should be “moved” to support better performance (Q224071)
- KB Articles ⇨ Q224071 and Q328551 both on SQL Server 2000; best practices still apply

Components to Install

- Separate “packages” (.MSI files)
- High Level is for “typical” set of features
- Click Advanced for feature tree

Components to Install

SQL Server Database Services	Database Engine, Replication, Full-Text
Analysis Services	Analysis Services server components for Business Intelligence
Reporting Services	Reporting Services and Report Manager
Notification Services	Notification Services (Engine and Client), Bulk Event for XML
Data Transformation Services (Integration Services)	SQL IS replaces SQL DTS
Workstation components, Books Online and development tools	Management Tools, Command Prompt Tools, Reporting Services Tools, Connectivity Components, Programming Models, Management Studio, Computer Manager, Profiler, and Replication Monitor Business Intelligence Development Studio Documentation and samples - Books Online, code samples

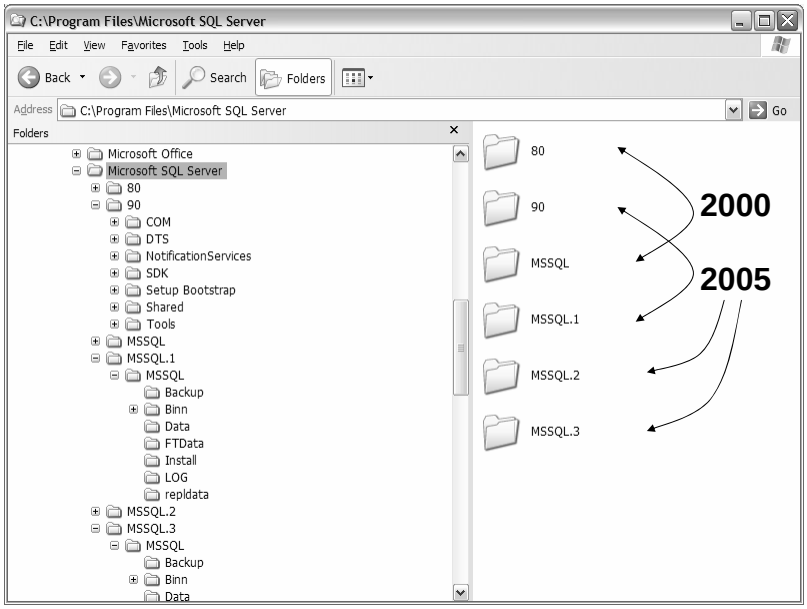
Default and Named Instances

- Default Instance
  - Accessed by “computername”
  - SQL Server Service = mssqlserver
  - SQL Server Agent Service = sqlserveragent
  - Full-Text = MSFTESQL
  - Analysis Server = MSSQLServerOLAPService
- Named Instance
  - Accessed by computername\instancename
  - SQL Server Service = mssql$instancename
  - SQL Server Agent Service = sqlagent$instancename
  - Full-Text = msftesql$instancename
  - Analysis Server = MSOLAP$instancename

Directory Structure

- No longer dependent on instance name
- All directories MSSQL.n even for other components such as Analysis Services
- How can you detect?
  - Registry: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Instance Names\SQL
  - Name = Instance name (without MSSQL)
  - Data = Install Directory (mssql.n)

Installation Directories

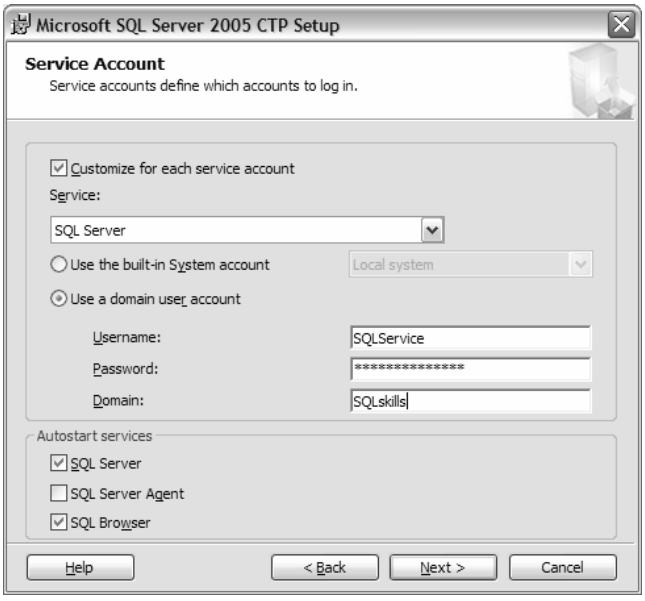


Programmatically Detecting Dir

```
DECLARE @InstanceName      sql_variant,
        @InstanceDir sql_variant,
        @SQLDataRoot nvarchar(512),
        @ExecStr             nvarchar(max)

SELECT @InstanceName = ISNULL(SERVERPROPERTY('InstanceName'),
    'MSSQLServer')

EXECUTE master.dbo.xp_regread 'HKEY_LOCAL_MACHINE',
   'SOFTWARE\Microsoft\Microsoft SQL Server\Instance
   Names\SQL', @InstanceName, @InstanceDir OUTPUT

SELECT @ExecStr = 'EXECUTE master.dbo.xp_regread '
   + '''HKEY_LOCAL_MACHINE'', '
   + '''SOFTWARE\Microsoft\Microsoft SQL Server\'
   + convert(varchar, @InstanceDir)
   + '\Setup'', ''SQLDataRoot'', @SQLDataRoot OUTPUT'

EXEC master.dbo.sp_executesql @ExecStr
   , N'@SQLDataRoot nvarchar(512) OUTPUT'
   , @SQLDataRoot OUTPUT
```


Startup Service

- Customize allows you to set each individually and to different accounts
- Local System *or*
- Domain Account…



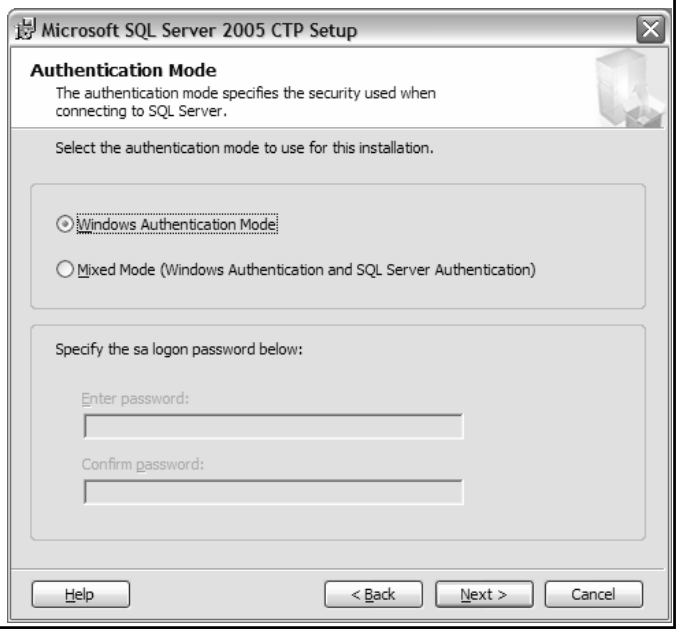
Service Account Choices

- Starting with most secure/controlled
  - Dedicated Local account* (not local administrator)
  - Local service (do other services use this?)
  - Dedicated Domain account (not local administrator)
  - Network service (do other services use this?)
  - Local account* which is a local administrator
  - Domain account which is a local administrator
- NEVER use:
  - Local System
  - Domain Administrator

*** Local account information is stored in the local SAM. Domain accounts are likely to be far more secure from a physical perspective.**

Authentication

- Windows
- Mixed
- Password only specified for Mixed (change as of IDW13) due to unattended installation requirement



Windows Authentication

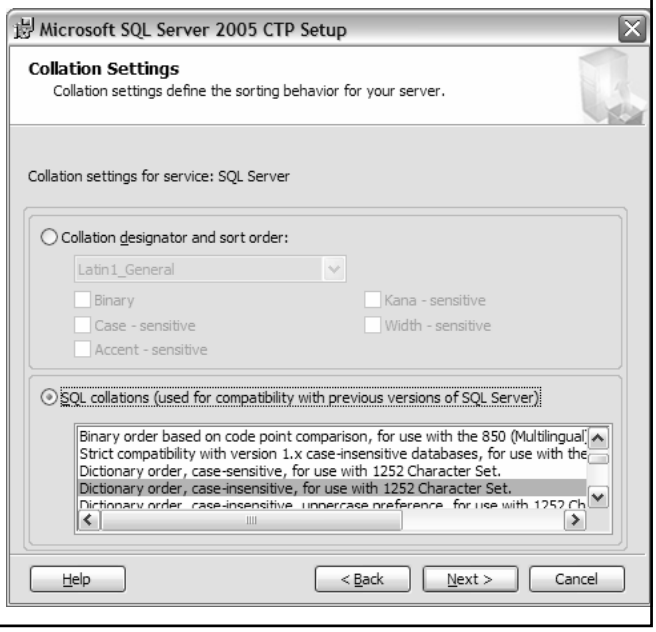
- More Secure
  - Strong Password requirements
  - Account Lockout policies
  - Password expiration
- Can be changed later
- After installation be sure to manually set the password of the sa account otherwise internally set to a randomly generated password—if you change to mixed mode you will “enable” your sa account but not know the password. However, you can easily change it.

Mixed Mode Authentication

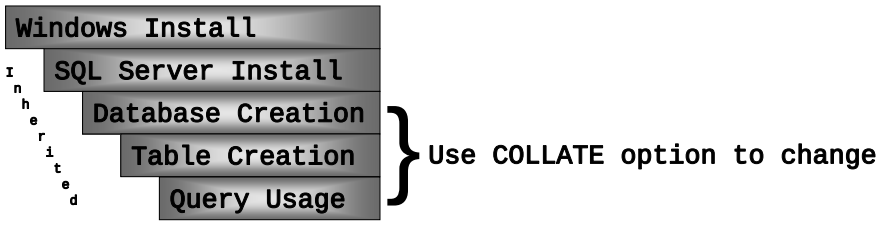
- Windows Authentication plus
- SQL Server Authentication
  - Uses Windows 2003 Server password policies, if installed on Windows 2003 Server
  - Requires sa password to be set during installation if Mixed mode is chosen
  - Stronger password requirements in 2005
    - All passwords must be at least 6 characters long and satisfy at least three of the four criteria:
      - It must contain uppercase letters (A-Z)
      - It must contain lowercase letters (a-z)
      - It must contain numbers (0-9)
      - It must contain non-alphanumeric characters (!, #, %, or $)
    - See BOL for complete list of requirements

Collation Setting

- Affects searches, lookups and ORDER BY
- Character data is affected:
  - ASCii
    - char
    - varchar
    - text
  - Unicode
    - Nchar
    - Nvarchar
    - Ntext



Collation Concepts



- May be changed at any point in chain
- Different options offer different gains – most in language sorting, some in performance
- Can assign on the fly for just a single query – or within a view for subsequent use
- Computed Columns and Indexed Views can aid in performance for lookups with changed collations (there are likely to be version specific restrictions)

Collation Options

- To see the list of collations
  SELECT * FROM ::fn_helpcollations()
- To see the server's setting
  sp_helpsort
- To see the database's setting
  sp_helpdb dbname
- To see the table's setting (for each column)
  sp_help tname
- For more information, check out these BOL topics:
  - COLLATE
  - "Specifying Collations"

Initial Configuration

- Off by default
- SSL Encrypted Communications
- Using SQL Server Configuration Manager
- Running SQL Server 2000 and SQL Server 2005 side-by-side
- Special Feature: Imaging
  - Renaming a Windows computername – without ever having started the SQL Server instance, special case
  - Database Servers/Instances
  - Analysis Services

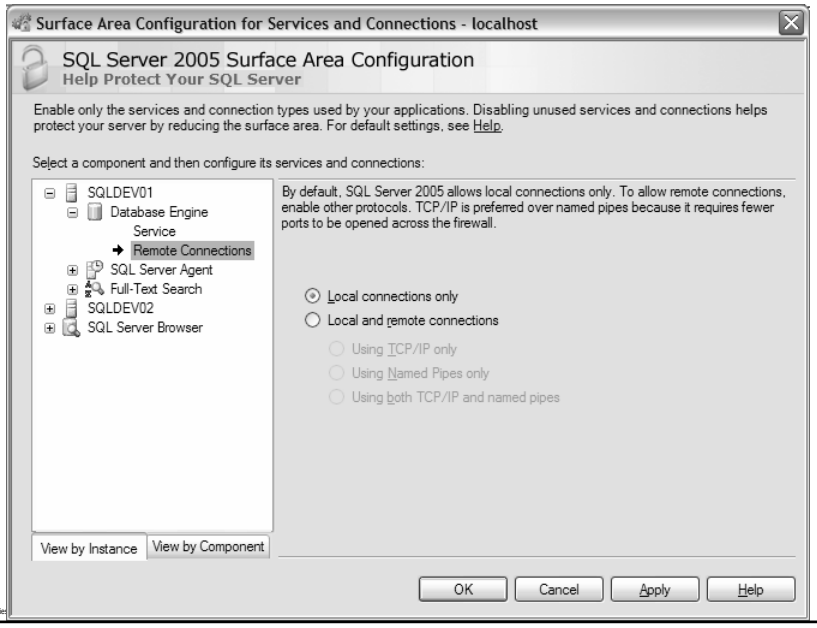
Off by Default

- Principle of least privilege
- Minimal Configuration to run – many installed services (even when chosen during installation) are “off by default”
- Secure by Design
- Secure by Default
- Secure by Deployment
- Where to enable, where to view?
  - sp_configure, Computer Management, catalog views, no single central way
  - ⇨ Enter SQL Surface Area Configuration

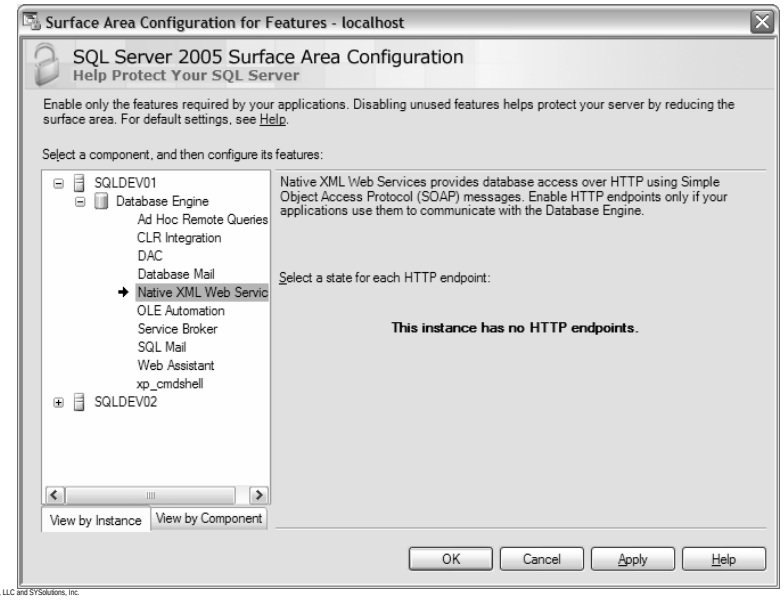
Surface Area Configuration

- SQL Server Program Group, Configuration Tools
- Review SQL Server Setup Help from link on main dialog
- Configure Services and Protocols
  - In this first release more options in Configuration Manager
- Surface Area Configuration for Features
  - “Lockdown” most common/vulnerable features
  - sp_configure settings
  - Catalog View queries to view (endpoints)

Services – Local Only?



