Indexing for Performance

Finding the Right Balance

Kimberly L. Tripp
Founder, President – SYSolutions, Inc.
www.SQLskills.com

Instructor – Kimberly L. Tripp

- Independent Consultant/Trainer/Writer/Speaker
- President, ***SY*Solutions, Inc.** (www.SQLskills.com)
  - EMAIL: Kimberly@SQLskills.com
- SQL Server MVP (http://mvp.support.microsoft.com/)
- Microsoft Regional Director (http://www.microsoftregionaldirectors.com/Public/)
- Writer/Editor for TSQL Solutions/SQL Magazine (www.tsqlsolutions.com and www.sqlmag.com)
- Presenter/Technical Manager for SQL Server 2000 High Availability Overview DVD (MS# 098-96661)
- Coauthor – MSPress title *SQL Server 2000 High Availability*
- Very approachable. Ask questions!

Overview

- Table and Index Concepts – Why Index?
- Table Structures
- Index Pros and Cons
- Clustered Table – the Pros and Cons
  - Choosing the Right Clustering Key
  - Clustered Index Criteria – Physical Dependencies
- Steps to Finding the Right Balance
  - Clustering Key
  - Additional "key" indexes
  - Additional indexes to improve join performance
  - Using ITW to help…
- Keeping the performance optimal
- Manually adding additional indexes

Index Concepts

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

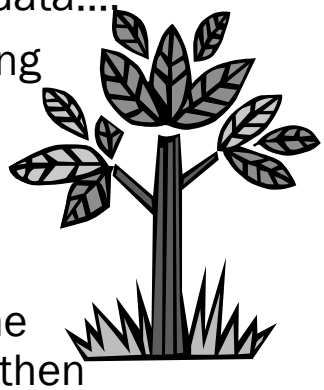
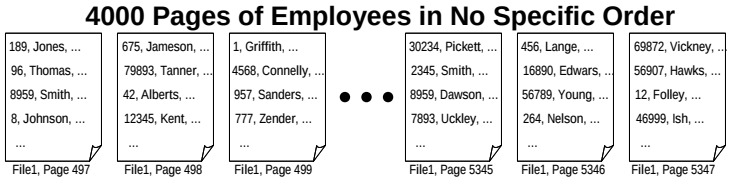


Table Concepts

- Base unit for data storage = table
- Collection of unordered pages by default = heap
- Heap access solely through exhaustive table scans…why exhaustive?
  - No indexes means no guarantee of uniqueness
  - No order to data and no uniqueness – could be more records even if you find a match on the first row of the first page (ex. where ID = 1)
- Locking cannot guarantee data modification consistency without table level locks

Table Structure – Heap

- Table without a Clustered Index
- Records are NOT ORDERED
- No Doubly-Linked List
- Access via Index Allocation Map (IAM)
  - IAMs = 8K Page (Chain) which Tracks Object Usage
    - 1 - IAM per Table/Index, per File, per 4 GB
- If NO Indexes exist – a full Table Scan required. At least 4000 I/Os on the Employee Table Heap.

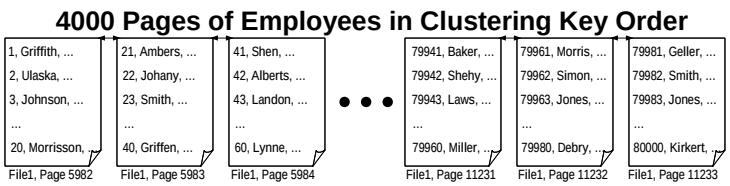


demo

Table Scan IOs on a Heap

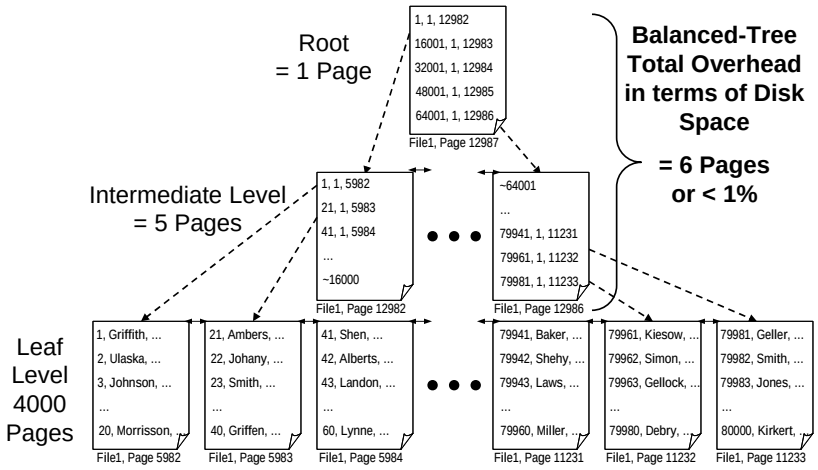
Table Structure – Clustered Table

- Clustered Index defines order – applied at CREATION
- Table is a doubly-linked list – order maintained LOGICALLY
- Expensive – in time and space – to build (1.21x table size)
- *Might* be expensive to maintain



