IDS SQL Performance Tuning

Lester Knutsen

Advanced DataTools

Topics

- Indexes
- Update Statistics
- Query Plan – set sqexplain on
- Optimizing Sorts
- Using PDQ

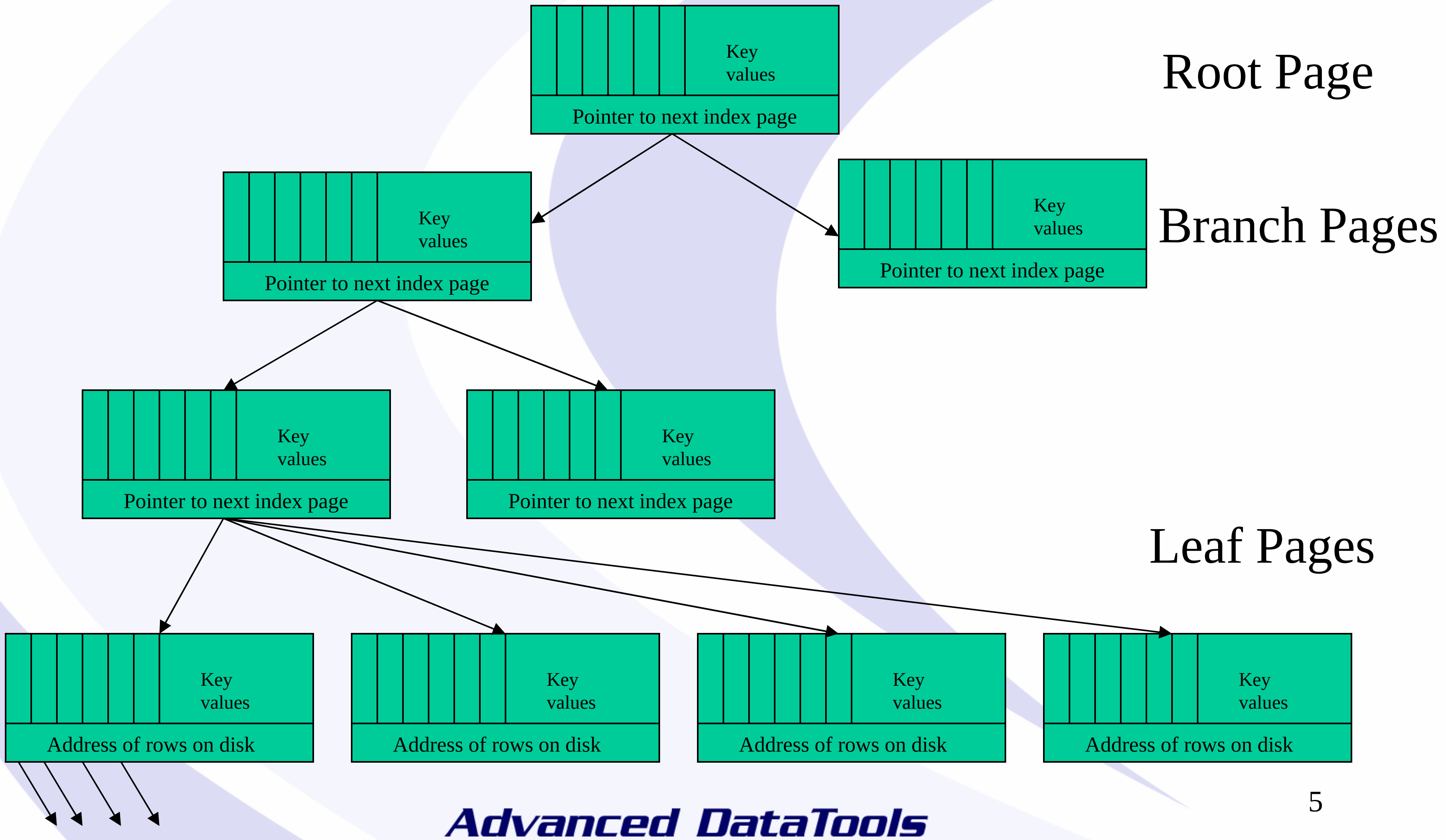
Key Performance Considerations

- How many pages are read?
- How many CPU cycles are used?
  - “if” test
  - Conversions
- How much memory/sort space is required?