SQLSAC – HTTP Endpoints

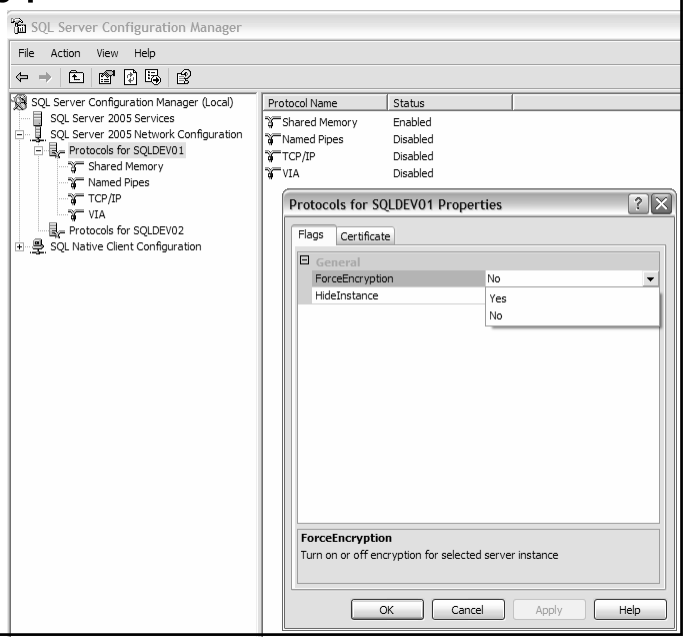


Encrypted Communications

- SSL is used even if not present at installation
  - Checked at installation, if present used
  - If not present, SQL Server generates a 1024 bit certificate
- Logins/passwords encrypted always
- Uses certificate created at installation if SSL not present
- Use “Certificate Picker” in SQL Server Configuration Manager to explicitly choose SSL certificate

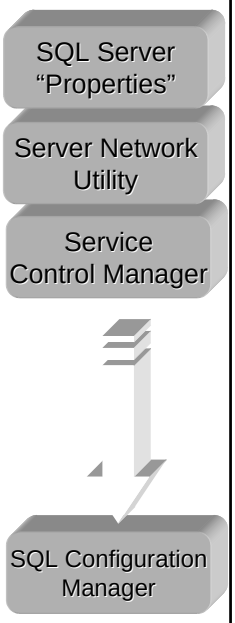
SSL Encryption/Certificate Picker

- Protocol Properties
- Login/pwd encrypted
- ALL traffic encrypted = YES
- Certificate Picker tab

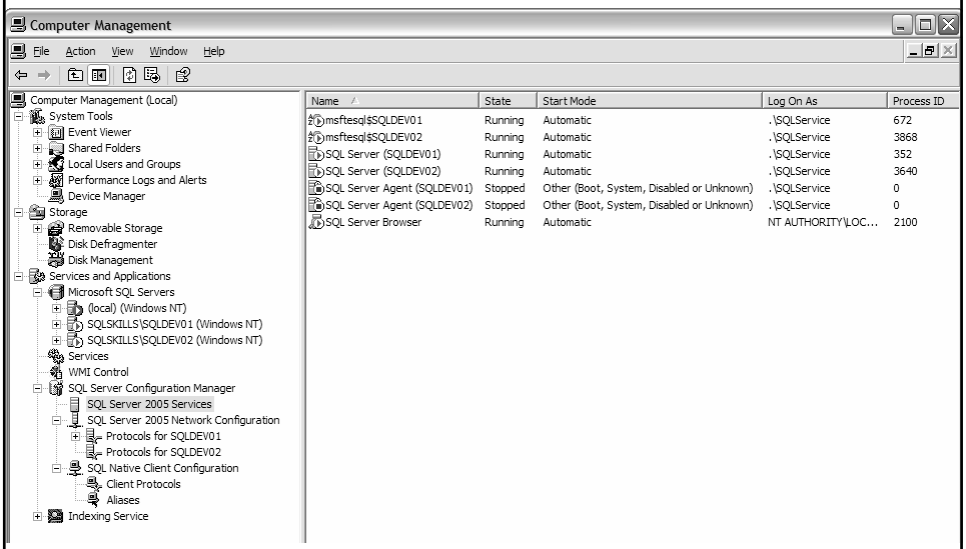


Configuration Manager

- Start and Stop Services
- Change auto start and service properties
- Does not require your services to be started to change properties
- Service Account password changes don't require restart of service
- Control Supported Network libraries and SSL encryption
- Create aliases
- Launch from:
  - SSMS, Right click Configure Services
  - SQL Configuration Manager
  - Computer Management



Computer Manager – 2000/2005



Running 2000/2005 “Side-by-side”

- Possible combination:
  - SQL Server 2000 default/named instance(s)
  - SQL Server 2005 default/named instance(s)
    - SQL Server 2005 can support multiple instances of multiple versions (Developer and Express)
- SQL Server 2000
  - Directory structure was related to instance type and name
- SQL Server 2005
  - Directory structure NOT related to instance type or name
- Re-register COM Components?

Renaming A Server After Startup

- Server name is NOT detected by SQL Server
- Review the list of servernames

```
sp_helpserver
```

- Drop the old servername

```
sp_dropserver 'oldservername'
```

- Add the new servername – make sure you define it as the “local” instance of SQL Server

```
sp_addserver 'newservername', local
```

- May have problems with servername related objects/jobs – i.e. msdb.dbo.sysjobs

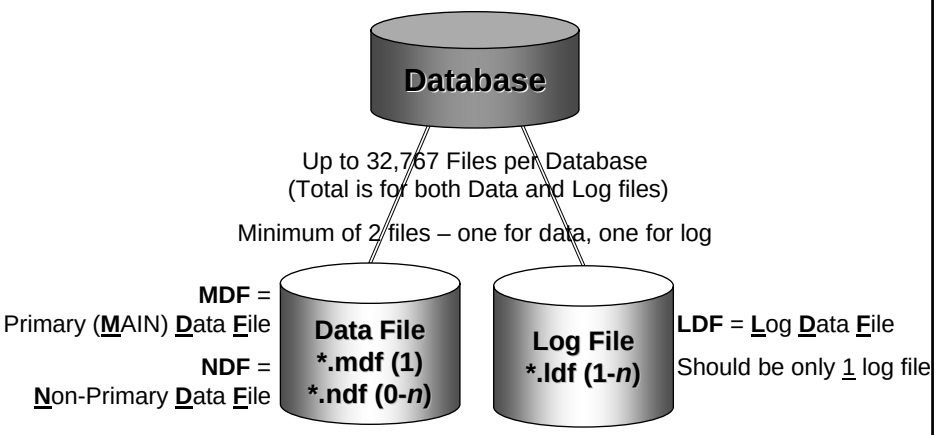
```
update sysjobs
  set originating_server =  'newservername'
where originating_server =  'oldservername'
```

Optimal Database Structures

- Anatomy of Data Modifications
- Optimizing Data Files
- Logging and Recovery
- Transactions

Database Structure

Up to 32,767 Databases per Instance



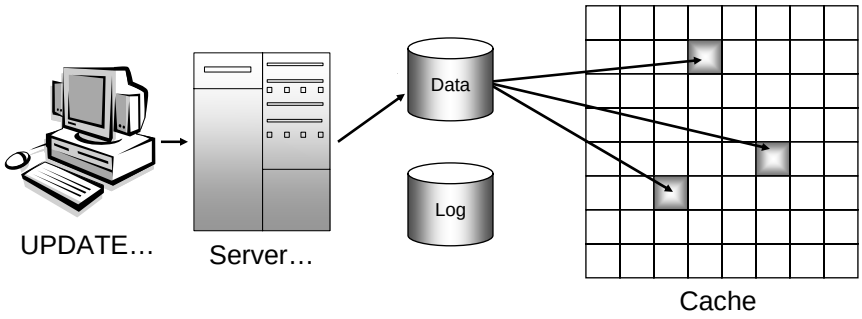
The Anatomy of a Data Modification

1. User sends UPDATE
   - Update is highly selective (only 5 rows)
   - Indexes exist to aid in finding these rows efficiently
   - The update is a SINGLE statement batch NOT enclosed in BEGIN TRAN…COMMIT TRAN block therefore this is IMPLICIT transaction

The Anatomy of a Data Modification

2. Server receives the request and locates the data in cache OR reads the data from disk into cache
   - Since this is highly selective only the necessary pages are read into cache (maybe a few extra but that's not important here)
   - Let's use an example where the 5 rows being modified are located on 3 different data pages

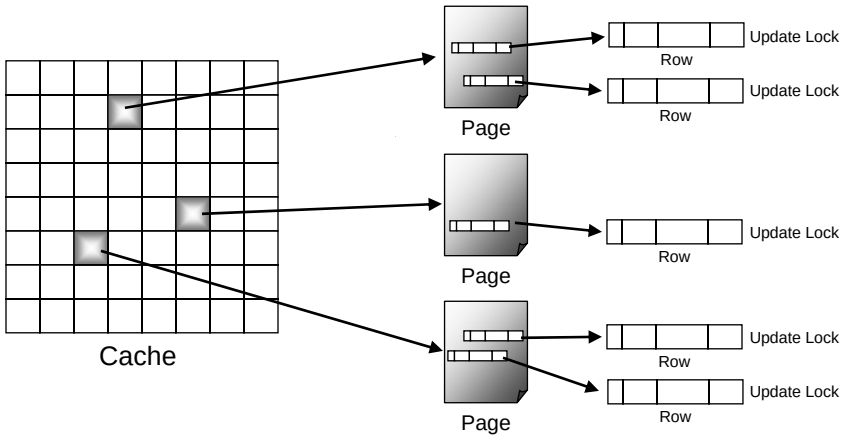
What it looks like:
Data



The Anatomy of a Data Modification

3. SQL Server proceeds to lock the necessary data
   - Locks are necessary to give us a consistent point FOR ALL rows from which to start
   - If any other transaction(s) have ANY of these rows locked we will wait until ALL locks have been acquired before we can proceed.
   - In the case of this update *(because it's highly selective and because indexes exist to make this possible)* SQL Server will use row level locking.

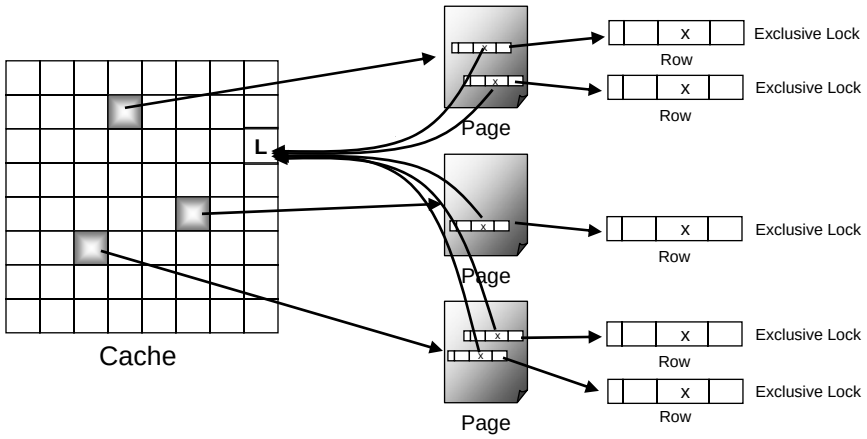
What it looks like:
Locks



The Anatomy of a Data Modification

4. SQL Server can now begin to make the modifications – for EVERY row the process will include:
   1. Change to a stricter lock (eXclusive lock)
      - **An update lock helps to allow better concurrency by being compatible with other shared locks (readers). Readers can read the pre-modified data as it is transactionally consistent**
      - **The eXclusive lock is required to make the change because once modified no other reads should be able to see this un-committed change**
   2. Make the modification (in cache)
   3. Log the modification to the transaction log pages (also in cache)

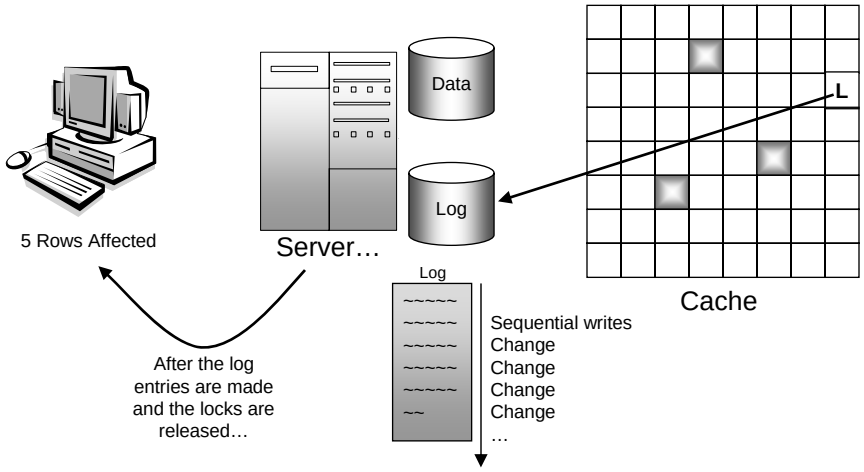
What it looks like:
Modifications



The Anatomy of a Data Modification

5. Finally, the transaction is complete – this is the MOST critical step
   - All rows have been modified
   - There are no other statements in this transaction – i.e. Implicit transaction
   - Steps are:
     1. **Write all log pages to transaction log ON DISK**
     2. **Release the locks**
     3. **Send a message to the user:**
        (5 Rows Affected)

What it looks like:
Write-Ahead Logging



So now what?

- The transaction log ON DISK – is up to date
- The data in CACHE – is up to date
- But when does the data get written from cache to disk?

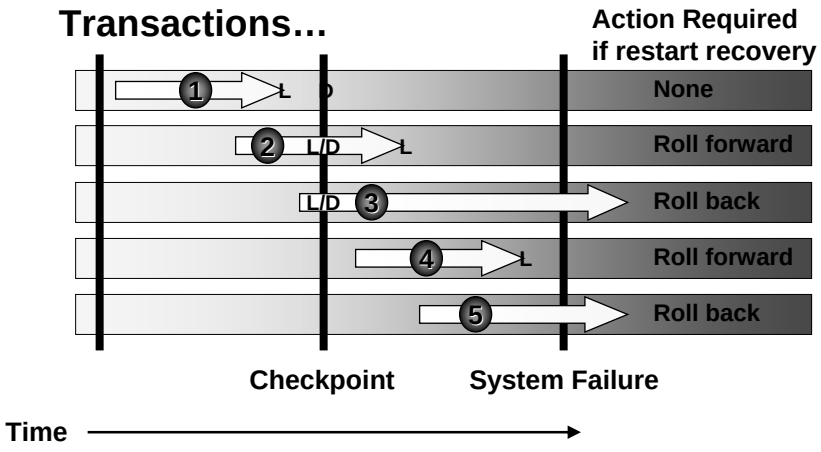
Checkpoint

Checkpoint

*It's important to realize that a checkpoint does NOT just write committed pages… Instead a checkpoint writes **ALL** pages which have changed since they were brought into cache – **regardless** of the state of the transaction which changed them!*

- Why?
  - To reduce roll-forward recovery time during restart recovery
  - To batch I/Os to disk and reduce disk thrashing for data writes

Transaction Recovery and Checkpoints



Restart Recovery

- Automatically on system restart
- SQL Server 2000
  - Redo
  - Undo
  - Users allowed access to database
- SQL Server 2005 (Enterprise Edition Only)
  - Redo
  - Users allowed access to database
  - Undo
  - Faster access to database – including faster failover on Cluster
  - Records are protected (i.e. Locked)

Key Points

- Data Portion mostly random reads – except at checkpoint
- Log Portion mostly sequential writes
- Separate physical disks minimizes contention at the drive level – first choice in tuning
- Log is critical in recovery
- Protect the log
- Minimize impact to log
- Be aware of environmental changes

Optimizing Data Files

- Defrag the physical disks
- Effective RAID array configuration
- Pre-allocate to a reasonable initial size
- Don't let auto-growth get out of control
- Consider allowing Fast File Initialization
  (Enterprise Edition Only—Perform Volume Maintenance Tasks to SQL Server service account)
- While a lot of these will help for the log (and I'll explain why in a moment) there are more important things to be aware of IN the data portion – tables and indexes (rebuilding)

Growing Files Automatically

- Enable Data Files and Log File to Grow Automatically
- Allocate Reasonable Initial Sizes
- Set Maximum Size for Data Files so you don't run out of space when you have no disk space!
  - Especially important for server consolidation and/or shared disks re: multiple databases
- Set File Growth Increment to Reasonable Size
  - Preference – fixed increment less than or equal to 1GB
  - Helped by fast file initialization (SQL Server 2005 only)

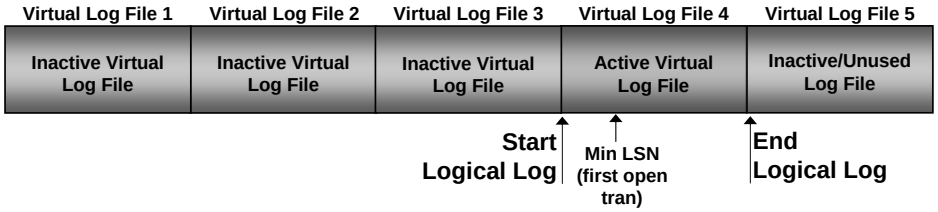
Optimizing Log Files

- Isolate **the** Transaction Log (only one!)
- Defrag the physical disks
- Effective RAID array configuration
- Pre-allocate to a reasonable initial size
- Don't let auto-growth get out of control
- Check and fix your internal fragmentation
- Don't be caught up in nothing but speed!
- See Kim's blog entry: 8 Steps to Better Transaction Log Throughput, June 25, 2005

Log files

- Limited read activity
  - Replication
  - Rollback
- Optimize for write activity
  - RAID 0+1 w/hot spare
  - RAID 1
  - RAID 1+0 better redundancy
  - preferably NOT RAID 5
- Optimize for SQL Server "logging" activity
  - Capacity Planning (initial and reasonable size)
  - Minimize excessive autogrowths (clean up excessive VLFs)

How the Transaction Log Works



- On commit, activity is written to the log – active and likely sequential
- Activity moves through log sequentially, fills and then goes to a second file or autogrows
- Excessive autogrowth causes:
  - Slight pause on autogrow
  - Windows call to extend a file (may cause file fragmentation)
  - Adds “VLFs” to the file on each autogrowth

Transaction Log Optimization

- Pre-allocate the log to appropriate size
- Minimize autogrowths
- Log might NOT show 0 Percent Log Used after backup for example - the % Log Used will be ~6%
- Disk Bound Systems might experience some performance degradation at log backup
- Consider RAID 1+0 instead of RAID 1, RAID 5 generally is not recommended
- Optimize existing file – minimize VLFs

The Effects of Logging

- Log is written AHEAD of the data portion
- Log is the ONLY place where transactional consistency is known (i.e. guaranteed)
- Once a checkpoint occurs SQL Server doesn't need the information in the log – for committed (or inactive) transactions (the log could even be cleared however…)
- Without the transaction log a database cannot function (i.e. marked suspect)
- Need to make sure this is redundant AND optimal…
- What can effect the logging?