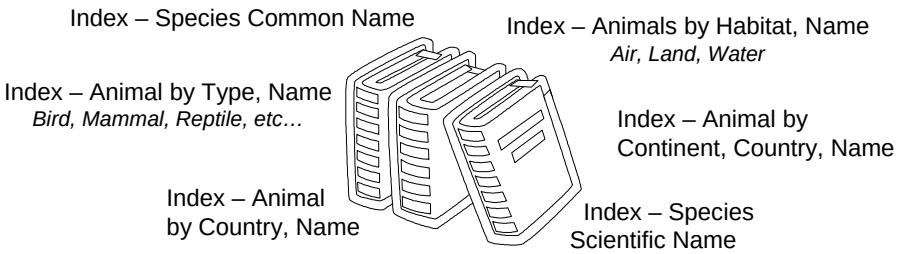
Clustered EmployeeID

Consider a clustered index on an identity column (which is the Primary Key) in an Employee Table



Non-clustered Indexes - Conceptual

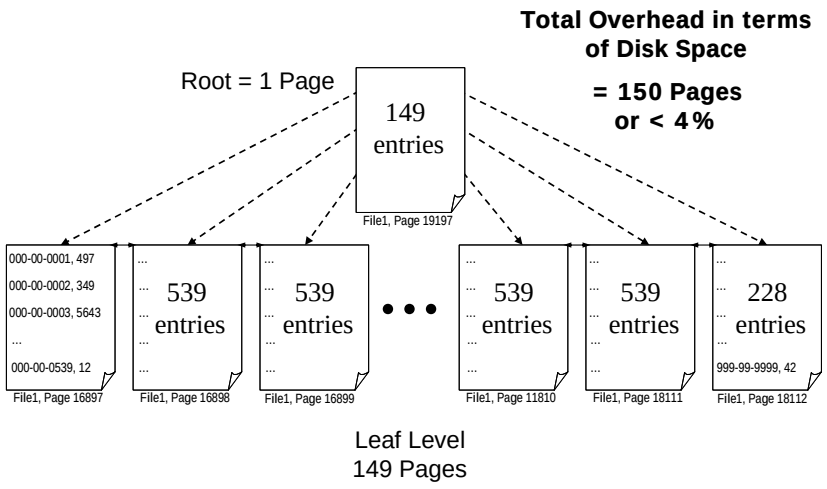
- Think of indexes in the back of a book
- To find the data in which you are interested you look up the key
- When you find the key – you must lookup the data based on its location… i.e. a "bookmark" lookup
- Always depend on the content order



Non-clustered Indexes - Physical

- Heap
  - Rows use a FIXED RID to identify the rows
  - Adds an 8 Byte Row Identifier at Insert – uses a FIXED Identifier – even if row is relocated due to Update to varchar
  - Minimizes NC Index manipulation if rows move (the NC still points to the FIXED RID)
- 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

Non-Clustered Unique Key SSN



Heap - Pros

- Effective for "staging" data
- Excellent as a partition to a view
- Indexes can be created after load
- Efficient for SCANs ONLY – when no UPDATEs (otherwise, forwarding pointers – scans become significantly less efficient)
- Space Efficient – Space from Deletes is re-used on subsequent Inserts (at the cost of performance)

Index Concepts

- PROs
  - Can speed up access to data – more options over exhaustive table scans
  - Can guarantee uniqueness of data
  - Can offer better lock granularity
  - Generally lead to better balanced performance – when indexed appropriately
- Cons
  - Add necessary overhead in INSERTs and DELETEs, add overhead in UPDATEs only when indexed column is modified
  - Add overhead in terms of Disk space
  - Add overhead in terms of Maintenance *(there are strategies for minimizing this)*

Indexing for Performance
Key Points

- Don't index columns for just for the heck of it.
  One customer said their strategy was to place a single column index on every column…just in case.
- Have a strategy!
  - Have a strategy with knowledge of data to back it
  - Look at activity and data usage in terms of volatility and query access
  - Prioritize access – in terms of user queries/type
- Minimize total indexes and find the right balance!
  - Start with only the necessary indexes and ADD from there.
  - Removing Indexes is tad harder (and more costly in terms of space and time)
  - Fewer wide indexes are usually better than lots of narrow indexes!

