Preparing to Get the Best Performance Out of DB2 for i & the SQL Query Engine (SQE)

Tom McKinley, mac2@us.ibm.com
DB2 for i Center of Excellence
IBM Rochester, MN USA

i want stress-free IT.
i want control.
i want an i.
Query Optimization – On Demand

- After an OS upgrade or DB PTF apply, query plans are invalid.
- Invalid plans are discarded and the SQL request is reoptimized.
- Reoptimization results in a new plan.
- The new plan might use the same methods and strategy.
- The new plan might use different methods and strategy.
- Invalidation and reoptimization allows the use of new techniques.
- New techniques provide the potential for faster plans.
Query Optimization – On Demand Example

Before

Results
  Join
    Table 1
    Temp IX
      Table 2
      Local select

After

Results
  Join
    Hash 2
      Table 2
      Local select
    Hash 1
      Table 1
      Local select
A New SQL Query Engine

- Using IBM i Operating System V5R2 and later means using the new DB2 for i SQL Query Engine (SQE)
- Preparing for the OS upgrade will provide the best “out of the box” experience
- In other words… getting the most out of DB2 for i
Pre-V5R2 Database Architecture

The optimizer and database engine are separated at different layers of the operating system.
Post-V5R2 Database Architecture

The optimizer and database engine merged to form the SQL Query Engine and much of the work was moved to SLIC.
Post-V5R2 Database Architecture

The optimizer and database engine merged to form the SQL Query Engine and much of the work was moved to SLIC.
What’s in a Name?

- IBM DB2 for i represents the integrated relational database for the IBM i Operating System
- The Optimizer now determines what engine will handle the query request through via the Query Dispatcher
  - Classic Query Engine (CQE)
  - SQL Query Engine (SQE)
- Still only one interface into the optimizer
The Query Dispatcher

• Determines which engine will optimize and process the query request
  – Only SQL queries are considered for the SQL Query Engine
  – OPNQRYF, QUERY, Query API do not use the SQL Query Engine

• Initial step for all query optimization that occurs on the system

• The use of SQE is being phased in over several releases
The Query Dispatcher → SQE

• V5R2
  – Read only SQL (subSELECT of an INSERT supported)
  – Single Table (i.e. no joins)
  – AND, OR, IN predicates (no LIKE support)
  – Joins
  – SMP requested

• V5R3
  – All of the above plus
  – VIEWs
  – UNIONs
  – SubQueries
  – INSERT, UPDATE, DELETE
  – Star Schema Join (STAR_JOIN without QAQQINI support)
The Query Dispatcher \(\rightarrow\) SQE

- **V5R4**
  - All of the above plus…
  - LIKE predicates
  - LOB columns
  - ALWCPYDTA(*NO) & SENSITIVE Cursors

- **IBM i 6.1**
  - All of the above plus…
  - User Defined Table Functions (UDTFs)
  - Translation (UPPER, LOWER, etc.)
  - Alternate sort sequences
  - DEFAULT FOR `IGNORE_DERIVED_INDEX` QAQQINI file changes to *YES
The Query Dispatcher → SQE

• IBM i 7.1
  – All of the above plus
  – Logical File references on FROM clause
    • Latest IBM i 6.1 Database Group PTF enables SQE to process simple Logical File references on FROM clause

• Remaining SQE restrictions:
  – Non-SQL interfaces (Query/400, OPNQRYF, QQQQRY API)
What are Statistics, and why are they important?

- Table and column statistics describe the data to the DBMS
- Query optimizers use statistics to understand the data
- If the correct statistics are collected and available, the cost based optimizer will more accurately estimate the number of rows to process
- Better estimates will allow for better query optimization and the selection of the best query plan
Statistics

• All query optimizers rely upon statistics to make plan decisions
  – DB2 for System i has always relied upon indexes as its source for column statistics
  – Other databases rely upon manual stats collection for their source

• SQE offers a new, hybrid approach
  – In addition to indexes, single **column** statistics can be gathered and used
  – **Automatically** collected for cases where indexes do not already exist
  – Column statistics stored with table object
  – Column statistics do not have any maintenance overhead when row is inserted, updated or deleted
  – Automatically refreshed as they become stale
What are Indexes, and why are they important?

- Indexes logically organize and reference rows in a table.
- Indexes describe the data to the DBMS.