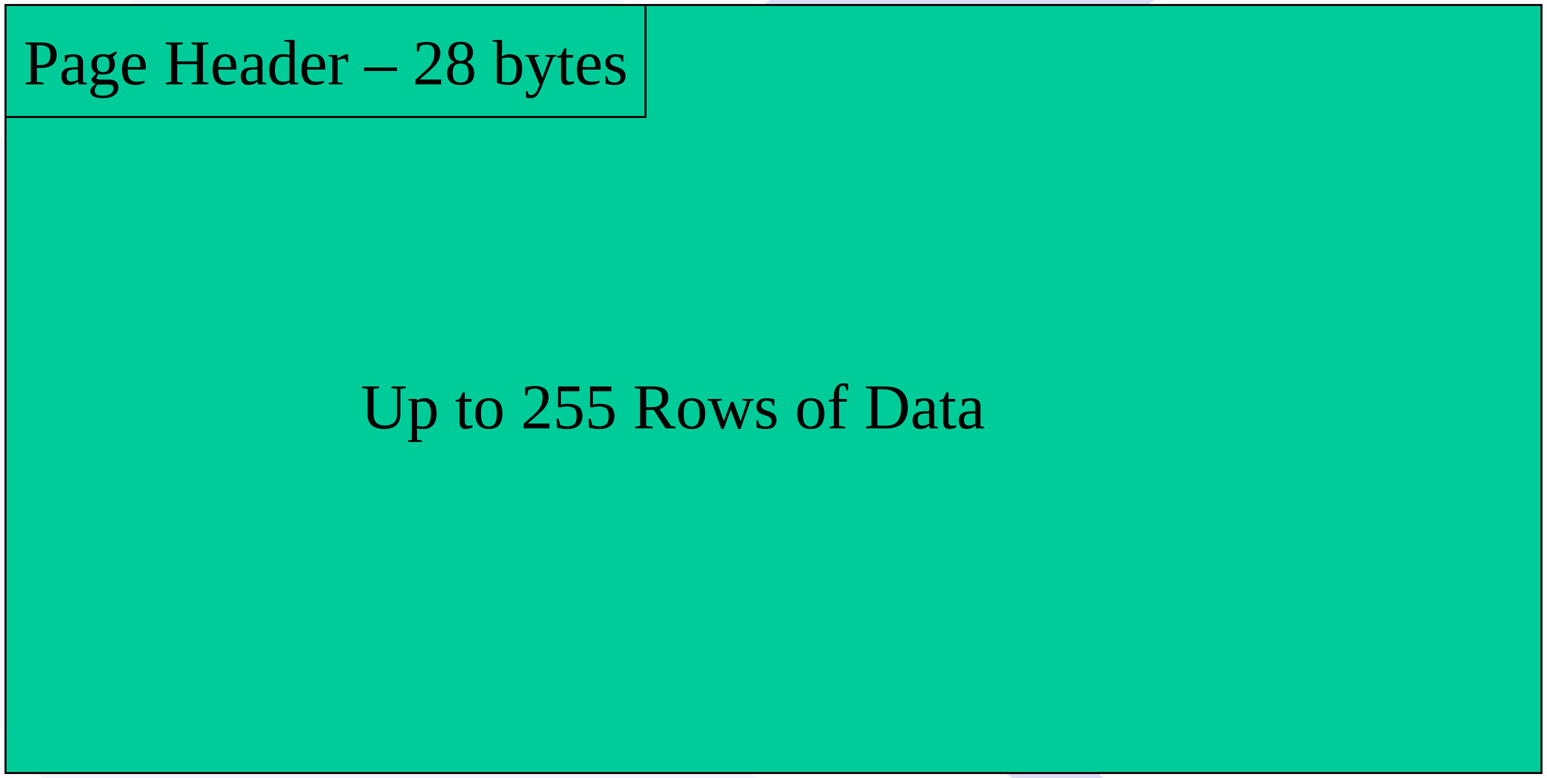
How Does the Database Get Data?