Finding the Right Balance
Overview

- Start with a minimal number of indexes
  - Clustered Index
  - Primary Key
  - Unique Keys
- Manually index foreign keys
  - Non-unique indexes
  - Speed up join performance
- Use ITW
- Manually index based on either:
  - Specific query tuning where ITW didn't help
  - Query frequency

Finding the Right Balance
The Clustered Index Decision

- Very Important decision
- Should be the first decision (NC based on CL)
- Must understand table structure
  - Heap v. Clustered table
  - Non-clustered indexes depend on clustered index
- Clustered Index impacts:
  - Insert performance (location and fragmentation)
  - Update performance (cl key should not be volatile)
  - Maintenance and therefore table availability

All of which impact server performance!

Clustered Index Criteria

- Unique
  - Yes – No overhead, data takes care of this criteria
  - NO – SQL Server must "uniqify" the rows on INSERT. This costs time and space. Each duplicate has a 4-byte uniqifier. When CL is rebuilt all uniqifiers are regenerated causing all NC indexes to be rebuilt as well
- 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 – Pros

- Naturally Unique
  (although not guaranteed – should combine with constraint)
- Naturally Static
  (although should be enforced through permissions and/or trigger)
- Naturally Narrow
  (only numeric values possible, only whole numbers with 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 on an Identity – Cons

- Not appropriate for extremely HIGH Insert volume (400+/sec for one table)
- Non-clustered Indexes or Indexed Views needed to handle low selectivity/range queries…
- Is that really a con? NO
  - 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. BUT must be CLUSTERED UNIQUE – if the table already has a CL Unique Key it simplifies IVs)
  - Non-clustered indexes are easier to rebuild when they become fragmented (only Shared Table Lock)
  - Non-clustered indexes are easier to keep less fragmented (more frequent rebuilds and fillfactor help – because row is smaller and more fit on a page)

Non-Clustered Indexes – Physical

- NC Indexes with a table as a Heap
  - Rows use a FIXED RID to identify the rows
  - Adds 8 Byte Row Identifier at Insert – NEVER changes even if row is relocated due to Update
  - Minimizes NC Index manipulation if rows move
- NC Indexes with a table as a 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
  - Clustering key should be static and narrow

What do we know?

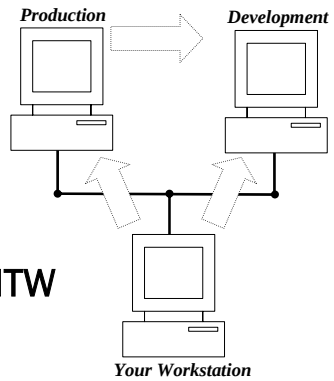
- Heaps offer excellent benefits for staging data *(for more details, see www.sqldev.net for Gert Drapers' presentation(s) on high performance data loading)*
- For OLTP/DS tables – user based modifications (not batch) – performance is generally better with a clustered index
- However, CL indexes require administrative maintenance to alleviate negatives with regard to space
- Are all Clustered Indexes going to give the same gains?
- For true performance gains you must have the RIGHT Clustered index

Finding the Right Balance
Adding additional "key" indexes

- Define the Primary Key
  - Automatically creates a unique CLUSTERED index to enforce uniqueness (can specify NONCLUSTERED)
  - Does not allow nulls
  - Should be static (relational theory rule)
  - Might be your clustered – and already created?
- Define additional alternate keys – a.k.a. Unique Keys
  - Automatically creates a unique NONCLUSTERED indexAllows NULL values but for only one "row"
- Manually index foreign keys
  - Generally improves join performance between tables
  - Not always the best for joins – but good first step
  - Non-unique indexes
  - Speed up join performance

Finding the Right Balance
Using Index Tuning Wizard

- Create a TRACE of your Production Server from your main workstation
  Realistic Sample Workload
- Backup/Restore Production Data to Development Environment
  Real Data/Real Statistics for ITW
- Point Profiler at Development Server in order to perform Index Tuning Wizard
  Does not Impact Production Server