- If the correct indexes are available, the cost based optimizer will have more options available for implementation:
  - Based on the range and cardinality of the data.
  - Based on the selectivity of the query.

- More options should allow for better overall query performance.

- Indexes are maintained as the underlying data changes:
  - Column statistics are not maintained, only refreshed.
SQE Statistics Manager

- Controls access to all meta-data used for query optimization
  - Does not actually run or optimize a query
- Answers questions posed by the SQE Optimizer
  - Accuracy of the answers will dictate the optimizer’s ability to choose the best plan
  - Must always provide an answer to a question
- Answers are derived from different stats sources
  - Indexes – if available
  - Column stats – if available
  - Default values
Sources for Answers

- Meta-data sources
  - Permanent Indexes (Radix or Encoded Vector)
    - More accurately describes multi-key values
    - Stats available immediately as the index maintenance occurs
    - Can be used for query implementation
  - Column Statistics
    - Column Cardinality, Histograms & Frequent Values List
    - Constructed over a single column in a table
      - Stored internally as a part of the table object
    - Collected automatically by default for the system
    - Stats not immediately maintained as the table changes. Instead stats are refreshed as they become “stale” over time
    - Column stats are not mirrored or propagated to another system or LPAR
    - Cannot be used for query implementation
  - Default or Stale Sources
    - No representation of actual values in columns
Automatic Statistics Collection

• Allows Statistics Manager to self-manage the collection of stats
  – Automatic collection is performed by default
  – Statistics On-Demand!
  – Might answer the questions more accurately

• As queries are run, the Statistics Manager will request a background collection on columns where the answers are derived from default sources
  – The goal is the stat will be available for future executions of the query
Statistics Collection Interface

- System value to control automatic stats collection
  - QDBFSTCCOL Database file statistics collection
    - *ALL Allow both user & system requested stats collections
    - *NONE No stats collection is allowed
    - *USER Only user requested stats collection allowed
    - *SYSTEM Only system requested stats collection allowed

- QDBFSTCCOL – Stats collection job
  - Automatic collection runs in this background job at very low priority
  - Statistics Manager continuously analyzes entries in the Plan Cache and queues up requests for the collection job

- iSeries Navigator graphical interface to manage stats collected by the system
  - API’s also provided to manage the stats programmatically
Auto Stats Considerations

- When an index is not available to answer a stats question, the query might run significantly longer on the first execution
  - The SQE Optimizer may interpret the same set of default statistics differently than CQE, resulting in a different plan
  - Plans are re-optimized on-demand as the new stats become available
- This warm-up effect can be prevented by proactively creating indexes
  - Columns most at risk are those with no indexes or those that are not a part of the leading keys of the index
  - Local selection columns
  - Joining columns
  - Grouping columns
  - Ordering columns
Automatic Statistics Example

Q1 = ...WHERE Customer_No > 112358  GROUP BY Customer_No...

<table>
<thead>
<tr>
<th>No Index</th>
<th>No Index Stat</th>
<th>Index Stat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 optimizes with defaults, runs without index</td>
<td>Q1 optimizes with stat, runs without index</td>
<td>Q1 optimizes with stat and index, runs with index</td>
</tr>
</tbody>
</table>

Queue up stats request for Customer_No

Generate Stats for Customer_No

Create index for Customer_No

Time

Is there a difference in performance?
SQE Plan Cache

• Incorporates Self-Managing Technology
  – Cache is automatically maintained to keep most active queries available for reuse
  – Plans are optimized on-demand as new stats or indexes become available
  – Foundation for a self-learning query optimizer to interrogate the plans to make wiser costing decisions

• Caches all access plans optimized by the SQE Optimizer
  – Allows more reuse of existing plans regardless of interface for identical SQL statements
  – Works in conjunction with the System Wide Statement Cache and the SQL programs, packages and service programs

• Cache is cleared during an IPL

• New interface to observe and analyze SQE plan information
SQE Plan Cache Example

Plan Cache

Plan X

Plan Y

Plan Z

Statement 1
Statement 2

Statement 3
Statement 4

Statement 3
Statement 5

CQE

CQE
SQE Plan Cache

- Better centralized management of plan information:
  - Plans are stored in a compressed mode
  - Plans stored independent of job information for better sharing of plans
  - Access is optimized to minimize contention on plan entries across the system
  - Out of date plans are cycled from the cache as more space is needed
  - Multiple plans can be maintained for identical SQL statements (library list or environmental changes)