- Sequence Scan
  - Loop until end of file
    - Read row
    - Test row
    - Continue – get next row
- Index Read
  - Loop through index pages until find row
  - Read row

Structure of an Index



Structure of a Data Page



A 2048 byte page can hold 2020 bytes of data

What is an Index?

- Root page – points to branch pages
- Levels of branch pages – point to other branch pages until the bottom level is reached, this is a leaf page.
- Leaf pages point to where the row is on disk (row id or partition and row id)
- The levels and size of an index depend on:
  - the number of unique keys in an index
  - the number of index entries that each page can hold
  - the size of the columns being indexed
- Index leaf pages contain:
  - Sorted column values – the key
  - Pointer to the data row – the address of the row
- A unique index contains one index entry for every row in the table.

Performance Costs of an Index

- Read a row using an index - may require reading several index levels to get to the data row page.
- Delete a row and all index entries that point to the row must be deleted. This may require re-writing several pages.
- Insert a row and new index entries must be added. If space does not exist, new index pages and levels may need to be created.
- Update a row and any index entries that point to the row via columns that are updated will need to be updated.
- When a row is inserted or deleted at random, allow three to four added page I/O operations per index.
- When a row is updated, allow six to eight page I/O operations for each index that applies to an altered column. If a transaction is rolled back, all this work must be undone. For this reason, rolling back a transaction can take a long time.

When is an Index Required?

- Enforce unique columns in a table
- Primary keys
- Foreign keys

When are Indexes Helpful?

- Columns used in joins
- Columns used in filter expressions (SQL where clause)
- Columns used for ordering or grouping
- Columns that do not contain duplicate keys
- Let’s talk about clustered indexing

Clustered Indexes

- Perform one-time sorts that reorders physical data rows to match index order
- Requires enough free space to build a copy of the table during the process
- Locks table during rebuild process
- Syntax:
  - alter index index_name to cluster
- Maintenance operation

Composite Indexes

- Order of columns is important
  - Example: Which indexes are better?:
    - (first_name, last_name, company)
    - (company, first_name, last_name)
    - (company, last_name, first_name)
    - (company)
    - (first_name, last_name)
- Avoid unneeded redundant indexes

Index Fill Factor

- What percent of an index page is filled when the index is first created?
- Use 100 for tables that never grow
- Use 90 - default
- Use 50 for high-growth tables
  - Create index index_name on table X (col1) fillfactor 50;

Index Fragmentation

- Locate indexes in separate dbspace from data
- Fragment indexes like the data by expression
- Fragment indexes using a different expression than the data

Indexes – Attached vs Detached

- Default in version 7.X – Attached Indexes
  - Index pages are stored within the same tablespace as data pages
- Default in version 9.X and 10.X– Detached Indexes
  - Index pages are stored in separate tablespace from the data pages

How to Monitor Indexes?

- Compare reads and writes on an index - less reads indicates the index may not be needed unless it is a constraint.

```
Select idxname,
      isreads,
      bufreads,
      pagreads,
      iswrites,
      bufwites,
      pagwrites,
      lockreqs,
      lockwts,
      deadlks
from sysmaster:sysptprof a, sysindexes b
where a.tabname = b.idxname
```

Update Statistics Tasks

- Updates the system catalogs
- Collects data distribution
- Collects information for the Query Optimizer
- Re-optimize (compiles) stored procedures

Update Statistics and Data Distribution

Update statistics [LOW|MEDIUM|HIGH] for table tablename (columns);