Index Tuning Wizard
Simple Workload

- Create a workload using SQLProfilerTuning template
  - Defines the Events to monitor
  - Defines the Data Columns to monitor
- Capture a realistic workload to a FILE
- Set a max file size < 1GB and do NOT enable "rollover" as it creates a NEW file once the first file fills and could fill your harddrive
- Set filters
  - DatabaseName LIKE *YourDBName*
  - ☑ Exclude system IDs (checkbox on filters tab)

Index Tuning Wizard
Detailed Workload

- Create a workload and set your events
- You can start with SQLProfilerTuning template but make sure to add:
  - Event:StoredProcedures:SPStmtCompleted

    The ensures that even stored procedures are "stepped" into with regard to ITW. Does take longer and creates a MUCH larger workload.
- Everything else should be the same as the first run

Setting ITW Settings
Simple Route

- Launch ITW from…
  - Profiler: Tools, Index Tuning Wizard
  - SQLEM: Run a Wizard – Management, ITW
  - Query Analyzer: Query, Index Tuning Wizard
- Select your DB and Point ITW at your workload
- Keep all of the default values
  (*next until you can next no more* ☺)
- Select to tune "all tables"
- Let ITW generate recommended indexes in .sql script (so that you can review it if necessary)
- Execute the script immediately (in QA) or schedule the script to run at off hours (Job)

Setting ITW Settings
Key Options to consider changing

- Keep All Existing Indexes
  - Only turn this OFF when you're NOT going to just execute the script
  - Manually review their recommendations and check to make sure your workload is realistic
- Tuning Mode
  - Fast is good if you only want a rough and quick set of index suggestions
  - Medium looks for more options and takes a bit longer – medium is usually ideal
  - Thorough is excellent IF you're going to review the recommendations and more manually pick and choose the best indexes

Setting ITW Settings
Key Options to consider changing

- Specify Workload, Advanced Options
  - Remove the check on “Limit number of workload queries to sample” option
  - Increase the space for indexes – default is data x 3, give it more and then make sure to manually pick and choose the best indexes for YOU based on the analysis results
- Index Recommendations, Analysis button
  - Each result must be saved individually – to a text file. Favorites are:
    - Index Usage Report (Recommended v. Current)
      - Can see size and percentage used
    - Query–Index Relations Report
      - Can see which queries use which indexes

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

- Are you done?

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
- Are you done? NO!

Keeping Performance Optimal

- For the Optimizer to use indexes appropriately you must have Statistics!
  - Auto Update Statistics
  - Auto Create Statistics
- For the database to stay compact and for indexes to stay efficient you must maintain indexes:
  - Defrag indexes
    + Doesn't take the table offline
    - Creates a lot of log space
  - Rebuild indexes
    - May take the table offline (depends on index being rebuilt)
    + Completely restructures the index, all statistics, etc.

Index Fragmentation

Leaf Level

Data Modifications [can] lead to Fragmentation

- INSERT
  - Yes – Key value is not ever increasing/decreasing
  - NO – Key is ever increasing/decreasing
- UPDATE
  - Yes – Updates… to variable width columns – where the values are getting wider
  - NO – Columns are fixed width, columns have "place holder" values (i.e. DEFAULT constraints) to minimize row expansion on update OR no updates
- DELETE
  - Yes – Deletes are singleton deletes (swiss cheese problem)
  - NO – Deletes are RANGE deletes for archival purposes

How to See Fragmentation

- Poor Query Performance over time
- More disk activity
- Poor Cache Utilization
- Verify Query I/O
  - SET STATISTICS IO ON
- Verify Scan Density
  - DBCC SHOWCONTIG
    - Manually
    - Programmatically
    - Automatically
- Periodically re-verify Query I/O

DBCC SHOWCONTIG Output

- External Fragmentation
  - Scan Density
    - Uses a percentage to show extent switches
  - Logical Scan Fragmentation
    - Determines whether or not the pages owned – are contiguous (useless on a HEAP)
  - Extent Scan Fragmentation
    - Determines whether or not the extents owned have gaps (interleaved objects/indexes)
- Internal Fragmentation
  - Average Bytes Free Per Page
    - Amount of free space (if OLTP +, if OLAP -)
  - Average Page Density (fullness)
    - Shown as a percentage – how FULL are the pages

How?