Minimize impact of logging

- Cannot clear the log unless the transactional information is inactive – if a transaction is pending then it stays in the log until it's completed…
- Restart recovery will take longer if there were a lot of long running transactions pending at the time of failure
- Always try to avoid:
  - Long running transactions
  - Transactions which span more than one batch
  - Transactions that require user interaction
  - Nested Transactions (complex and usually longer running)

Consider changing recovery model

What can impact your ability to recover? Recovery Models

- Full ⇨ Everything is Fully Logged
  - All operations allowed and FULLY logged
  - Operations like creating or rebuilding an INDEX takes as large a log as the size of the operation
- Bulk_Logged ⇨ Minimal Logging for SOME Operations (not NON-LOGGED)
  - Operations whose performance is affected, see BOL Topic ⇨ Minimally Logged Operations
  - ALL other operations (i.e. updates, inserts, etc. take the same log space and time as the FULL recovery model
- Simple ⇨ Log Truncation on Checkpoint

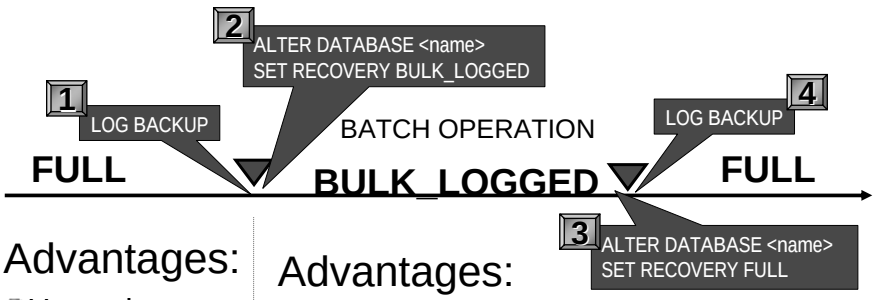
Logging, Recovery Models, Recovery and Performance

- Acceptable to Switch?
- How, Why and Why Not?
- Batch Processing Scenario
- Batch Processing Script
- Switching Recovery Models – Summary

Acceptable to Switch?

- Yes
  - If users are not allowed in database during batch processing
  - No modifications are made during batch process that are not recoverable through some other means (i.e. re-running the batch process)
  - Data loss during the batch process is OK
- No
  - Modifications are made by users/processes that are not recoverable
  - OLTP System requiring 24x7x365
  - Data loss is NOT acceptable
  - Database Mirroring

Switching Recovery Models



Advantages:

- Up-to-the-minute recovery
- Point-in-time recovery
- Access to "tail" of the log

Advantages:

- Minimal logging
- Faster Bulk Operation

Disadvantages:

- No *STOPAT*
- Data loss possible if media failure…
- If database becomes suspect, cannot back up tail of log.

Switching Recovery Models
Using ALTER DATABASE

```
-- Check the status before changing it!
IF DATABASEPROPERTYEX('dbname', 'Updateability') =
'READ_ONLY'
      ALTER DATABASE dbname
            SET RESTRICTED_USER,
            READ_WRITE WITH ROLLBACK AFTER 60
go
-- Perform Batch Operations…
go
ALTER DATABASE dbname
      SET READ_ONLY, MULTI_USER
      WITH ROLLBACK AFTER 30
go
```

Make sure to review the sample script with tons of comments!

Availability, What About Locking?

- ACID Transaction Design Requirements
  - Atomicity Consistency Isolation Durability
- Isolation Levels
  - Level 0 – Read Uncommitted
  - Level 1 – Read Committed
  - Level 2 – Repeatable Reads
  - Level 3 – Serializable
- Default Isolation Level in BOTH 2000/2005 is ANSI/ISO Level 1, Read Committed
- In implementation this default level uses locking

ACID Properties

- Atomicity
  - A transaction must be an atomic unit of work; either all of its modifications are performed, or none
- Consistency
  - When completed, a transaction must leave all data and all related structures in a consistent state
- Isolation
  - A transaction either sees data in the state it was in before another concurrent transaction modified it, or it sees the data after the second transaction has completed, but it does not see an intermediate state
- Durability
  - Transaction should persist despite system failure

Isolation Level 0 – Read Uncommitted
Phenomenon: Dirty reads

- A read transaction can read another transaction's uncommitted (or in-flight) changes – resulting in "dirty reads"
- DML statements always use exclusive locking
- In Implementation: row locks are not used (SCH_S locks are used) for the dirty read transaction and locks against data being accessed are not honored
- Resulting Phenomenon: statements execute with the possibility of inaccurate data since the "in-flight" data read may continue to change or even be invalidated (rolled back)

Isolation Level 1 – Read Committed

Phenomenon: Inconsistent analysis

- The default behavior in ALL releases
- In-flight transaction's data cannot be read by a read committed transaction – only committed changes are visible
- DML statements always use exclusive locking
- In Implementation: locks are released (for readers – not modifiers) as resources are read, a row may be read more than once in some scenarios
- Resulting Phenomenon: Reads are not repeatable through the life of a transaction – as a result a row may not be read consistently during the life of a transaction

Isolation Level 2 – Repeatable Read

Phenomenon: Phantoms

- In-flight transaction's data cannot be read and data modified is accessible only to the repeatable read transaction
- Data read, but not modified is accessible to other transactions for reads, but not DML
- In Implementation: locks are held for the life of a transaction, rows which are read are locked and can be repeatably read during the life of a transaction
- Resulting Phenomena: rows which were not present at the beginning of the transaction can appear – in the result

Isolation Level 3 – Serializable
Phenomenon: None

Key range and index reminder

- In-flight transaction's data cannot be read and data modified is accessible only to the serializable transaction
- Data read, but not modified is accessible to other transactions for reads, but not DML
- In Implementation: locks are held for the life of a transaction and held at higher levels within indexes to prevent rows from entering the "set"
- Implementation side effect: to prevent rows from entering the "set" of data, the "set" of data needs to be locked. If appropriate indexes do not exist then higher levels of locking might be necessary – i.e., table-level locking.

Isolation in Implementation

- All versions of SQL Server – including SQL Server 2005 – use a locking-based implementation by default
- In the default environment and to reduce the possibility of various anomalies, transactional duration and possibly higher levels of locking are necessary – resulting in blocking
- There are benefits of locking, if true "isolation" is desired

Isolation in Implementation (cont'd)

- SQL Server 2005 combines an optional row versioning-based implementation with this locking-based implementation
- Two end results (and methods) for implementing row-level versioning
  - Read Committed using Statement-level Snapshot

    Also known as:
    Statement-level Read Consistency
  - Transaction-level Read Consistency
    - Snapshot Isolation

SQL Server 2005 Isolation Options

- Default – Uses Locking, no options set
- Read Committed with Statement-level Snapshot
  - DB Option: READ_COMMITTED_SNAPSHOT enabled
  - Uses locking for writes, versioning for reads for statement-level consistency
- Transaction-level Snapshot Isolation
  - DB Option: ALLOW_SNAPSHOT_ISOLATION enabled
  - Uses default locking for everything, ALLOWS snapshot for transactions when requested through
    SET TRANSACTION ISOLATION LEVEL SNAPSHOT
  - If requested, transaction-level consistency through row versioning
- Both Options Configured
  - Statement-level consistency through versioning automatically
  - Transaction-level consistency where requested

Setting Isolation Behavior

- Default Behavior
- Read Committed using Statement-Level Snapshot

```
ALTER DATABASE <database_name>
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK AFTER 5
```

- Snapshot Isolation

```
ALTER DATABASE <database_name>
SET ALLOW_SNAPSHOT_ISOLATION ON
```

- Both RCSI and Snapshot Isolation

```
ALTER DATABASE <database_name> SET RCSLS...
ALTER DATABASE <database_name> SET Snapshot...
```


Read Committed

Default Behavior

- All phenomena – except dirty reads are possible, even in the bounds of a single select query
- In volatile databases, a long running query may produce inconsistent results
  - Can increase isolation to remove phenomena
  - Increasing isolation requires locks to be held longer
  - This can create blocking

Tips to Minimize Blocking

- Write efficient transactions – keep them short and in one batch
- Do not allow interaction in the midst of the batch
- Use indexes to help SQL Server find – and lock – only the necessary data
- Consider estimates for long running queries and/or migrating data to a secondary analysis server
- Problems with locking becoming blocking are often when long running (and conflicting) transactions execute

Locking Prevents Conflicts

At the expense of blocking

- No other transactions can invalidate the data set through modifications
- Is this always what you want or need?
  - Queuing applications typically want locks, if someone is modifying that row – we want to go to another not see last transactional state
  - Very volatile prices – don't want to give them the last price if it's currently being updated, so wait… (*but the update's fast*)
- What about long running transactions or cases where you don't need absolute current…

Read Committed Snapshot Isolation

Database Changed to READ_COMMITTED_SNAPSHOT

- No phenomena are possible in the bounds of a single statement
- In volatile databases, a multi-statement transaction may yield different results for different statements which access the same data
- Each statement is consistent but only for the execution of that statement, not for the life of the transaction (*if the transaction has multiple statements*)
- Each time data is read by a new statement the latest version is used

Snapshot Isolation

Database Changed to ALLOW_SNAPSHOT_ISOLATION

- Setting ALLOWS users to ask for Snapshot Isolation – NOT on by default
- ALL phenomena and ALL default locking behavior is EXACTLY the same unless you explicitly ask for Snapshot Isolation
- Once requests, no phenomena are possible in the bounds of a transaction running under snapshot isolation
- In volatile databases, a multi-statement transaction will always see the transactionally accurate version which existed when the transaction started
- Versions must stick around longer, multi-statement transactions may have conflicts

Isolation Levels: In Summary

- READ UNCOMMITTED (Level 0)
  - "Dirty Reads" – An option ONLY for readers
  - Any data (even that which is in-flight/locked) can be viewed
- READ COMMITTED (Level 1 – Default)
  - Only committed changes are visible
  - Data in an intermediate state cannot be accessed
- Read Committed w/statement-level Snapshot
  - Statement-level read consistency
  - New non-blocking, non-locking (ex. SCH_S), version-based Level 1

Isolation Levels: In Summary (cont'd)

- REPEATABLE READS (Level 2)
  - All reads are consistent for the life of a transaction
  - Shared locks are NOT released after the data is processed – does not allow writers (does allow other readers)
  - Does not protect entire set (phantoms may occur)
- SERIALIZEABLE (Level 3)
  - All reads are consistent for the life of a transaction
  - Avoids phantoms – no new records
- Snapshot Isolation – 2005
  - Transaction-Level consistency using snapshot
  - New non-blocking, non-locking, version-based transactions

Controlling Isolation Levels
Table-level changes

From Clause, per table (no spaces)

- Level 0 – READUNCOMMITTED, NOLOCK
- Level 1 – READCOMMITTED (locking)
- Level 1 – READCOMMITTED (versioning)
  - Only in 2005 and *only* if the database option to READ_COMMITTED_SNAPSHOT is on
  - Can be overridden with READCOMMITTEDLOCK
- Level 2 – REPEATABLEREAD
- Level 3 – SERIALIZABLE, HOLDLOCK

```
FROM dbo.titles WITH(READUNCOMMITTED)
  JOIN dbo.publishers WITH(SERIALIZABLE)
```


Controlling Isolation Levels
Session-level changes (cont'd)

Session level settings impact entire session but can be overridden with table-level settings

- Level 0 – READ UNCOMMITTED
- Level 1 – READ COMMITTED
- Level 2 – REPEATABLE READ
- Level 3 – SERIALIZABLE

```
SET TRANSACTION ISOLATION LEVEL opt
  READ UNCOMMITTED
  READ COMMITTED
  REPEATABLE READ
  SERIALIZABLE
  SNAPSHOT
```

Only in 2005 and *only* if the database option is on

Allowing Read Committed using Statement-level Snapshot

- Database option

```
ALTER DATABASE <database_name>
  SET READ_COMMITTED_SNAPSHOT ON
  WITH ROLLBACK AFTER 5
```

- No other changes necessary…
- No changes to your queries or your applications – they automatically return with statement-level consistency (*NOTE: You may need to modify your applications if you depend on locking – re: queues, readers to wait for change*)
- Changes to blocking…
- However, if this is NOT your performance problem (meaning concurrency isn't your bottleneck) then you may hinder performance not improve
- Expect this change in behavior at a cost

Allowing Snapshot Isolation

- Database option

```
ALTER DATABASE <database_name>
  SET ALLOW_SNAPSHOT_ISOLATION ON
```

- Session setting

```
SET TRANSACTION ISOLATION LEVEL
  SNAPSHOT
```

- Changes to applications:
  - Request snapshot isolation
  - Test for conflict detection
- Expect this change in behavior at a *higher* cost

Potential Issues

- Cost in row overhead – when enabled, 14 bytes added to row
- If snapshot, do you depend on locking?
  - OK, most of you will say no but what about status queues…
  - Tip: Use READCOMMITTEDLOCK hint
- If Snapshot Isolation, could you have conflicts?
  - Be sure to have proper conflict detection and error handling, see Kim's Snapshot Isolation whitepaper for details and examples

Management/Monitoring

- Version Store in TempDB
- Versions removed when no longer needed
- Read committed using statement-level snapshot won't hold versions as long – in theory because only statement-level
- Snapshot isolation may have more impact on TempDB as versions held for life of transaction
- Lots of long running transactions may stress TempDB
- See Kim's Snapshot Isolation whitepaper for monitoring with DMVs (Dynamic Management Views)

Entities

- Basic principle behind a table – the way we describe some specific person, place or thing as a base object within the database
- Focus on a single element - Person, place, thing…
  - Authors – Each author is an Entity described by a row in the authors table.
  - Titles – Each title, separate from the author(s) that wrote the title is described by a row in the titles table.
  - Publishers – Each publisher, separate from any title(s) they may have published is described by a row in the publishers table.

Attributes

- Basic principle behind a column – the way we describe some specific element about the entity
- Each column of a table should describe ONLY that entity…
- Core elements of an attribute
  - Attribute Name price
  - Data type money
  - Nullability NULL

Every attribute should describe the primary key, the whole primary key, and nothing but the primary key.