LOW-	least amount of data gathered; updates systables, syscolumns, sysindexes; does NOT update sysdistrib
HIGH -	most amount of data gathered; requires full table scan for each column updates sysdistrib; default resolution of 0.5 % - (200 bins)

MEDIUM - data obtained by sampling; not full table scans, faster than HIGH; default resolution is 2.5% - (50 bins)

Drop Distributions:

Update statistics LOW for table tablename (columns) drop distributions;

Stored Procedures:

Update statistics for stored_procedure_name;

Statistics Collected During Update Statistics Low

- Systables
  - Number of rows
  - Number of pages to store the data
- Syscolumns
  - Second largest value for a column
  - Second smallest value for a column
- Sysindexes
  - Number of unique values for the lead key
- Sysindexes
  - How highly clustered the values are for the lead key

Data Distributions

- Created with update statistics MEDIUM or HIGH
- Distributions are a mapping of the data in the column into a carefully chosen set of column values.
- The data in the column is examined and divided into bins, which represent a percentage of data. For example, if a bin might hold 2 percent of the data; 50 bins would hold all the data.
- RESOLUTION is the percent of data that is held in each bin.
  - MEDIUM = 2.5%   (50 bins)
  - HIGH =   0.5%   (200 bins)
- The Optimizer uses distributions for columns referenced in a WHERE clause to estimate the effect of a WHERE clause on the data.
- You must have DBA privileges, or be the owner of the table, in order to create HIGH or MEDIUM distributions.

Use Dbschema to View Distributions

- dbschema –d database –hd table
  - dbschema –d stores_demo –hd orders
- Displays histograms of the distribution for columns of a specified table, or "all" for all tables.

Distributions Output

```
Distribution for lester.orders.paid_date

--- DISTRIBUTION ---
1: (  1,   1,  06/03/1998)
2: (  1,   1,  06/14/1998)
3: (  1,   1,  06/21/1998)
4: (  1,   1,  07/10/1998)
5: (  1,   1,  07/21/1998)
6: (  1,   1,  07/22/1998)
7: (  1,   1,  07/31/1998)
8: (  1,   1,  08/21/1998)
9: (  1,   1,  08/29/1998)
10: (  1,   1,  08/31/1998)
11: (  1,   1,  09/02/1998)
12: (  1,   1,  09/20/1998)

--- OVERFLOW ---
1: (  6,      NULL)
2: (  2,      08/06/1998)
3: (  3,      08/22/1998)
```

Bin (# of rows, # unique, highest value)

When one value exceeds 25% of a Bin it will get placed in the Overflow Bin (# values, value)

Process for Collecting Distributions

- Develop scan plan based on available resources
- Scan table – Isolation Dirty Read
  - High = All rows
  - Medium = Sample of rows
- Sort each column
- Collect distributions into bins
- Begin transaction
  - Delete old columns distributions
  - Insert new columns distributions
- Commit transaction

Update Statistics Medium Resolution Clause

- Use the Resolution clause to adjust the size of the distribution bin, designate whether or not to avoid calculating data on indexes, and with the MEDIUM mode, to adjust the confidence level.
  - confidence_level – A factor of the number of samples (estimated fraction of the time that sampling in MEDIUM mode should produce the same results as the exact HIGH mode). Default level is 0.95. Must be within the range from 0.80 (minimum) to 0.99 (maximum).
  - percent - Percentage of sample in each bin of distribution. Default is 2.5 for MEDIUM and 0.5 for HIGH.
    - Example
      - 100,000 rows in the table
      - Resolution of 2%
      - Each bin will represent 2,000 rows
  - Default Medium with confidence_level = .95 and percent = 2.5 will sample 2,963 records.