- Repository of information that is used to determine feedback and automatic stats generation

- Enabling auto stats collection causes the Statistics Manager to interrogate the Plan Cache looking for plans where stats would have been helpful
Query Optimization Feedback

SQL request → Query Optimization

Indexes Advised → V5R4
SQE Plan Cache → V5R4
SQE Plan Cache Snapshots → V5R4

Visual Explain

Detailed DB Monitor Data
Summarized DB Monitor Data
Debug Job Log Messages
Print SQL Information Messages
SQE Autonomic Indexes

• New V5R4 feature
• SQE optimizer watches query requests and learns
• **Autonomic** capability to create an index based on:
  – Observing *n* number of queries
  – Assessing the benefits of creating and using an index
  – Optimizer’s own index advice
• Indexes are:
  – **Temporary**
  – Maintained while any cursor is open
    • Maintenance is delayed when all cursors are closed
  – **Shared** between queries and jobs
  – Primarily created on smaller tables
  – Not used for statistics
• Allows additional queries to use SQE (ex. Sensitive cursors)
• Allows DB2 for System i to “tune” itself
## How Do I Get It? - Upgrade Paths (CQE → SQE)

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>V5R1</td>
<td>V5R2</td>
</tr>
<tr>
<td>V5R1</td>
<td>V5R2 + PTF SI07650</td>
</tr>
<tr>
<td>V5R1</td>
<td>V5R3</td>
</tr>
<tr>
<td>V5R2</td>
<td>V5R2 + PTF SI07650</td>
</tr>
<tr>
<td>V5R2</td>
<td>V5R3</td>
</tr>
<tr>
<td>V5R2 + PTF SI07650</td>
<td>V5R3</td>
</tr>
<tr>
<td>V5R2</td>
<td>V5R4</td>
</tr>
<tr>
<td>V5R2 + PTF SI07650</td>
<td>V5R4</td>
</tr>
<tr>
<td>V5R3</td>
<td>V5R4</td>
</tr>
<tr>
<td>V5R2</td>
<td>V6R1</td>
</tr>
<tr>
<td>V5R2 + PTF SI07650</td>
<td>V6R1</td>
</tr>
<tr>
<td>V5R3</td>
<td>V6R1</td>
</tr>
</tbody>
</table>
Migration Tips
Remember... Optimization Factors

- Server configuration
- Server attributes
- Version/Release/Modification
- SMP
- Database design
- Table sizes, number of rows
- Views and Indexes (Radix, EVI)
- Work management
- Static
- Dynamic
- Extended Dynamic Interfaces
- Job, Query attributes
- SQL Request
- Server performance

The Plan
Migration Tips

• Collection of feedback information before any changes can dramatically help problem determination later

• Any change to the environment in which queries run can affect the plans chosen (re-optimization on-demand)
  – Optimizer strategy or algorithm changes
  – Hardware or system changes
  – Changes to the underlying tables, indexes or statistics

• Implementing a good indexing strategy will help tremendously
  – Identify and eliminate full tables scans
  – Identify and eliminate temporary indexes
  – Identify and eliminate hash joins
  – [ibm.com/servers/enable/site/education/abstracts/indxng_abs.html](http://ibm.com/servers/enable/site/education/abstracts/indxng_abs.html)
Migration Tips

• Allow SQE to tune itself over time?
• Column statistics identified and collected over time
• Temporary indexes identified and created over time
• Use the various feedback mechanisms to watch behavior
  – System wide Index Advisor
  – SQE Plan Cache
  – SQE Plan Cache Snapshots
  – Detailed DB Monitor
  – Summary DB Monitor
• Remember what happens at IPL!
A Recommended Initial Column Statistics Strategy...