Data Types

- System Supplied Data Types
  - Character
  - Unicode Character
  - Numeric
  - Date and Time
  - Exact Numeric
  - Uniqueidentifier
  - Binary
  - Miscellaneous

Find the right data type for the job:

Use Unicode over ASCii

If the datatype varies:
- **< 5 chars should be nchar**
- **5-20 chars – questionable**
- **> 20 char – lean more towards nvarchar**

For numeric data:
- **Find the right range**
- **Standardize on decimal or numeric**

Use uniqueidentifier sparingly
Understand precision and range

Data Type	Max/Exact Storage	Notes
char(n) varchar(n)	**8000 bytes**	**1 byte per char - 8000 Characters**
nchar(n) nvarchar(n)	**8000 bytes**	**2 bytes per char - 4000 Characters**
tinyint	**1 byte**	**0-255**
smallint	**2 bytes**	**-32768 to 32767**
int	**4 bytes**	**-2^{31} to $2^{31}-1$**
bigint	**8 bytes**	**-2^{63} to $2^{63}-1$**
uniqueidentifier	**16 bytes**	**98E94963-F193-4E69-9262-7B692125557F**
smalldatetime	**4 bytes**	**Jan 1, 1900 with precision to the minute**
datetime	**8 bytes**	**Jan 1, 1753 with precision to a timetick (3.33 ms)**
smallmoney	**4 bytes**	**- 214,748.3648 through +214,748.3647**
money	**8 bytes**	**-922,337,203,685,477.5808 through +922,337,203,685,477.5807**

SQL Server 2005 Data Types

- Large Value Data Types, n/varchar(max) and varbinary(max)
  - In row storage limit 8000 bytes
  - Work like typical varchar but support LOB values (231 bytes)
- Custom Types – CLR based
- Be aware that LOB types added to a table prevent online index rebuilds
- May want to consider vertical partitioning for special types

Character Data

- Unicode – nchar, nvarchar, ntext
  - Application (web data) is stored natively
  - Supports wide variety of client environments
  - Make sure to ALWAYS use 'N'
  - Examples:

```
SELECT * FROM tname WHERE col = N'value'
DECLARE @variable nvarchar(100)
SELECT @variable = N'value'
SELECT @Str = N'string' + N'string' + N'string'
```

  - CLR Extensibility requires unicode!
- ASCii – char, varchar, text
  - Think before using if new data/tables use Unicode instead

Nullability

- No specific value required at INSERT
- NULL values DO NOT mean empty space (stored separately from the column data)
- Working with NULLs
  - Accessing columns which allow NULL values can cause inconsistencies when developers/users are not aware of them
  - Math with NULL values can produce interesting results (? – NULL = NULL)
  - ANSI session settings can affect results sets when accessing columns that allow nulls
- NULL values v. DEFAULT value

Creating Tables

- Creating and Dropping a basic Table
- Determining Which Type of Constraint to Use
- Enforcing Data Integrity
- How SQL Server Organizes Data in Rows
- How SQL Server Organizes Rows on a Page
- Initial Table Structure with Clustered Index
- Intro to Vertical Partitioning
- Generating Scripts

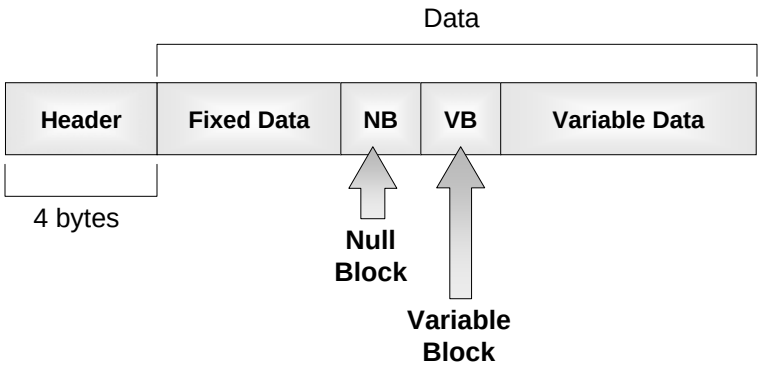
Creating and Dropping a Table

- Creating a Table

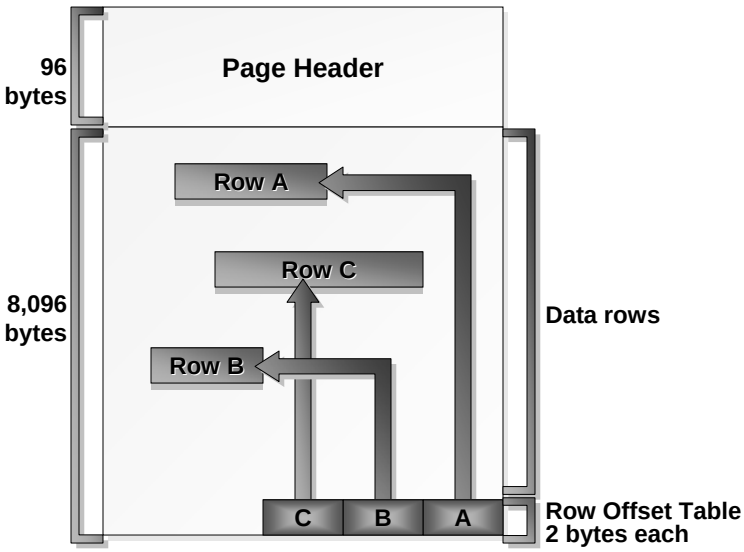
Column name	Data type	NULL or NOT NULL
CREATE TABLE dbo.publishers		
(pub_id	char(4)	NOT NULL,
pub_name	varchar(40)	NULL,
city	varchar(20)	NULL,
state	char(2)	NULL,
country	varchar(30)	NULL
)		

- Specifying NULL or NOT NULL
  - Not required
  - Defaults to session setting
    - SET ANSI_NULL_DFLT_ON ON

The Data Row



Rows on a Page



Optimal Row Width

- Consider Table Usage above all else
- Estimate Average Row Length
  - Overhead
  - Fixed Width Columns
  - Estimate Average from realistic sample data
    SELECT avg(datalength(columnname)) FROM tname
- Calculate Rows/Page

$$\frac{\text{8096 Bytes/Page}}{\text{??? Bytes/Row}} = \textbf{Rows/Page}$$

- Calculate Wasted Bytes—on Disk ∴ in Memory

Row Size/Table Structure

- Rows CAN span pages in SQL Server 2005
- Is that always the best choice?
- Table access/usage patterns should dictate structure
- Keep active/OLTP columns in primary table
- Separate LOB types to secondary table for online table/index rebuilds
- Create initial table structure to minimize the need for rebuilds (*Designing for Performance – more info coming up!*)

Design Techniques Summary

- Rows CAN span pages however, record size impacts disk space, cache utilization, concurrency and maintenance restrictions
- The lowest granularity of locking holds an entire row – partitioning a table vertically or horizontally can improve concurrency, minimize restrictions on online rebuilds and improve cache utilization
- Optimizer is aware of constraints – efficient horizontal partitioning requires constraints
- Duplicating Keys which define relationships can improve join performance by giving the optimizer more options for performing a join

Key Points

- Minimize data redundancy by using attributes specific only to the entity table definition
- Specify Appropriate Data Types and Data Type Sizes – try to keep row sizes compact
- Always Specify Column Characteristics in CREATE TABLE – especially column Nullability
- Generate Scripts as backup scripts and for learning
- Take into account the usage of the data when defining tables, consider partitioning tables based on usage
- Define Constraints on your tables – for performance as well as centralization

Why partition?

- Resource Contention Limitations
- Varying Access patterns
- Maintenance Restrictions
- Availability Requirements
- To remove resource blocking or minimize maintenance costs
- But:
  - Partitioning does not automatically mean Distributed Partitioned Views (DPVs)
  - DPVs are a form of scale-out partitioning

Table's Shared Data/Indexes

- A Single Large Table…
  - Presents different types of problems
    - Management
    - Index create/build or rebuilds
    - Backup/Restore and Recovery
    - Escalation
  - May have different access patterns
    - Some data new – OLTP (inserts/updates for new and current sales)
    - Some data old for historical lookups – small singleton for OTLP lookups or larger for analysis
- Does NOT have to be one table!

Indexes – Smaller Tables

- Index Creation/Building
  - Smaller Tables – take less time
  - Different index decisions
    - OLTP – less indexes
    - DSS – more indexes
- Index Maintenance
  - Smaller Tables – takes less time to rebuild/defrag
  - Different Access patterns – require less frequent maintenance or NONE
    - OLTP – rebuild/defrag more frequently
    - DSS – build once!

Backup Strategies Vary

- Transaction Processing
  - Backups (should be) more frequent
  - More critical data
  - Should be highly available
  - Usually smaller portion of VLDB
- Decision Support
  - Backups needed less frequently
  - Less critical data
  - Should be available but not “as” critical
  - Usually a larger portion of VLDB

Availability

- SQL Server 2000
  - Database online or not
  - Entire database offline, if damaged…
- SQL Server 2005
  - Database has an associated “state”
    - Online, restoring, recovering, …
    - Database in only one state at any given time
  - Files/Filegroups also have states
    - Secondary filegroups have independent state
    - Filegroups can be offline and/or recovering while remainder of database is available
  - State of the database
    - State of the primary filegroup
    - State of the transaction log

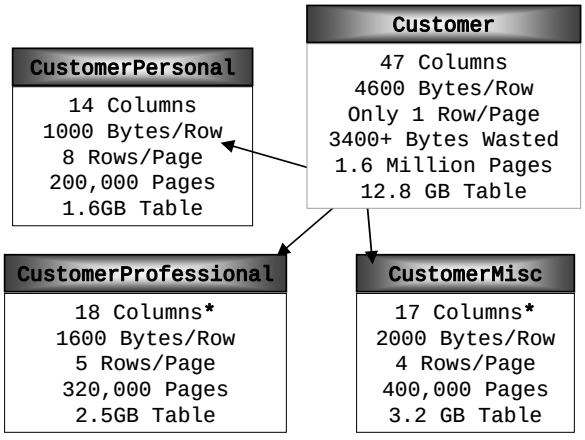
Access Patterns

- Transaction Processing
  - Smaller INSERT/UPDATE/DELETE activity
  - Speed/durability of highest importance
- Decision Support
  - Larger Range Queries
  - Less critical data
  - More indexes
  - Query access patterns may conflict with write activity

Partitioning Strategies

- Vertical Partitioning
  - Multiple tables – "subsets" of columns
  - Different column sets
    - Separate sets based on usage patterns
    - Primary key is redundant, possibly other columns
- Horizontal Range Partitioning
  - Multiple tables – all columns
  - Different row sets
    - Separate sets based on date range
    - Separate sets based on lists – also "range" partitions

Consider Customer Table with 1,600,000 Rows



Customer
= 12.8 GB

Partitioned Tables
= 7.3 GB

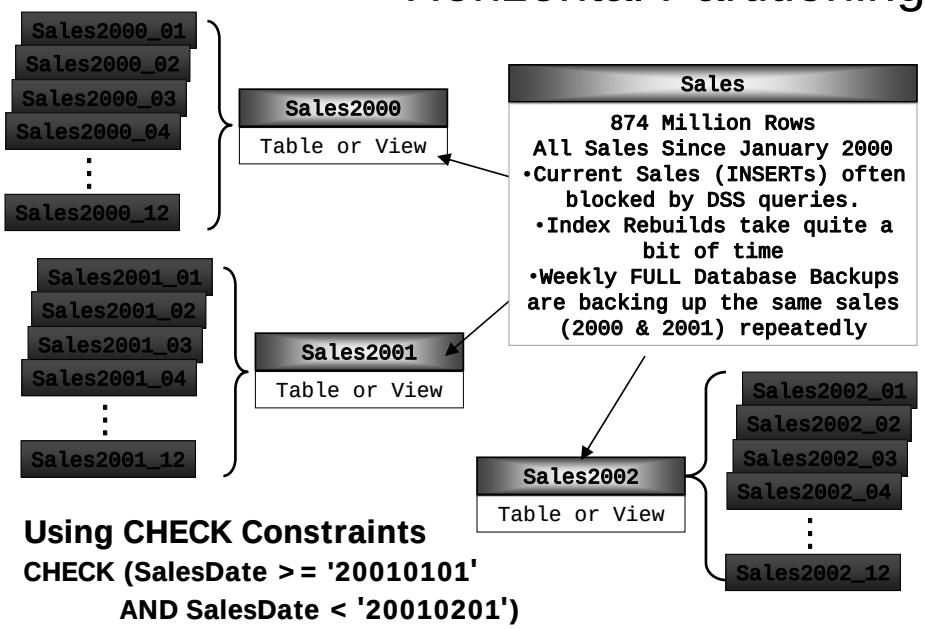
- Savings in Overall Disk Space (5.5 GB Saved)
- Not reading data into cache when not necessary
- Locks are table specific therefore less contention at the row level

***The Primary key column must be made redundant for these two additional tables. 47 Columns in Employee. 49 Columns total between 3 tables.**

Vertical Partitioning

- Optimizing row size for…
  - Caching
    - More rows on a page, more rows in memory
  - Locking
    - Only locking the columns that are of interest. Minimize row based conflicts
- Usage Defines Vertical Partitions
  - Logically Group Columns to minimize joins
  - Consider Read Only vs. OLTP Columns
  - Consider Columns often used together

Horizontal Partitioning



Using CHECK Constraints

```
CHECK (SalesDate >= '20010101'
       AND SalesDate < '20010201')
```

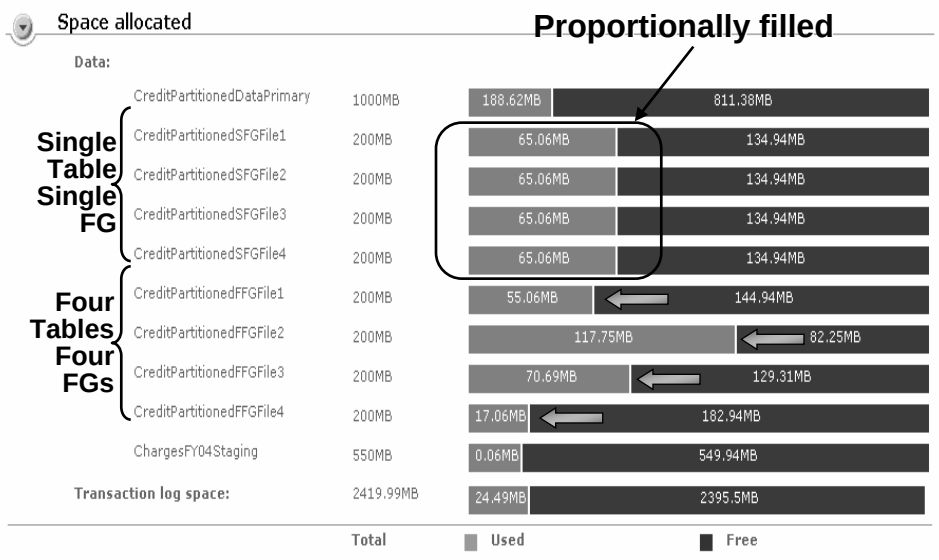
Horizontal Partitioning

- More manageable chunks
- Minimize the impact of maintenance
  - Improve loading options for sliding range scenario
  - Index Maintenance (Smaller/Less Frequent)
  - Backup/Restore (File/Filegroup strategies to minimize backup frequency of predominantly ReadOnly tables – more options for restore when isolated disk/RAID array corruption)
  - Lock Escalation (Modifications to one partition do not cause escalation against the others – as they likely would if everything were in one table!)