Recommendations for Update Statistics

- Execute UPDATE STATISTICS (LOW) for the database on a regular basis.
- Execute UPDATE STATISTICS MEDIUM DISTRIBUTIONS ONLY for selected tables, or for the entire database if time is available.
  - The DISTRIBUTIONS ONLY keyword prevents re-updating the index data and speeds up the process.
- Execute UPDATE STATISTICS HIGH DISTRIBUTIONS ONLY for:
  - all columns that are listed first in an index
  - key columns used in joins
  - key columns used in SQL where clauses as filters

- **The goal is to balance the time required to execute update statistics vs improving query performance.**

Improving Performance for Update Statistics

- Turn on PDQ when running update statistics, but only for tables
- DO NOT use PDQ when updating statistics for Stored Procedures
- When running High or Medium, increase the memory update statistics has to work with
- Enable parallel sorting (i.e. PDQPRIORITY, PSORT_NPROCS)
- Enable parallel temp dbspace (PSORT_DBTEMP, DBSPACETEMP)

Improving Sorts

- Avoiding Sorts
  - Improve indexes on tables
  - However, sorts consume less resources than reading the index when the number of index pages is large
- Narrow columns require less work to sort
- Reduce the data to sort, only select the rows you need
  - SELECT FIRST N
  - SELECT FIRST N SKIP M

Memory Required by Sorts

- The default memory used for a sort is 128KB
- If the sort takes more than 128KB of memory, the sort will go to disk
- New parameter in IDS 10:
  - DS_NONPDQ_QUERY_MEM
    - The amount of memory given to sorts who have a PDQ of 0
    - Minimum Value 128KB
    - Maximum Value 25% of DS_TOTAL MEMORY
    - onmode -wf DS_NONPDQ_QUERY_MEM=500

Disk Used by Sorts

- DBSPACETEMP as defined in Onconfig
- Rootdbs
  - When database is logged and DBSPACETEMP is a nonlogged dbspace
  - When no DBSPACETEMP is defined
- /tmp
  - When PSORT_DBTEMP or DBSPACETEMP is not set
- PSORT_DBTEMP
  - Best performance for sorts – several file systems

Big Sorts

- Enable PDQPRIORITY to allow more threads and resources
- Ensure DS_TOTAL_MEMORY in the onconfig file is set properly
- Make sure PSORT_NPROCES is set to the number of CPU VPs
- Make sure PSORT_DBTEMP is set

Monitoring Sorts from Sysmaster

- System Level:

select * from sysprofile where name matches "*sort*“

  – totalsorts, memsorts, disksorts, maxsortspace

- User Level:

select sid, total_sorts, dsksorts, max_sortdskspace from syssesprof;

Advanced DataTools

PDQ

- What is PDQ?
- How do you configure PDQ?
- How do you manage PDQ?
- Examples and Exercise

What is PDQ?

- PDQ divides large database queries into multiple parallel tasks.
- Parallelized PDQ operations include:
- Scans
- Sorts
- Joins
- Aggregates
- Inserts
- Deletes

How Do You Enable PDQ?

- PDQ can be turned on for a particular query or set of queries using the SQL statement:
    - SET PDQPRIORITY
- PDQ may be turned on for all queries run by particular users using the environmental parameter:
    - PDQPRIORITY

How Do You Manage PDQ?

- PDQ administration involves managing resources allocated for parallel operations through OnLine Dynamic Server’s:
  - ONCONFIG file configuration parameters
  - Memory Grant Manager
  - **onmode**

PDQ Configuration Parameters

- MAX_PDQPRIORITY
  - System-wide governor of memory and CPUVP resources
- DS_MAX_QUERIES
  - Maximum number of decision support queries that may run concurrently
- DS_TOTAL_MEMORY
  - Maximum memory available for decision support queries
- DS_MAX_SCAN
  - Maximum number of concurrent scan threads executed for parallel queries

Configuration Parameters that Affect PDQ

- Two other configuration parameters which are important to PDQ operations are:
  - RA_PAGES
  - RA_THRESHOLD