- Rebuilding an Index
  - DROP and re-CREATE
  - DBCC DBREINDEX
  - CREATE with DROP_EXISTING
- Defraging an Index
  - DBCC INDEXDEFRAG
- Comparisons

Rebuilding an Index

- Generally better results w/rebuild v. defrag
- Completely removes all levels of fragmentation – from both the leaf level as well as the b-tree (i.e. completely rebalances the tree)
- Completely updates statistics – with the equivalent of a "full scan" (i.e. accurate statistics)
- Requires Locks ⇨ downtime
  - Rebuild CL requires Exclusive Table Lock
  - Rebuild NC requires Shared Table Lock
- How?
  - DBCC DBREINDEX (easier to automate)
  - CREATE w/DROP_EXISTING
    (can change the CL index definition)

Defraging an Index

- Does not completely rebuild ALL levels of index – focuses on LEAF level
- Does not update statistics
- Does not require locks for length of transaction – defrag executes as mini-trans
- May take longer if table is extremely fragmented
- May take less time if table is not fragmented
- Does not run as a single large transaction – transaction log backups can execute and the transaction log space can be freed WHILE running
- How ⇨ DBCC INDEXDEFRAG

Comparisons

	Option	If Used with Clustered Index Rebuilds NonClustered	Updates Statistics	Lock on Base Table for CL	Lock on Base Table for NC	Transaction Log Impact
Rebuilding	DROP and re-CREATE	Yes, twice	Yes	Exclusive	Shared	Longest Time, Most Activity
	DBCC DBREINDEX	Pre-sp2 YES (and only once) 2000sp2 NO - unless the CL Key is NON-unique. When the CL is not unique, the NC indexes are rebuilt EACH time the CL is rebuilt (the uniqifiers are regenerated).	Yes	Exclusive	Shared	Cannot Clear Until Complete
	CREATE with DROP_EXISTING	Same as above EXCEPT when the Clustering Key CHANGES - then all NC indexes are rebuilt only once. Otherwise, NO*	Yes	Exclusive	Shared	Cannot Clear Until Complete
	DBCC INDEXDEFRAG	No	No	N/A	N/A	Minimal Impact in sp1+

** In SQL Server 2000 sp2 DBCC DBREINDEX and CREATE with DROP_EXISTING are almost identical.The only exception is that CREATE with DROP_EXISTING can be used to CHANGE the definition of the clustered index.*

Manually Indexing for Performance

- Selectivity
- How to Improve Queries with Varying Search Arguments (SARGs)
  - Indexing for AND (any selective criteria?)
  - Indexing for OR (can you re-write as UNION?)
- How to Improve Joins
  - Are Join Conditions and SARGs individually indexed?
  - Are the combinations indexed?
  - Consider covering?
- How to Improve Aggregations
  - Consider Covering the group by/aggregated col!

Key Points

- Write Better Queries
- Limit the Search
- Limit Columns Requested
- Prioritize OLTP/OLAP
- Add Indexes for all Keys (PK/UK/FK)
- Add Nonclustered for SARGs
- Consider Covering for Low Selectivity
- Use the Index Tuning Wizard
- Test, Test, Test!

Review

- Table and Index Concepts – Why Index?
- Table Structures
- Index Pros and Cons
- Clustered Table – the Pros and Cons
  - Choosing the Right Clustering Key
  - Clustered Index Criteria – Physical Dependencies
- Steps to Finding the Right Balance
  - Clustering Key
  - Additional “key” indexes
  - Additional indexes to improve join performance
  - Using ITW to help…
- Keeping the performance optimal
- Manually adding additional indexes

Best Practices Analyzer

- New Tool which as RELEASED v1
- Checks for more than 70 best practice violations
- Helps you get ready for SQL Server 2005
  Microsoft SQL Server Best Practices Analyzer is a database management tool that lets you verify the implementation of common Best Practices. These Best Practices typically relate to the usage and administration aspects of SQL Server databases and ensure that your SQL Servers are managed and operated well.
- Check out microsoft.com/sql and then choose Best Practices Analyzer from the RIGHT pane titled "Top Downloads"*(or search for it)*

What does BPA check for?

- General Best Practices
- Rules in the following categories
  - Backup and Recovery
  - Configuration Options
  - Database Design
  - Database Administration
  - Deprecation
  - Full-Text
  - General Administration
  - Generic
  - T-SQL
  - SQL Server 2005 Readiness

Resources

- Check out www.SQLskills.com for information about upcoming events, useful downloads and excellent scripts! There are quite a few resources and/or links to use.
- MSPress title: *SQL Server 2000 High Availability A*uthors: Allan Hirt with Cathan Cook, Kimberly L. Tripp, Frank McBath ISBN: 0-7356-1920-4
- Check www.SQLskills.com to download for a sample chapter!