"Partitioning" Strategies

- Proportional Fill using Filegroups
- One table on one filegroup with four files
  - Easy to administer
  - Data is proportionally filled among four files
  - Data on each file is not "similar" in the sense that it's not 1st Q, 2nd Q, etc.
  - All Objects and all dates are interleaved

First Glance…



Functional Partitioning Strategies

Data-based partitioning

- SQL Server 2000 – Partitioned Views
  - Four tables created on four separate filegroups
  - More complex to administer/create
  - Data is NOT proportionally filled
  - Data on each file IS “similar” as the data is filled based on date – i.e. 1st Q, 2nd Q, etc.
  - The same quarter’s dates are on the same file (index and table are storage aligned)
- SQL Server 2005 – Partitioned Tables
  - One table created on a partition scheme
  - Data placement and breakdown is same as PVs

Functional Partitioning

- **More granular control** – both 2000/2005
  - Varying access patterns
  - Varying index defrag patterns
  - Very fast sliding window control (*more details next!*)
- **More backup/restore choices** – in 2000 but many restrictions removed in 2005
  - File/filegroup backups
  - Even support for file/filegroup in Simple Recovery Model (backups separated between RW and RO backups and RO backups can be performed at any frequency!)
- **Higher Availability** – in 2005 because of partial database availability features

VLDB – Sales Reporting

- Active Sales in OLTP system
- Secondary VLDB used for reporting
- Range of two year's data
- Monthly the active OLTP data is copied over and the oldest month is backed up and archived

On October 1, 2004

Rolling Range (or Sliding Window)

Key Components

- Data Load
  - Single Table
    - Active Table impacted
    - Indexes need to be updated
  - Partitioned Object (PV in 2000/PT in 2005)
    - Table outside of active view manipulated
    - Indexes can be built separately of active tables
- Data Removal
  - Single Table – same problem
    - Active Table impacted
    - Indexes need to be updated
  - Partitioned Object (PV in 2000/PT in 2005)
    - Table can be removed from PO and then dropped

Total Times

"Single" table process: **40+ minutes**

1. Alter the table's check constraint to support new quarter (drop the old constraint)
2. Load the data using BULK INSERT
3. Delete the first quarter of F04
4. Alter the table's check constraint to show new year range (drop the old constraint)

Partitioned table process: **1 min 36 sec**

1. Create New table - with constraints
2. Load the data using BULK INSERT
3. Build Indexes – CL + 3 NC on FYQ1 Filegroup
4. Change the view to use FY05Q1 and remove FY04Q1
5. Drop the table which represents FY04Q1 (could archived/backup)

Range Partitioned Tables

- Scripts from Whitepaper on Partitioned Tables
- Step 1: Create Filegroups
- Step 2: Create Files in Filegroups
- Step 3: Create Partition Function
- Step 4: Create Partition Scheme
- Step 5: Create Table(s) on Scheme
- Step 6: Verify Data using system table (optional)
- Step 7: Add data to tables – SQL Server redirects data and queries to appropriate partition

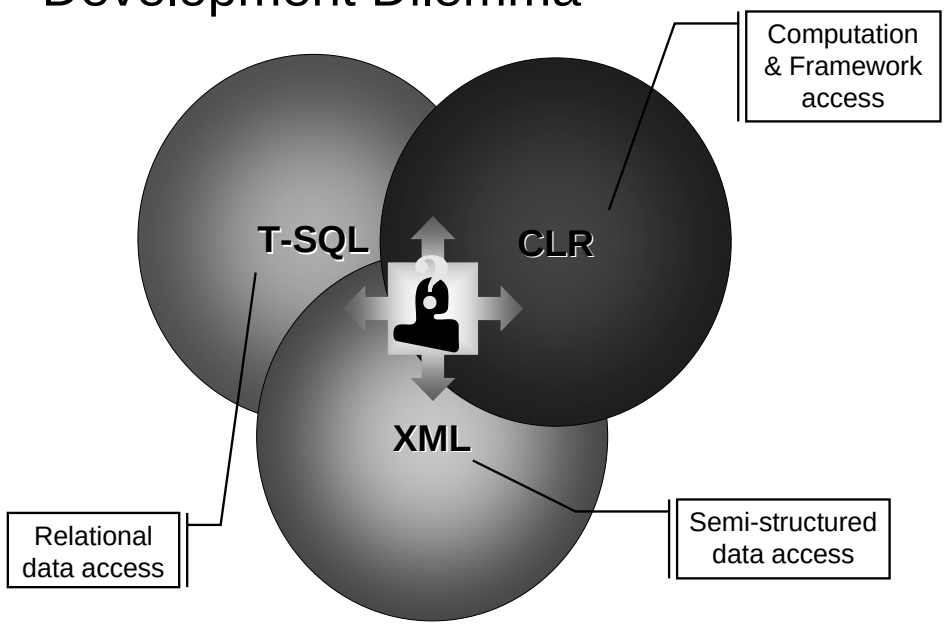
The Sliding Window Scenario

- New Data must come in
- Old data must be removed
- The table should only change it's overall data set… with simple metadata changes
- To create metadata ONLY changes
  - Create a new table on the same filegroup where the partition resides
  - Structure it IDENTICALLY to that of the partitioned table
  - SWITCH the partition to live in the new table and the table to become the partition… this either brings data IN or OUT of the table

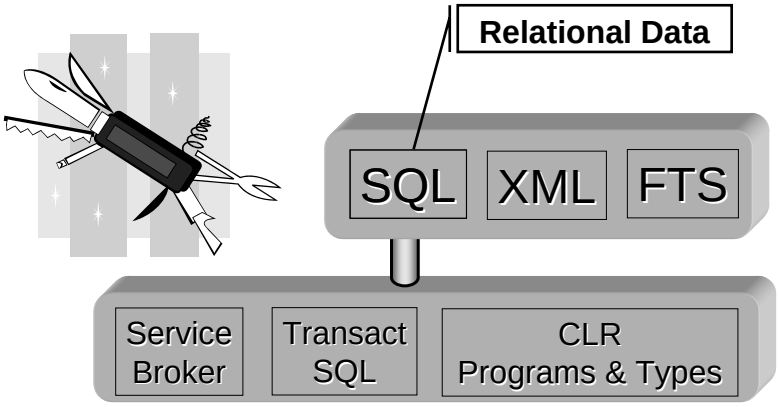
Developing on SQL Server 2005

- Choices, Choices and More Choices
- Secure Development
- Performance and Scalability

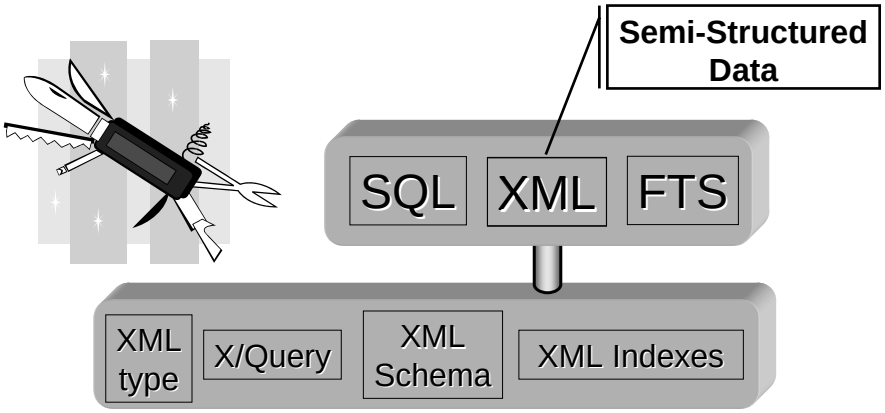
Development Dilemma



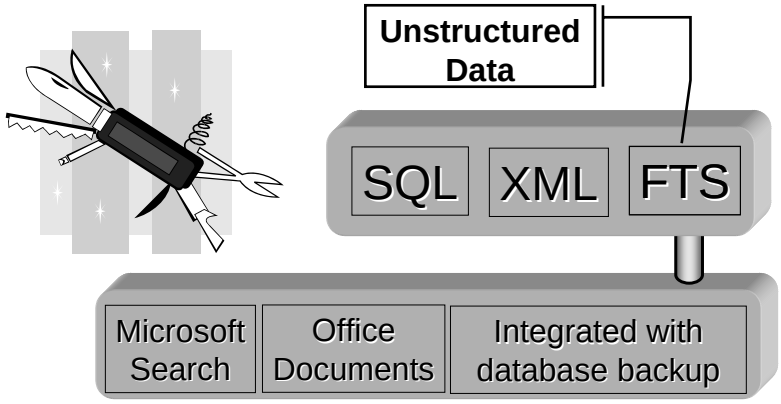
SQL Server 2005 Toolkit



SQL Server 2005 Toolkit



SQL Server 2005 Toolkit



Good Scenario for CLR Usage

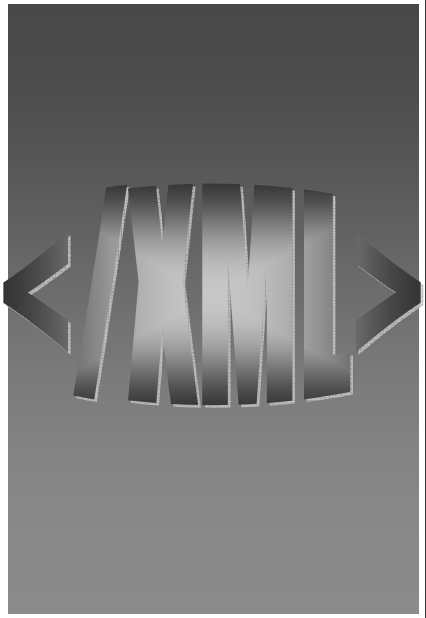
- Data validation & network traffic reduction
- Writing general purpose functions:
  - Data passed as arguments
  - Little/no additional data access
  - Complex computation applied on a row by row basis to the data
- Scalar types & custom aggregations
- Leveraging the power of the .NET Framework
  - Access to a rich set of pre-built functionality
- Replacing Extended Stored Procedures (XP)
  - The CLR is safer:
    - No access violations making SQL Server crash
    - No leaks making SQL Server slow down & crash
    - Better performance & scalability (managed memory model)
    - No security issues…

Bad Scenario for CLR Usage

- Heavy data access – Transact-SQL set based access will be faster
  - Don't write SELECT statements as CLR procedures!
- Complex types
  - 8K size limitation
  - All data is read/re-written when updated
- Pre-Aggregation for Reports
  - CLR Aggregates cannot be used in Indexed Views
- Your application must support previous versions of SQL Server
- Technology for technology's sake…

XML Development

- Native XML datatype
  - Optimized on-disk structure (not "just a blob")
- XML Indexes
  - Entities & Attributes
- X/PATH Query Engine
  - Integrated with core relational operators
- Schema Support
  - Strongly typed XML
- CLR/XML Integration
  - XMLTextReader
  - Access to XSLT
- Full Text Search



XML Indexes

- Primary Index:
  - Requires a clustered primary key on the base table
  - B-Tree of element and attribute names, node values, and node types, retains document order and structure, as well as the path from the root of the XML instance to each node for efficient evaluation of path expressions
- Secondary Indexes off of the primary XML index
  - PATH ➔ columns (path, value)
  - PROPERTY ➔ columns (PK, path, value)
  - VALUE ➔ columns (value, path)

Scenario for XML Development

Good Scenario:

- Data is semi-structured, small core of fixed data with many, sparsely populated extended attributes
  - Multi-value Property bags
  - Complex Property bags
  - "WordXML"
- Documents are large but rarely updated
  → Indexing will pay off
- Data is hierarchical
  → regular expressions are well suited for finding data

Bad Scenario:

- "Database in a Cell"
- Documents are large and updated frequently
- Document update contention is likely
- Data is fully structured & populated → candidate for conversion to relational schema
- Data contains large binary objects (2GB limitation)

Transact-SQL Enhancements

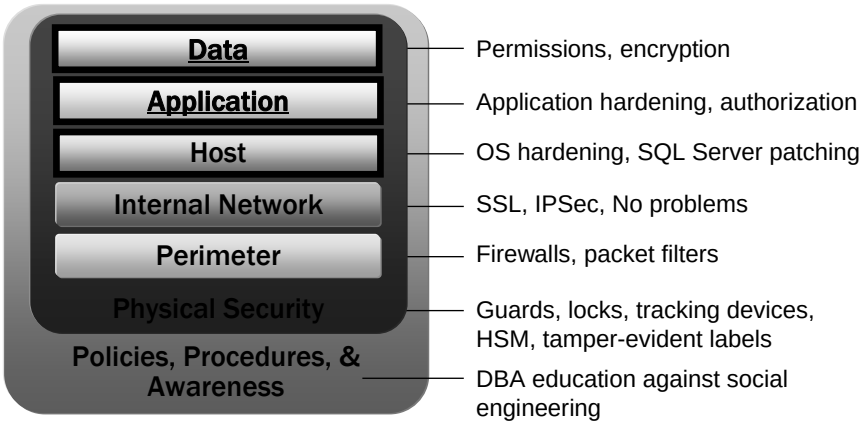
- ROW_NUMBER
- RANK, DENSE_RANK
- Common Table Expressions
- PIVOT/UNPIVOT
- CROSS APPLY and OUTER APPLY
- Error handling – TRY/CATCH
- DDL Triggers (synchronous)
- Event Notifications (asynchronous)
- Parameterized TOP

Whose Fault was SQL Injection?

- OK. We got your attention. It was not you anyway, of course.
- Security for a database developer was a very rude wake-up call.
- We *feel* your pain.
- We can help you – in this session!

Defense in Depth

- Using a layered approach:
  - Increases an attacker's risk of detection
  - Reduces an attacker's probability of success



Application Protection

- *Protection against attacks against the application*
- Techniques and practices used by the database code (modules, stored procedures) to avoid being exploited by callers
  - Permissions and ACLs on the data are also part of application protection from the perspective of the calling application

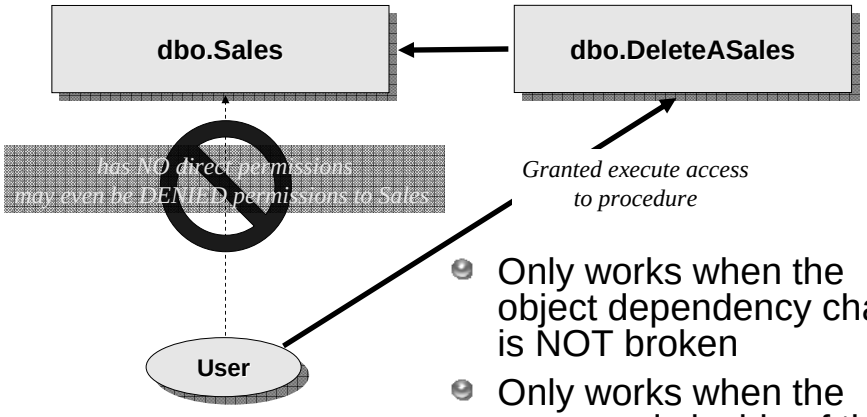
Secure Development

- How things used to work?
- User-schema separation
- Granular permissions
- EXECUTE AS
- SQL Injection (Classic)
- SQL Injection (Future)

Security through Encapsulation

- Granting access to a process or a method without direct access to base objects
- How?
  - Grant access to the stored procedures, views and/or functions without granting access to the base object
- Requires
  - The objects in the dependency chain cannot be broken (object ownership chaining)
  - The user has to have direct base object access (which is what you're trying to avoid)

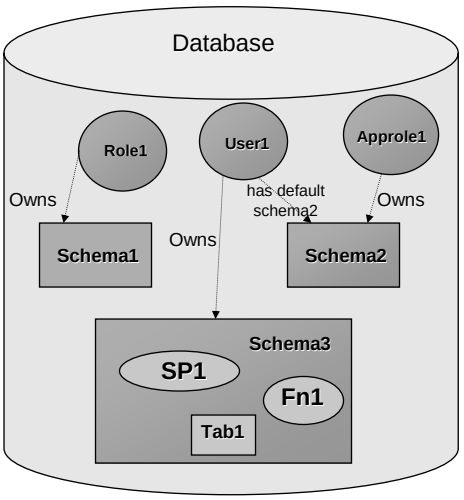
Do NOT Give Base Object Access
The SQL Server 2000 Way



- Only works when the object dependency chain is NOT broken
- Only works when the commands inside of the stored procedure are not built/executed dynamically

User-Schema Separation

- Database can contain multiple schemas
- Each schema has an owning principal – user or role
- Each user has a default schema for name resolution
- Database objects live in schemas
- Object creation inside schema requires CREATE permission and ALTER or CONTROL permission on the schema
- Ownership chaining still based on owners not schemas



Default Schema

- Used for name resolution purposes
- Common name resolution across multiple users
- No need to rely on DBO schema
- Using DBO schema may result in security issues
  - Object Creation requires higher privileges
  - Mitigates concerns resulting from ownership chaining
- Instead create “buckets” of objects through “schemas” where schemas have owners and developers have default schemas and/or control on needed schemas

Do NOT Give Base Object Access

The SQL Server 2005 Way

- Create a schema – for example one which contains procedures (and possibly even base tables) and then GRANT EXECUTE at the schema level
- Add objects to appropriate schema
- Grant access to the schema or the individual objects – if chaining is required, it is still based on the owner… the owner of the schema

What if Dynamic String Execution

- By default – and for better security – if the stored procedure has a statement which is built dynamically (using EXEC('string') or EXEC(@variable)) then the context under which the dynamically constructed string executes is ALWAYS the caller
- Which is what helps to prevent some forms of SQL Injection
- This is really a **good** thing BUT…
- Can be limiting
  - Enter: *EXECUTE AS*

Module Execution Context

- Execute AS CALLER
  - Default behavior, same as SQL Server 2000
  - Use when caller's permission needs to be checked – or when ownership chaining will suffice
- Execute AS 'UserName'
  - Statements execute as the username specified
  - Impersonate permission required on user specified
- Execute AS OWNER
  - Statements execute as the current owner of the module
  - Impersonate privileges on owner required, at setting time
  - On ownership change, context is new owner
- Execute AS SELF
  - Statements execute as the person specifying the execute as clause for the module – even if the ownership changes
  - May be useful in application scenarios where calling context may change

EXECUTE AS…

- Solves problems
  - Allows permissions to be granted where never possible (e.g. granting truncate table)
  - Wrap ANYTHING inside a stored procedure and set the context to run as someone who has permissions – even dynamic string execution – then give execute permission
- Creates potential for further SQL Injection
  - What if you're code is not well tested and uses dynamically executed strings

SQL Injection

- SQL statement(s) or clause(s) injected into an existing SQL command
- Injected string appended to application input
  - Text boxes
  - Query strings
  - Manipulated values in HTML
- Why SQL injection works?
  - Application accepts arbitrary user input
  - Connection made in context of higher privileged account

SQL Injection Mitigation

- Do not trust the user's input!
  - Look for valid input and reject everything else
  - Protect identifiers with QUOTENAME()
  - Regular expressions are your friend!
- Do not use string concatenation
  - Use parameterized queries to build queries
- Restrict information in error messages
- Use a low-privileged account
  - DO NOT use sa or sysadmin role member