  Note: RA_PAGES and RA_THRESHOLD control the number and size of light scan buffers allocated for large sequential scans:
  - Size of light scan buffers
    - RA_PAGES / (MAXAIOSIZE / PAGESIZE)
  - Number of light scan buffers
    - (RA_PAGES + RA_THRESHOLD) / (MAXAIOSIZE / PAGESIZE)

Changing PDQ Parameters When the System is Online

- Change DS_TOTAL_MEMORY:
  - **onmode -M *kbytes***
- Change DS_MAX_QUERIES:
  - **onmode -Q *max_queries***
- Change MAX_PDQPRIORITY:
  - **onmode -D *priority***
- Change DS_MAX_SCANS:
  - **onmode -S *max_number_scan_threads***

Memory Grant Manager (MGM)

- The Memory Grant Manager manages and reserves resources for Parallel Database Queries including:
  - The number of concurrent queries
  - The number of scan threads
  - The number of PDQ threads
  - The amount of memory and CPU

How Memory is Granted

- The smallest unit of memory that may be *granted* to a query is a quantum.
  - quantum = DS_TOTAL_MEMORY/DS_MAX_QUERIES
- Total memory reserved for a query when the query registers with the MGM is influenced by many factors, but generally the memory granted to a single query is:
  - DS_TOTAL_MEMORY * (PDQPRIORITY / 100) * (MAXPDQPRIORITY / 100)
  - rounded down to the nearest quantum, but not less than one quantum.

Scan Threads and Secondary Threads

- The number of scan threads allocated will be the minimum of either:
  - the number of fragments being scanned
  - ((PDQPRIORITY/100) * (MAXPDQPRIORITY/100) * DS_MAX_SCANS)
- The number of secondary threads allocated is determined by:
  - ((PDQPRIORITY/100) * (MAXPDQPRIORITY/100) * NUMCPUVPS

Factors to Consider in Tuning PDQ

- How many users will be running decision support queries at one time?
- Do some queries have higher priority than others?
- Should all DSS queries run at once, or should some wait?
- Is OLTP SQL running at the same time as DSS queries?
- How much memory is available?

The Query Plan

- Set SQEXPLAIN ON
  - – Display the query plan that the optimizer chooses, and execute the query.
- Set SQEXPLAIN ON AVOID_EXECUTE
  - – Display the query plan that the optimizer chooses, but do not execute the query (new in IDS 10).
- onmode -Y sid
  - – Display the query plan that the optimizer chooses for a sid (new in IDS 10).

EXPLAIN Output

- The SELECT statement for the query
- Estimate of the query cost (in units) the optimizer uses to compare plans. These units represent a relative time for query execution, with each unit assumed to be roughly equivalent to a typical disk access. The optimizer chose this query plan because the estimated cost for its execution was the lowest among all the evaluated plans.
- An estimate for the number of rows that the query is expected to produce
- The order to access the tables during execution
- The access plan by which the database server reads each table

Explain Output - Access Plan

- SEQUENTIAL SCAN
  - Reads rows in sequence
- INDEX PATH
  - Scans one or more indexes
- AUTOINDEX PATH
  - Creates a temporary index
- REMOTE PATH
  - Accesses another database (distributed query)
    - The table column or columns that serve as a filter, if any, and whether the filtering occurs through an index
    - The join plan for each pair of tables

Explain Output - Join Plan

- DYNAMIC HASH
  - Use a hash join on the preceding join-table pair. The output includes a list of the filters used to join the tables. If DYNAMIC HASH JOIN is followed by (Build Outer) in the output, the build phase occurs on the first table. Otherwise, the build occurs on the second table, preceding the DYNAMIC HASH JOIN.
- NESTED LOOP
  - Use a hash join on the preceding join-table pair. The output includes a list of the filters used to join the tables. The optimizer lists the outer table first for each join pair.

Thank You

Lester Knutsen

Advanced DataTools Corporation

Lester@advancedatatools.com