Resources

- Credit Database and Demo Scripts are on www.SQLskills.com, Events, Past Events
  - All the conference scripts are listed as zips in descending order based on conference dates
  - Find useful demo scripts in many areas!
- Articles on programming best practices, performance, etc. check out www.SQLskills.com, Articles – for a well ordered list of SQL Server Magazine Links
  - Many articles are free after registering on SQLMag
  - Check out the 5-part series on Joins including "n-Table Joins" where I discuss adding redundant keys. Instant Doc ID#23733

KB Articles Worth Reading!

- Q224587: *INF: Troubleshooting Application Performance with SQL Server*
- Q224453: *INF: Understanding and Resolving SQL Server 7.0 or 2000 Blocking Problems*
- Q271509: *INF: How to Monitor SQL Server 2000 Blocking*
- Q243586: *INF: Troubleshooting Stored Procedure Recompilation*
- Q263889: *INF: SQL Blocking Due to [COMPILE] Locks*

KB Articles Worth Reading!

- Q243588: HOW TO: Troubleshoot Ad-Hoc Queries
- 243589: HOW TO: Troubleshoot Slow-Running Queries
- 319942: HOW TO: Proper Configuration Settings
- 319892: INF: New Concurrency and Scheduling Diagnostics Added to SQL Server
- 305977: INF: Frequently Asked Questions - SQL Server 2000 - Table Variables
- 323630: INF: Resolving Blocking Problems That Are Caused by Lock Escalation in SQL Server

Resources

- Whitepaper: *Statistics Used by the Query Optimizer in Microsoft SQL Server 2000*, http://msdn.microsoft.com/library/techart/statquery.htm
- Whitepaper: *Query Recompilation in SQL Server 2000*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/sql_queryrecompilation.asp
- Whitepaper: Index Tuning Wizard for Microsoft SQL Server 2000, http://msdn.microsoft.com/library/en-us/dnsql2k/html/itwforsql.asp?frame=true
- Whitepaper: *Improving Performance with Microsoft SQL Server 2000 Indexed Views*, http://msdn.microsoft.com/library/en-us/dnsql2k/html/indexedviews1.asp?frame=true

Resources

- Whitepaper: Microsoft SQL Server 2000 Index Defragmentation Best Practices http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/sql/maintain/Optimize/SS2KIDBP.asp
- Whitepaper: *Using Partitions in a Microsoft SQL Server 2000 Data Warehouse,* http://msdn.microsoft.com/library/techart/partitionsindw.htm
- *Support WebCast: SQL Server 2000 Profiler: What's New and How to Effectively Use It* http://support.microsoft.com/default.aspx?scid=%2Fservicedesks%2Fwebcasts%2Fwc111400%2Fwcblurb111400%2Easp

Support Resources

- Lots of Great links www.microsoft.com/sql
- Support Overview
  http://www.microsoft.com/sql/support/default.asp
- Newgroups – Listings, info on how to use, etc…
  http://www.microsoft.com/sql/support/newsgroups/default.asp
- Technical Resources
  http://www.microsoft.com/sql/techinfo/default.asp
- SQL Server Whitepapers (some already listed)
  http://www.microsoft.com/isapi/redir.dll?Prd=msdn&Ar=sqlserver

Resources

- See www.sqlmag.com & www.tsqlsolutions.com for articles on Backup/Restore
- From Books Online “Home Page” select White Papers to get to msdn
- For Tech Net articles use:
  http://www.microsoft.com/technet/prodtechnol/sql/default.asp?frame=true
- See www.microsoft.com/sql for all sorts of stuff!
- Support Resources listed:
  http://www.microsoft.com/sql/support/default.asp
- Webcasts:
  http://support.microsoft.com/default.aspx?PR=pwebcst&FR=0&SD=MSDN&LN=EN-US&CT=SD&SE=NONA

Community Resources

- Community Resources
  http://www.microsoft.com/communities/default.mspx
- Microsoft Regional Directors (MS RD)
  http://www.microsoftregionaldirectors.com/Public/
- Most Valuable Professional (MVP)
  http://www.mvp.support.microsoft.com/
- Newsgroups
  Converse online with Microsoft Newsgroups – worldwide!
  http://www.microsoft.com/communities/newsgroups/default.mspx
- User Groups
  http://www.microsoft.com/communities/usergroups/default.mspx

Questions and Answers

- Submit questions using the text box

Thank you!

Kimberly L. Tripp
Consultant . Trainer . Writer . Speaker

email: **Kimberly@SQLskills.com**
Make sure to register for special offers and other helpful information and resources!
www.SQLskills.com