Data Protection

- Use of cryptography to protect the data layer from theft or manipulation
  - Particularly important if offline data is in transit
  - Important for regulatory reasons
    - Prevent admin from access
    - Sarbanes Oxley
- Manipulation (integrity) protection uses digital signatures
  - Not implemented for data in SQL Server 2005
    - Can overcome this with judicious use of encryption alone
  - Implemented for code signing

Check Your OS

- Verify what algorithms and key lengths are supported by your OS, as this depends on your CSP (Cryptographic Services Provider)
  - Generate keys for all algorithms then look in the sys.symmetric_keys
- For example, we found the following during testing – this may change in the future
- At present, there is no way to change the CSP…

	XP SP2	WS2003
DES	56 (64)	56 (64)
3DES	128	128
AES128	-	128
AES192	-	192
AES256	-	256
RC2	128	128
RC4	40	40

Before You Begin

- Your DBA must ensure that the server has a Service Master Key (SMK) that agrees with your disaster recovery strategy and a Database Master Key (DMK) has been created for each database that will use encryption
  - There are serious implications on security (use strong password for SQL service account) and for disaster recovery
    - ALTER SERVICE MASTER KEY can be used to change recovery options
  - CREATE MASTER KEY ENCRYPTION BY PASSWORD =
    - Use a strong password, write it down, keep it safe

Key Generation

- The key should be impossible to guess. Preferably random.
  - CREATE SYMMETRIC… will generate a fairly random key for you – good!
  - You can base the key on data supplied by you, use KEY_SOURCE* clause – good for generating identical keys from a high-quality password
  - Note: KEY_SOURCE may be renamed to DERIVED_FROM in the released version

Passwords

- Make sure passwords used to protect or create keys are very strong
- In our opinion, it is better to create a very long and complex password that you will have to write down and store in a well-protected safe in your company
  - You can divide it into two halves and store in separate envelopes in different safes
- E.g.: *87(HyfdlkRM?_764#{(**%GRtj*(NS£”_+^$(
  - No dictionary words, more than 20 characters, many non-printing characters
  - Challenge: usability!
- Consider also:
  - Not keeping passwords as text in your code, but store and retrieve them through DPAPI and a .NET component
  - Using good quality password generators

Key Protection

- SQL Server 2005 insists that the key you create is further encrypted for its protection
  - Yes, that's double-encryption, but that is not double-security. Actually, it reduces security a little in some cases
- CREATE SYMMETRIC…ENCRYPT BY
  - PASSWORD
    - Your (v. good) password generates a key to 3DES encrypt the key you are protecting
    - Note, 3DES is less secure than AES, so this
  - CERTIFICATE
    - Your key is encrypted using the public key of a certificate
    - This, in essence, is hybrid encryption
    - If private key is kept secure (and offline), this is a very good way to protect a symmetric key
  - Or another SYMMETRIC or ASYMMETRIC key – less useful but interesting

Encryption

1. Create or retrieve the key
2. Open the key – this means decrypt it with its (secondary) password or certificate or other key
3. Use function ENCRYPTBYKEY inside DML

Decryption

1. Create or retrieve the key
2. Open the key
3. Use function DECRYPTBYKEY inside SELECT and all other DML

Performance Issues

- As you know, security is expensive
- *Asymmetric* encryption alone is not viable for large amounts of data
- *Symmetric* cryptography can be both strong and fast: AES
- *Hybrid* encryption is fast enough and solves the problem of symmetric key distribution logistics
- Don't encrypt everything, this rarely makes sense and would be very costly

- But, our (suspect) performance tests showed that all algorithms were as fast as each other. This is not what we expected, and either signifies other overheads which cancel out the differences, or we need to re-test on a released product (RTM).

Top 10 Best Practices in Security

1. Risk assessment and threat analysis
2. Trap “developer quality” error messages – avoid indecent disclosure
3. Avoid using dynamic string execution
4. Think encryption for sensitive data
5. Do not create objects in DBO schema
6. Hide underlying application schema
7. Use Roles for managing permissions
8. Account for catalog security
9. Avoid encryption as it precludes indexing and performance
10. Know your CSP. Check allowed key lengths if needed. Chose carefully!

Performance and Scalability

- Using Indexes
- Index Concepts
- Table and Index Internals
- Initial Table Design
  - The Clustered Index Debate
- Secondary Non-clustered Indexes
- Letting ITW/DTA Help!

Index Usage
Conceptual

- Allows faster access to data, improving:
  - Lookup/query time
  - Insert/Update time – record location defined
  - Delete time – record location also defined
- Con: overhead for modifications
- Index Strategy Concepts
  - Fewer indexes are better than lots of indexes
  - Wider indexes have more uses
    - Can be used for point queries
    - Can be used by NARROW low selectivity queries
- Write Effective Queries to better take advantage of indexes…

Accessing Data: Limit Data

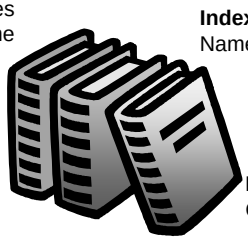
- Subset of columns = Projection
  - Do not use * (unless against view)
  - Optimizer has more chances for optimizing query when result set is NARROW (only the required columns)
- Subset of rows = Selection
  - Use positive search arguments
  - Isolate the column to one side of the expression
    - USE: MonthlySalary > value/12 (constant, seekable)
    - DO NOT USE: MonthlySalary * 12 > value (must scan)
  - Be cautious with LEADING wildcards
    - USE: LastName LIKE 'S%'
    - Try not to just append %val% to every application
- Consider using Views, Stored Procedures and Functions to limit the columns/rows

Index Concepts
Book analogy

- Think of a book—with indexes in the back
- The book has one form of logical ordering
- For references, you use the indexes in the back…to find the data in which you are interested you look up the key
- When you find the key, you must lookup the data based on its location… i.e., a "bookmark" lookup
- The bookmark always depends on the (book) content order

Index – Species Common Name

Index – Animals by Habitat, Name *Air, Land, Water*



Index – Animal by Type, Name *Bird, Mammal, Reptile, etc…*

Index – Species Scientific Name

Index – Animal by Country, Name

Index – Animal by Continent, Country, Name

Index Concepts
Tree Analogy

- If a tree were data and you were looking for leaves with a certain property, you would have two options to find that data:
- 1) Touch every leaf – interrogating each one to determine if they held that property…SCAN
- 2) If those leaves (which had that property) were grouped such that you could start at the root, move to the branch and then directly to those leaves…SEEK

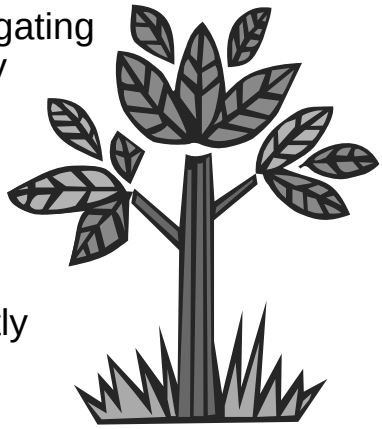


Table Structure

- HEAP: A table without a clustered index
- Clustered Table: A table with a clustered index
- Non-clustered Indexes *do not* affect the base table's structure
- However, Non-clustered Indexes are affected by whether or not the table is Clustered…

Hint: The non-clustered index dependency on the clustered index should impact your choice for the clustering key!

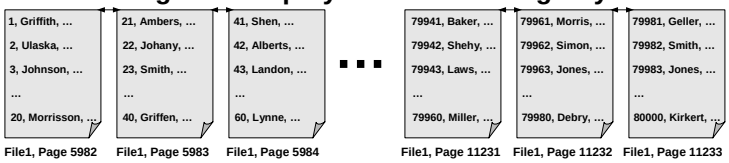
Heap: Pros

- Effective for Staging Data
- Indexes can be created after load and index creation can be parallelized
- Efficient for SCANs ONLY – when no UPDATEs (otherwise, forwarding pointers – scans become significantly less efficient)
- Space Efficient – Space from Deletes is reused on subsequent Inserts (at the cost of performance)
- Best used when table is not typical OLTP or combo OLTP/DSS table
- Forward only "logging" table

Table Structure
Clustered table

- Clustered Index defines order – applied at CREATION
- Expensive—in time and space—to build (an additional 1-2 times the index size for (re)build)
- Table is a doubly-linked list – order maintained LOGICALLY
- *Might* be expensive to maintain

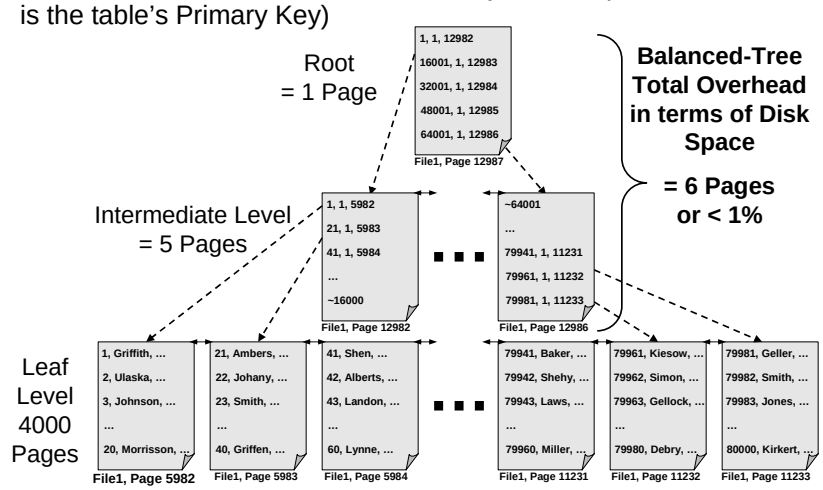
4000 Pages of Employees in Clustering Key Order



Clustered EmployeeID

Employee Table

Consider a clustered index on an identity column (even better if this is the table's Primary Key)

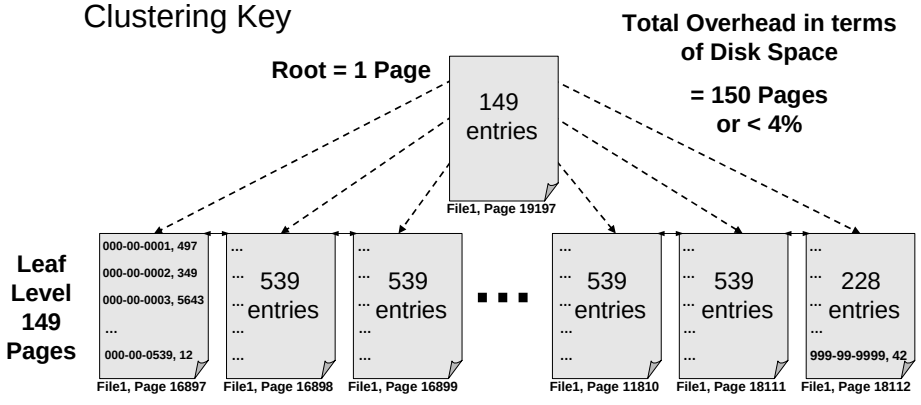


Non-Clustered Indexes
Physical

- Depend on whether the table is a Heap or Clustered
- Clustered Table
  - Rows use the Clustering Key
  - No additional OH to add this column – may use actual data to define the clustering key
  - Minimizes NC Index manipulation if rows move (the NC still points to the CL Key)
  - Clustering key should be static and narrow
- Heap
  - Generally, not recommended

Non-Clustered Index
Unique Key SSN

- Leaf level contains the non-clustered key(s) to define the order
- Also includes either the Heap's Fixed RID or the Table's Clustering Key



Clustered Index Criteria

- Unique
  - Yes – No overhead, data takes care of this criteria
  - NO – SQL Server must "uniquify" the rows on INSERT. This costs time and space. Each duplicate has a 4-byte uniquifier. In SQL Server 2000, all uniquifiers are regenerated when the CL index is rebuild. This is not true in SQL Server 2005.
- Narrow
  - Yes – Keeps the NC indexes narrow
  - NO – Possibly wastes space
- Static
  - Yes – Improves Performance
  - NO – Costly to maintain during updates to the key especially if row movement and/or splits

In fact, an identity column that's ever-increasing is ideal…

Clustering on an Identity

- Naturally Unique
  (although not guaranteed – should combine with key constraint)
- Naturally Static
  (although should be enforced through permissions and/or trigger)
- Naturally Narrow
  (only numeric values possible, whole numbers scale = 0)
- Naturally creates a hot spot…
  - Needed pages for INSERT already in cache
  - Minimizes cache requirements
  - Helps reduce fragmentation due to INSERTs
  - Helps improve availability by naturally needing less defrag

Clustering Key Choices

- Identity column
- Date, identity
  - Not date alone as not unique
- GUID
  - Populated by client-side call to .NET client GUID generator (NOT a good CL key but ok as NC Primary Key)
  - Populated by server-side newid() function (no pattern, not ever-increasing)
  - Populated by NEW server-side newsequentialid() function (Creates an ever-increasing GUID)

Key points: Narrow, static, unique, ever-increasing

Key Constraints Create Indexes

- Primary Key Constraint
  - Defaults to Unique Clustered
  - Only One Per Table

```
ALTER TABLE Employee
  ADD CONSTRAINT EmployeePK
    PRIMARY KEY CLUSTERED (EmployeeID)
```

- Unique Key Constraints
  - Default to Unique Non-Clustered
  - Up to 249 Per Table

```
ALTER TABLE Employee
  ADD CONSTRAINT EmployeeSSNUK
    UNIQUE NONCLUSTERED (SSN)
```

Optimal Table Structures
Poor clustered index choice

- LastName or non-sequential GUID
  - Fragmentation ⇨ poor performance
  - Non-unique ⇨ poor performance
    - Uniquifier wastes space and time
    - SQL 2000 only, rebuilding a clustered index on a non-unique clustered index forces non-clustered indexes to be rebuilt
  - Volatile ⇨ poor performance
    - Most-duplicated value
    - Could cause record relocation and therefore fragmentation
  - Wide ⇨ poor performance
    - Wastes space/time – wide keys take longer to maintain/insert.
    - The wider the key the wider the non-clustered indexes
- TIP: Don't do this! ☺

Optimal Table Structures (cont'd)
Heap is better

- No clustered Index
  - No data row Fragmentation → forwarding rows
  - Could have LOB fragmentation (new LOB_Compaction can help this type of fragmentation)
    - Wastes space in non-clustered indexes by using fixed-RID
    - Scans (due to locking/consistency requirements) can be more expensive!
    - Optimized for space over speed
- TIP: Create HEAPs for staging tables as an interim table for high performance data loading!
- See presentation "High Performance Data Loading" on www.SQLDev.Net for more tips!

Optimal Table Structures (cont'd)
The **RIGHT** Clustered Index

- The RIGHT Clustered Index
  - Cluster on identity column or possibly add one for clustering
  - Cluster on composite key (date, identity) for tables that also have this pattern
  - Key Criteria: Static, Narrow and Unique
  - Effective Hot Spot(s) using Identity (or date, id) or composite key to create multiple (but not random) hot spots – possibly consider partitioning for large tables/high volume OLTP

TIP: If this doesn't exist, add a surrogate column solely to cluster on it!

TIP: Make sure to find the RIGHT Clustered index for *your* environment

TIP: Make sure to AUTOMATE defragmentation if/when necessary

Clustering for Performance

- Cluster for internal benefits which lead to:
  - Better insert/update performance re: fewer splits
  - Better Availability as full table maintenance may not be needed
- Reliance on Nonclustered indexes is greater:
  - Faster access to narrow/low selectivity range queries
  - NC Indexes are used in numerous non-obvious ways (multiple NC Indexes can be joined to cover a query)

Reliance on Nonclustered Indexes…

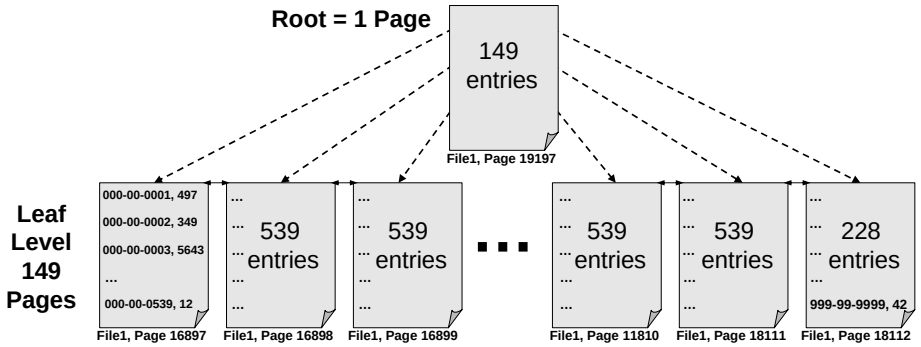
- Faster access to narrow/low selectivity range queries
- NC Indexes are used in numerous non-obvious ways (multiple NC Indexes can be joined to cover a query)
- More flexible in the definition (i.e., Indexed Views can include computations, substrings, etc.)
- New INCLUDE clause can allow wider covering indexes!
- Non-clustered indexes are easier/faster to rebuild (re: smaller) when they become fragmented
- Non-clustered indexes are easier to keep less fragmented – i.e., rebuilds with FILLFACTOR has greater benefit since the index row is narrower

Non-Clustered Indexes

- Internally, non-clustered indexes always store an entry for EVERY row of the table
- If the columns requested in the query are IN the index (regardless of order) then the index COVERs the query
- If queries are narrow this is more likely to happen
- Real-world queries…for example, accessing a large number of tables in a join are typically narrow (the join condition, maybe a search argument and possibly one or two columns in the select list)
- Is it likely that you would ever execute:

SELECT * FROM 8-table-join

What kind of structure is this?