1. Identify and implement a good indexing strategy first!
   - Indexes are best for "narrow" local selection and joins
   - Indexes provide correlation
   - Indexes are best for a “first I/O” optimization goal
     • Read by key for grouping and ordering

2. Turn off automatic stats collection (QDBFSTCCOL = *NONE)

3. Execute queries, application, or SQL workload
   - Analyze feedback to find statistics that have been advised
     • Toggle on/off the QDBFSTCCOL system value to guarantee all advised stats will be logged in the feedback
     • DB monitor can be used to capture feedback
   - Check for stats that have already been collected on existing tables (on another system) through the automatic stats collection

4. If appropriate, identify and document stats
   - Generate a script to create the stats and/or use the iSeries Navigator interface to process and manage the results
   - Example utility: ibm.com/iseries/db2/statsscriptprocedure.html

5. Leave auto stats collection off, or turn it on?
Monitoring Usage Before and After Upgrade or PTFs

**SQL Performance Monitor** (Database Monitor)

- 2 Monitors available
  - Summary level
    - System i Navigator interface
  - Detail
    - STRDBMON command with pre-filtering
    - System i Navigator interface with pre-filtering
    - Visual Explain
    - Trace of SQL requests
- Captures SQL requests to the database and optimizer feedback
- Will give a summary of SQL statements, tables used, columns used, indexes used, indexes created, response times, etc.
- All information placed into a file so it can be queried
- Be careful of overhead 20-30% for STRDBMON
Monitoring Usage Before and After Upgrade or PTFs

**PRTSQLINF** command

Can be run against program, SQL package, or job

- SQL "Explain"
- Queries run through "extended dynamic" are stored in a package object on the server
- Look in your ODBC / JDBC data source setup for package name
- Will create a spooled file with the SQL access plan information
- For use after the SQL request is issued and the access plan created by the optimizer

**Visual Explain** via iSeries Access – System i Navigator

- "Diagram" of the query plan
- Recommendations via optimizer feedback
Monitoring Usage Before and After Upgrade or PTFs

**Performance Monitor** via System i Navigator - **Collection Services**

- Great for looking at how entire system is performing
- Won't give individual query response times
- Look for IOP/IOA utilization, disk response times, lock conflicts
- CPU utilization may not be very useful since system tries to use all CPU available
  - Based on access plan for query
- Great for determining whether a query is I/O or CPU bound
- Look for hourly, daily, weekly trends
  - Basis for HW, SW upgrades
- Use iSeries Navigator to manage monitors
Benchmarking and Proof of Concept Centers
Rochester, Minnesota USA and Montpellier, France

- Staffed with iSeries experts and more than 75 years of collective experience in performance and testing methodology, this center is able to provide any iSeries and Storage system required to stress, tune, and test applications, measure performance, and determine workload capacity. The result of this experience provides the information needed to make sound business and computing decisions. If a testing “Sandbox” is needed, this is the place!

- This center operates on a cost recovery basis. Fees are determined based on the length of time and the size and quantity of hardware required. Visit the web site below for the request form.

http://www.ibm.com/eserver/iseries/benchmark
DB2 for i SQL Performance Workshop

• 4 days of Lecture and Hands-on Lab Exercises

• See the following link for scheduled classes
  ibm.com/systemi/db2/db2performance.html
Conclusion

• Initially a limited amount of queries took advantage of SQE
• Many more SQL queries take advantage of SQE in V5R4/V6R1
• In general, many queries perform better, some perform the same
• Awareness and preparation are the keys to success
• SQE is the strategic engine for DB2 for i
Additional Information

- **DB2 for i Websites**
  - Home Page: [ibm.com/systems/i/db2](http://ibm.com/systems/i/db2)
  - DeveloperWorks Zone: [ibm.com/developerworks/db2/products/db2i5OS](http://ibm.com/developerworks/db2/products/db2i5OS)
  - Porting Zone: [ibm.com/servers/enable/site/db2/porting.html](http://ibm.com/servers/enable/site/db2/porting.html)

- **Newsgroups**
  - USENET: comp.sys.ibm.as400.misc, comp.databases.ibm-db2

- **Education Resources - Classroom & Online**
  - [ibm.com/systemi/db2/gettingstarted.html](http://ibm.com/systemi/db2/gettingstarted.html)
  - [ibm.com/partnerworld/wps/training/i5os/courses](http://ibm.com/partnerworld/wps/training/i5os/courses)

- **DB2 for i Publications**
  - Online Manuals: [ibm.com/systemi/db2/books.html](http://ibm.com/systemi/db2/books.html)
  - DB2 for i Redbooks ([http://ibm.com/redbooks](http://ibm.com/redbooks))
    - Preparing for and Tuning the SQL Query Engine on DB2 for i5/OS (SG24-6598)
    - OnDemand SQL Performance Analysis … in V5R4 (SG24-7326)
    - Modernizing iSeries Application Data Access (SG24-6393)

- **Questions?** rchudb@us.ibm.com