- A non-clustered index on SSN on a table that has a clustered index on EmpID (as we've seen)
- A clustered in on SSN n a table that has ONLY two columns, SSN and EmpID

NOTE: A non-clustered index is EXACTLY the same as a clustered index – but only on the columns that make up the NC index (with the RID or CL Key).

Seems Like…

- Non-clustered indexes are only useful for relatively selective queries…
- What about the following queries?

```
SELECT EmpID, SSN
FROM Employee
WHERE SSN between x and y
```

Think about the access patterns:
- Table Scan (always an option)
- Index Seek w/partial scan
  The nc index on SSN "covers" the query

```
SELECT EmpID, SSN
FROM Employee
WHERE EmpID < 10000
```

Think about the access patterns:
- SARG is on clustering key

  The nc index on SSN "covers" the query

Finding the Right Balance

- Start with a minimal number of indexes
  - Clustered Index
  - Primary Key (nonclustered, if not clustered)
  - Unique Keys
- Manually index foreign keys
  - Non-unique indexes
  - Speed up join performance
- Use Database Tuning Advisor
- Manually index based on either:
  - Specific query tuning where DTA didn't help
  - Query frequency

Non-Clustered Indexes
For optimal query performance

- Non-Clustered Indexes are BETTER for low selectivity/range queries because:
  - Non-clustered indexes ARE EXACTLY the same as a clustered index—but narrower
  - If your query only needs columns IN an index then the index is said to "cover" your query
  - No – this does *not* mean that you need to cover every query!!!
  - What it does mean: You can make one or two existing indexes slightly wider and you probably will not hurt INSERT, UPDATE, and DELETE performance and may RADICALLY improve SELECT performance
- Non-Clustered Indexes are narrower and easier to maintain (let ITW/DTA help…)
- Use INCLUDE to create narrower trees, more relevant leaf levels for covering!

Covering an Aggregate

- Increasing Gains with different types of indexes can always be achieved
- Each query can be isolated and tuned as if it were in a sandbox – not the goal
- Finding the right balance starts with:
  - Base Indexes (CL, PK, UK, FKs)
  - Capturing a workload, tuning with DTA
  - Prioritizing the queries that are still performing poorly and then choose the optimal level of gain by understanding the trade-offs

Indexing for Aggregations

- Two types of Aggregates:
  Stream and Hash
- Try to Achieve Stream to Minimize Overhead in temp table creation
- Computation of the Aggregate Still Required
- Lots of Users, Contention and/or Minimal Cache can Aggravate the problem!

Aggregate Query

- Member has 10,000 Rows
- Charge has 1,600,000 Rows

```
SELECT c.member_no AS MemberNo,
      sum(c.charge_amt) AS TotalSales
FROM dbo.charge AS c
GROUP BY c.member_no
```

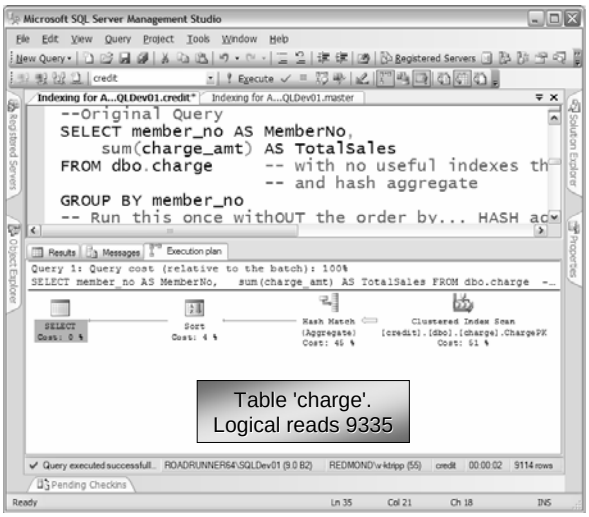

Aggregate Query (cont'd)
Table scan + hash aggregate

```
SELECT c.member_no AS MemberNo,
      sum(c.charge_amt) AS TotalSales
FROM dbo.charge AS c
GROUP BY c.member_no
```

- Table Scan of Charge Table
  - Largest structure to evaluate
  - Worst case scenario
- Worktable created to store intermediate aggregated results – OUT OF ORDER (HASH)
- Data Returned OUT OF ORDER – unless ORDER BY added
- Additional ORDER BY causes another step for SORT – sorting can be expensive!

Worst Case

- Clustered Index Scan
  (table scan)
  1,600,000 rows
- Hash Aggregate
  yields 9,114 rows
  out of order
- Sort
  only has to sort
  9,114 rows
  instead of
  1,600,000 rows
- Return Data



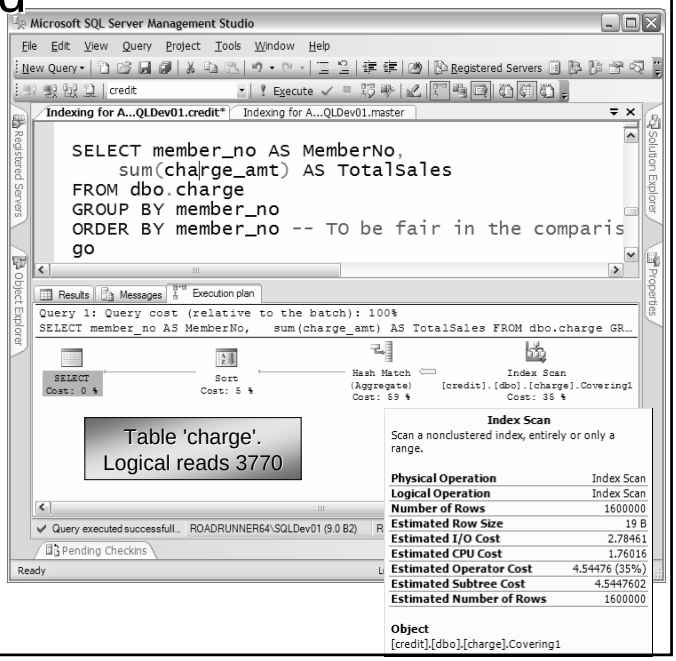
Aggregate Query
Index scan + hash aggregate

```
SELECT c.member_no AS MemberNo,
       sum(c.charge_amt) AS TotalSales
FROM dbo.charge AS c
GROUP BY c.member_no
```

- Out of Order Covering Index on Charge Table
  - Index Exists which is narrower than base table
  - Used instead of table – to cover the query
- Worktable still created to store intermediate aggregated results – OUT OF ORDER (HASH)
- Data Returned OUT OF ORDER – unless ORDER BY added
- Additional ORDER BY causes another step for SORT – sorting can be expensive!

Not as Bad

- COVERING Index Scan 1,600,000 narrower rows
- Hash Aggregate yields 9,114 rows out of order
- Sort only has to sort 9,114 rows instead of 1,600,000 rows
- Return Data



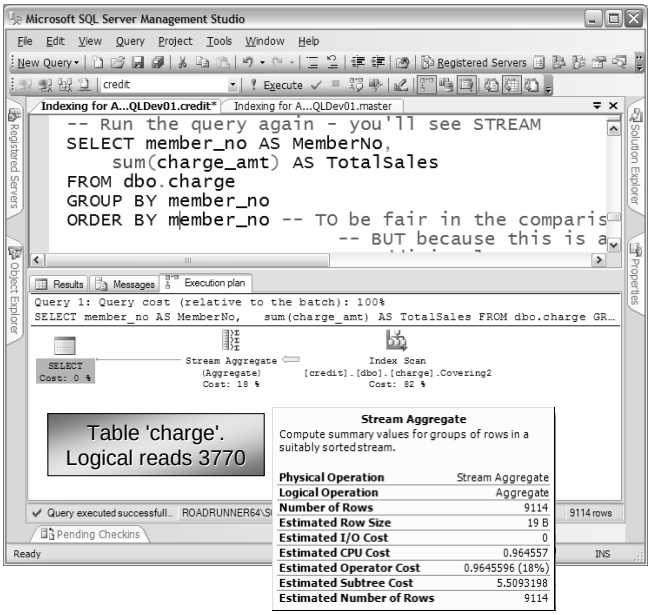
Aggregate Query
Index scan + stream aggregate

```
SELECT c.member_no AS MemberNo,
       sum(c.charge_amt) AS TotalSales
FROM dbo.charge AS c
GROUP BY c.member_no
```

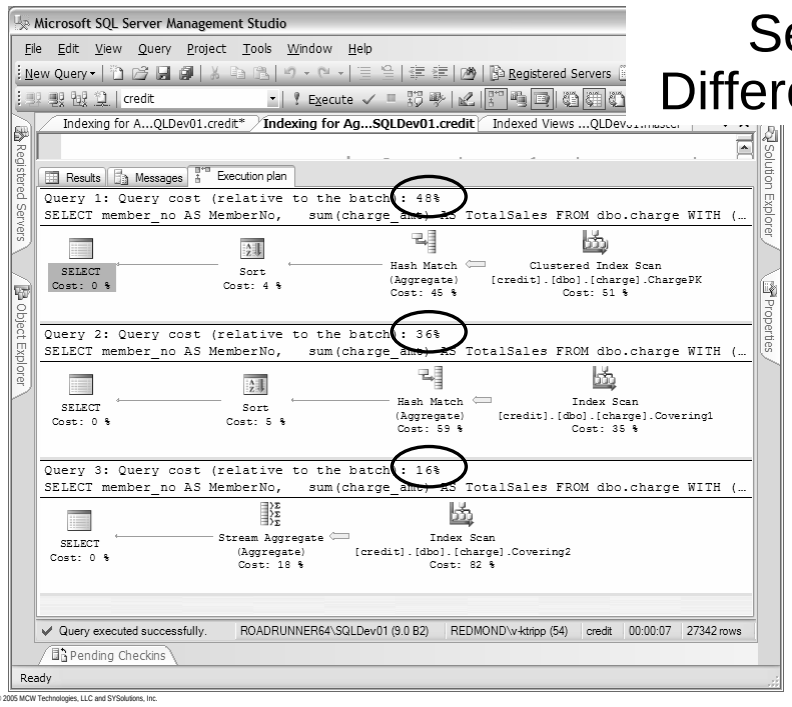
- Covering Index on Charge Table (GROUP BY first)
  - Index Exists which is narrower than base table
  - Used instead of table – to cover the query
  - Covers the GROUP BY so data is grouped
- Less work to aggregate results IN ORDER
- Data Returned IN ORDER – unless ORDER BY or other joins added
- Adding an ORDER BY identical to the GROUP BY does NOT cause any additional step for sorting!

Much Better!

- COVERING Index Scan 1,600,000 narrower rows
- Stream Aggregate also yields 9,114 rows IN ORDER NO SORT REQUIRED
- Return Data



See the Difference?

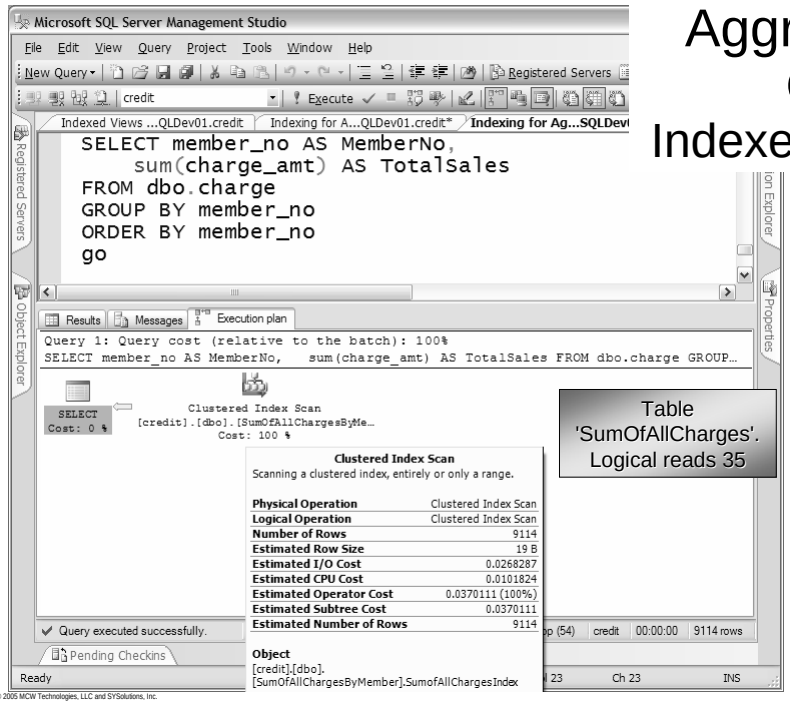


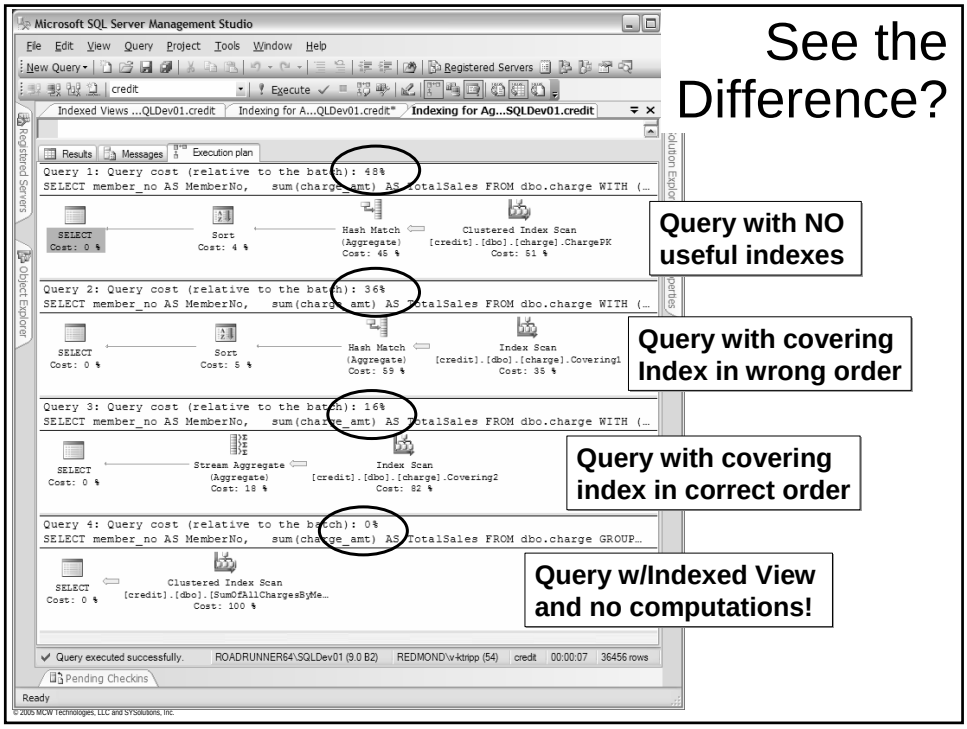
Concerns

- More temp tables
- More contention in tempdb
- Larger tempdb required
- Performance Varies on each execution
- Aggregate needs to be computed

Is there a better way?
Indexed Views!

Aggregate Query Indexed view





Indexed View for Aggregates
Key points

- TempDB access not necessary
- NO worktables are necessary
- Aggregated set should be small but not too small as to create a hot ROW spot of activity (which can create excessive blocking)
  - GROUP BY member_no – probably OK
  - GROUP BY state – too few rows in aggregate
  - GROUP BY country – *avoid* like the plague!
- Performance of data modification statements should be tested

INCLUDE Non-key Columns Compared to Indexed Views

- Indexed Views allow aggregates – adding interesting columns in the leaf level of an index offers “creative covering”
- With new INCLUDE clause, leaf level of index can include non-key columns
- Index key [has been since 7.0] limited to 900 bytes/16 columns – this is to keep tree structure and non-leaf levels optimal/small
- Allows more covering indexes
- Indexed View v. Include – depends on what needs to be in the leaf level

Finding the Right Balance

Index strategies: Summary

- Determine Primary Usage of Table – OLTP vs. OLAP vs. Combo? This determines Clustered Index
- Create Constraints – Primary Key and Alternate/Candidate Keys
- Manually Add Indexes to Foreign Key Constraints
- Capture a Workload(s) and Run through Index Tuning Wizard
- Add additional indexes to help improve SARGs, Joins, Aggregations

Are you done? NO!

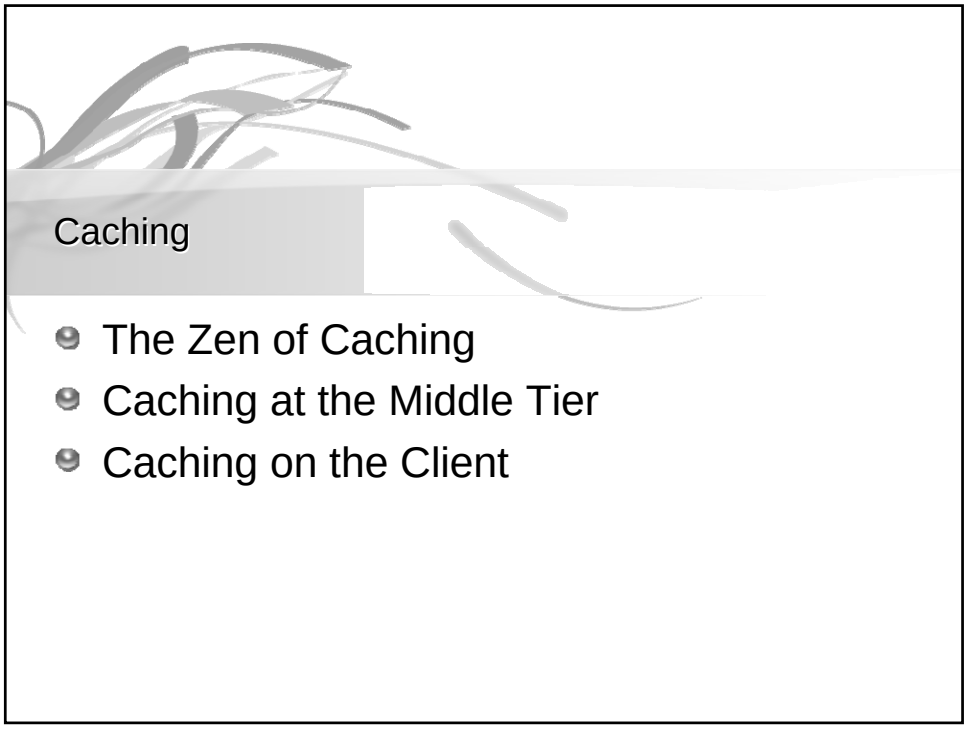
Keeping Performance Optimal

- Write effective queries to help optimizer use indexes
- Make sure statistics are accurate
  - Keep auto update statistics ON
  - Keep auto create statistics ON
- Make sure indexes don't become overly fragmented
  - Rebuild Table/Index Structures
  - Defrag Table/Index Structures

Indexing Webcasts

See Kimberly's blog for PDC Resources & Links

- MSDN Webcast: Indexing for Performance - Finding the Right Balance (SQL Server 2000)
  http://msevents.microsoft.com/CUI/EventDetail.aspx?EventID=1032254503&Culture=en-US
- MSDN Webcast: Indexing for Performance - Index Maintenance Best Practices (SQL Server 2000)
  http://msevents.microsoft.com/CUI/EventDetail.aspx?EventID=1032256511&Culture=en-US
- TechNet It's Sh0wtime Webcast: Index Creation Best Practices with SQL Server 2005
  http://www.microsoft.com/uk/technet/itsshowtime/sessionh.aspx?videoid=29
- TechNet It's Sh0wtime Webcast: Index Defragmentation Best Practices with SQL Server 2005
  http://www.microsoft.com/uk/technet/itsshowtime/sessionh.aspx?videoid=30
- MSDN Webcast Series: Effectively Designing a Scalable and Reliable Database



Caching Zen

- Moving information to a location where it can be accessed very quickly
  - From main memory to high-speed memory on the processor
  - From the hard drive to main memory
  - From main memory on one machine to another machine
- Caching is framework for managing state
  - State being data + status

Caching Zen: Why?

- Performance
  - Store the data as close as possible to consumer
- Scalability
  - Common data is often processed the same way for multiple users
    - Process it once and store the results
- Availability
  - Provide access to data even when the primary data store or service is unavailable

Caching Zen: When?

- On-demand: cache the data after it has been requested and formatted
- A priori: preload the cache with both raw and formatted data
- Combine the two

Caching Zen: Where?

- Generally as close to the data consumer as possible—wherever accessing the cache can help with
  - Performance
  - Scalability
  - Availability
- Actual cache storage can be
  - Memory resident—often on servers
  - Disk resident—often on client

Caching Zen: Issues

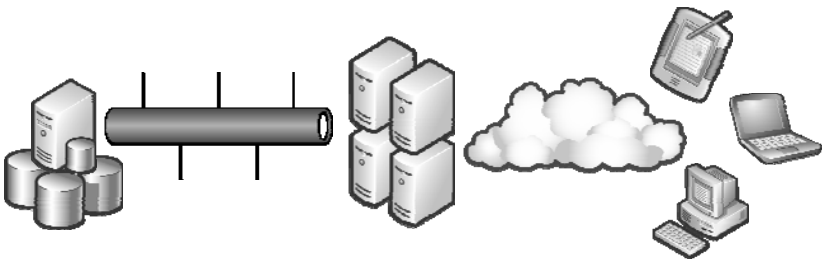
- Staleness
- Coherency
- Security

Staleness

- Data that is cached becomes obsolete
- Staleness is the difference between the master data's current value and the current version that is cached
- Staleness is relative to how often the master copy of the data changes and how tolerant an application is to stale data

Coherency

- Large systems often have multiple caches
  - Often seen with web server farms
- Caches need to kept in sync otherwise clients can get inconsistent results



Security

- Data stored in cache needs to protected
- Application logic can enforce access rules to in-memory cache on servers
- Client-side cache might require both in-memory and disk level protection
  - Spyware
  - Unsafe networks
  - Users

Staleness Solution

- Cache expiration settings based upon the application's tolerance to stale data
- Can be time driven where cache is reloaded based upon an expiration setting
- Can be data driven where cache is updated when master data is updated

Coherency Solution

- Depending upon the number of distinct cache stores, updates can occur
  - At the same time
    - Works when cache is small and speed of update is quick
  - At defined intervals
    - Works when cache is large and cache is load-balanced

Security Solution

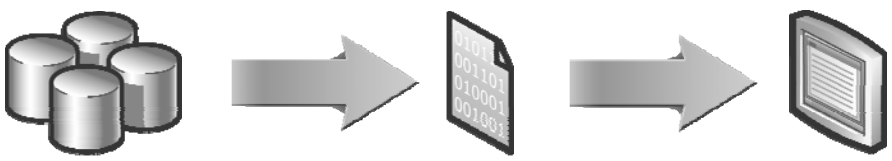
- Store cache data encrypted
- Ensure communications between cache and data source are secure

Cache Sync

- New feature of SQL Server 2005
- Cache source creates data based subscription that server will monitor
- When data changes, server notifies subscriber of change and reason
  - Uses new Service Broker infrastructure
- Exposed in SQLClient 2.0 and ASP.NET 2.0
  - High-level and low-level implementations available

State Transformation Pipeline

- State can be presented in many formats
  - Raw data object to business object to presentation object
- Caching possible at any and all points in the pipeline



Caching at the Middle Tier

- Prefer existing host processes and services
  - COM+ components and object pooling
  - IIS with ASP.NET
- Use custom services as needed
  - Windows Services with custom endpoint and cache engines

Caching at the Client Tier

- Local services
  - SQL Server Express
  - Custom Windows Services
- In-memory
  - System.Web.Caching
  - DataSets
  - Custom objects

Tuning, Monitoring, and Diagnosing System Problems

- SQL Server Profiler
- System Monitor
- Database Tuning Advisor

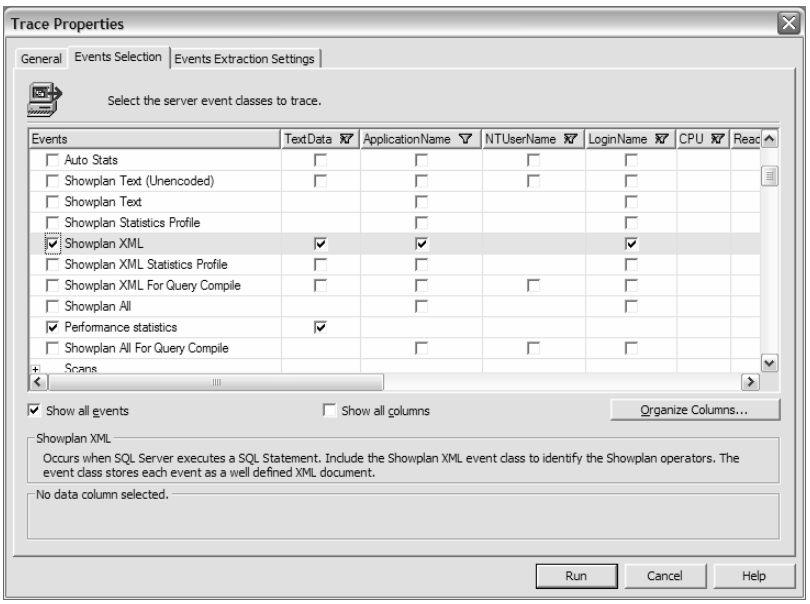
SQL Server Profiler

- Can analyze the SQL Server Database Engine and Analysis Services
- Significantly easier to setup (Events, Data Columns and Filter dialogs combined)
  - Special Events: Service Broker, Notification Services, etc…
  - Special Event types: Showplan XML and Deadlock Graph – can be saved to one (each type) or individual files for later review and interaction!
- Events that do/do not produce values for given data columns are obvious at selection (check boxes) and viewing (yellowed)
- Can Profile SQL Server 2000 and 2005
- Permissions to profile are grantable

Profiler Enhancements

- SQL Server 2000 profiler scripts run with out change on 2005
- Uses XML based definitions
- Save showplan in XML format
- Save results in XML format
- Can replay one or more rollover files
- Allows profiling of
  - Analysis Services
  - Data Transformation Services
- Correlation of Events to Performance Monitor data

Profiler UI Condensed



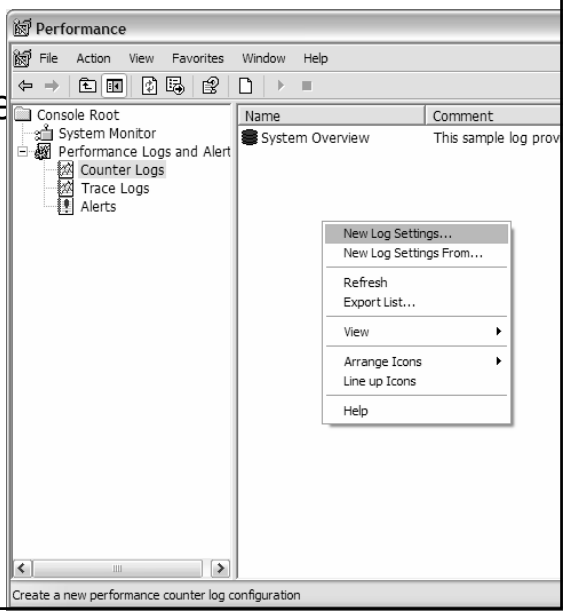
System Monitor (PerfMon) Integration
Performance Counter Logs

- Create a Profiler Trace
- Create a Performance Monitor Log
- Works solely based on time – make sure the two clients (if different) are time correlated
- Open Trace (complete load), Use File – Import Performance Data, Select Objects/Counters…
- Works with SQL Server 2000 and SQL Server 2005 as it's client side!

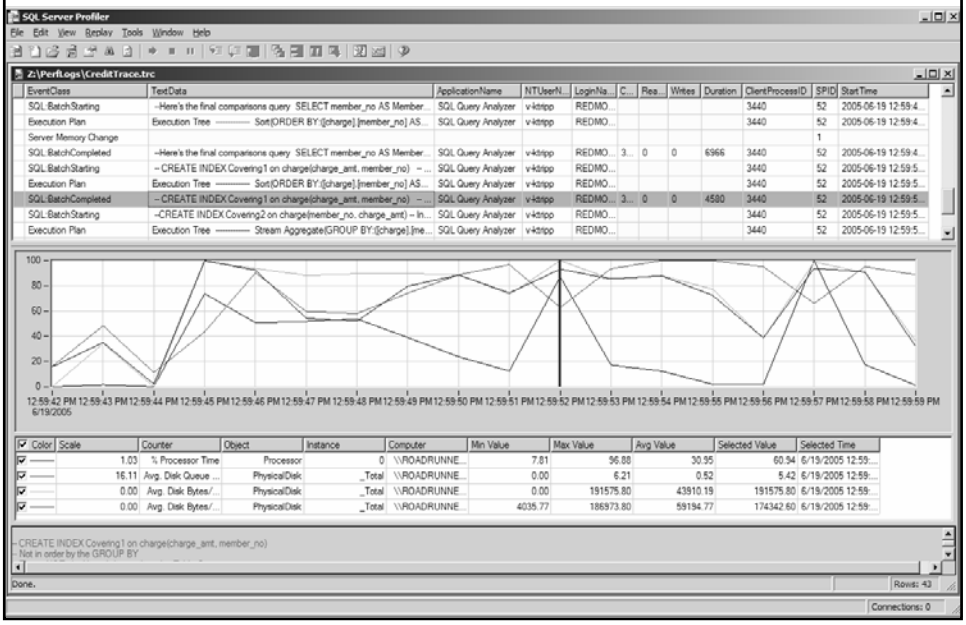
Performance Monitor Log

Define

- Objects (complete groups)
- Counters
- Sample Interval
- File Location
- Start log (time)
- Stop log (time)



Profiler with PerfMon Integration



Database Tuning Advisor

- Index Tuning Wizard replaced by DTA
- Adds:
  - Partitioning recommendations
  - Time-bound tuning
  - Indexes with Included columns -> more efficient covering
  - XML Input/Output
  - Drop ONLY mode
  - Parameterized command line execution

Database Tuning Advisor

- Can be launched from-
  - SQL Server management Studio
  - Profiler
- Import previously saved Session Definition
  - Stored in XML format
- Workload options
  - Can be a *.trc, *.sql or *.xml format
  - Can be a SQL Server Table
    - For table saved in 2000 need to launch DTA while connected to 2000 server

Database Tuning Advisor

- Workload options
  - Clone Session
    - Actions ⇨ clone session
    - Launches duplicate DTA window while maintaining chosen settings
- Tuning options
  - Physical Design Structures (PDS)
    - To use in database
    - To keep in database
  - Partitioning Strategy
  - Online index recommendations
    - Default is OFF
    - Change in Advanced Tuning Options

Database Tuning Advisor

- Optimization Process
  - Configuration info
  - Reading workload
  - Performing analysis
  - Reports
    - Tuning summary report
    - Event frequency
    - Query detail
    - Query-index relations
    - Query cost range
    - Index usage
  - Recommendations
    - Index, partition schemes

Minimizing the Impact of Profiling
Server-side Trace Queues

- INF: How to Create a SQL Server 2000 Trace (283790)
- HOW TO: Programmatically Load Trace Files into Tables (270599)
- How To: Stop a Server-Side Trace in SQL Server 2000 (822853)
- INF: How to Monitor SQL Server 2000 Traces (283786)
- INF: Stored Procedure to Create a SQL Server 2000 Blackbox Trace (281671)
- BUG: BOL Incorrectly States That Users Do Not Need to Be Sysadmin to Use Profiler or SQL Profiler SPs (this is a SQL Server 2000 limitation – 310175)
- INF: Job to Monitor SQL Server 2000 Performance and Activity (283696)

Session Summary

- Building a scalable, reliable, and performant real-world system takes MANY steps…
- **There's no magic bullet!**
- But, it is possible…

  using these techniques!

- Review Blog Entries for demo scripts and all resource links:
  - Kimberly's Blog:
    - http://www.SQLskills.com/blogs/Kimberly
  - Brian's Blog:
    - http://www.mcwtech.com/cs/blogs/brianr/

Please fill out your evaluation.
Thank you!

PDC05 DEVELOPER POWERED

Brian A. Randell
Senior Consultant
http://www.mcwtech.com

SQLskills
Kimberly L. Tripp
President and Founder
http://www.SQLskills.com