DB2 for i5/OS: Tuning for Performance

DB2 for i5/OS: Tuning for Performance

Jackie Jansen

Senior Consulting IT Specialist

jjansen@ca.ibm.com

August 2007

Agenda

- Query Optimization
- Index Design
- Materialized Query Tables
- Parallel Processing
- Optimization Feedback
- Visual Explain

Why Optimization?

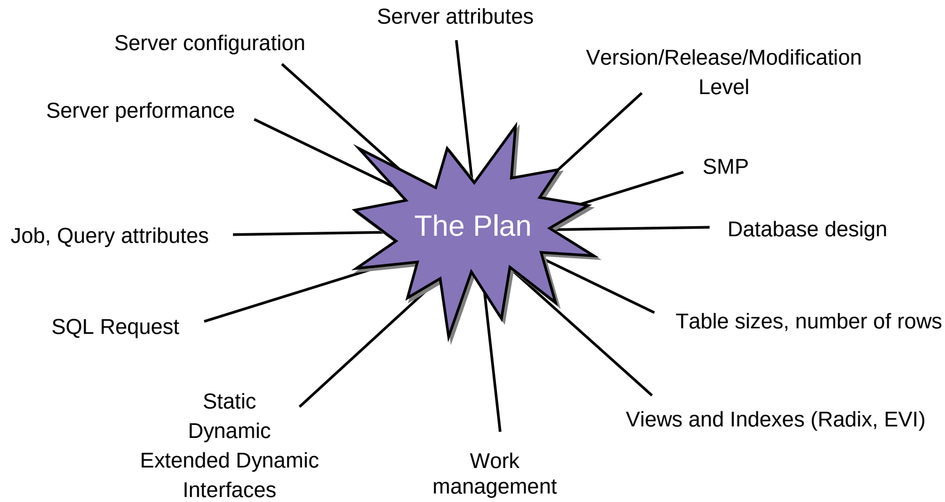
- The goal for the DB2 for i5/OS optimizer is to produce a plan that will allow the query to execute in the shortest time period possible
  - Optimization is based on time, not on resource utilization
- The DB2 for System i Optimizer performs "cost based" optimization
- "Cost" is defined as the estimated time it takes to run the request
- "Costing" various plans refers to the comparison of a given set of algorithms and methods in an attempt to identify the "fastest" plan
- The goal of the optimizer is to eliminate I/O as early as possible by identifying the best path to and through the data
- The optimizer has the ability and freedom to "rewrite" the query

The Optimization Goal

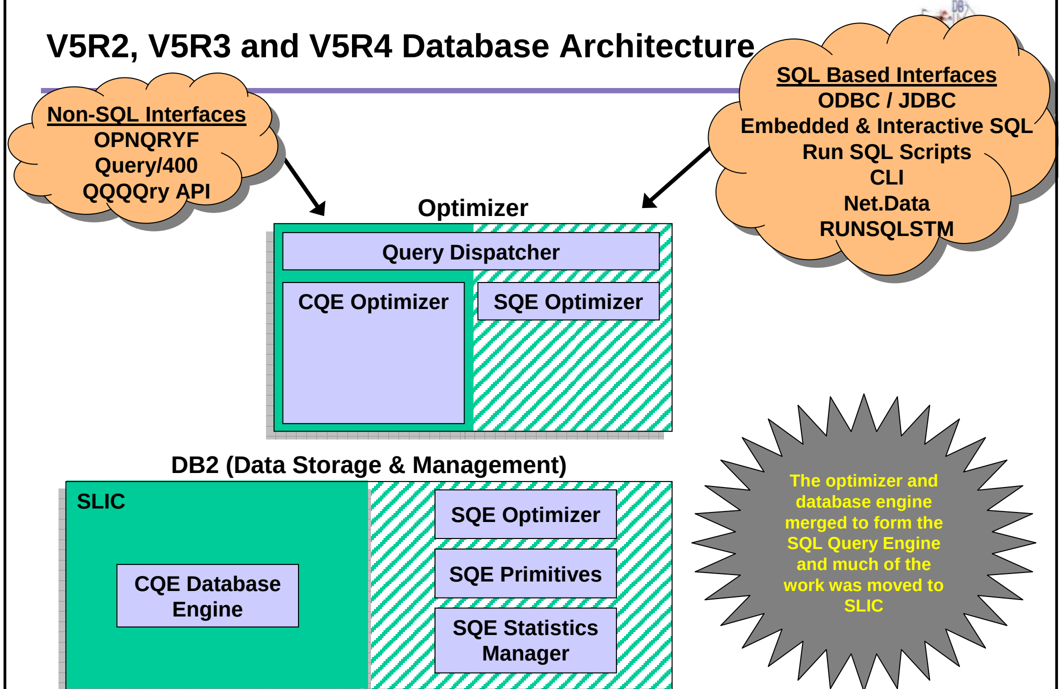
- **Set via optional SQL statement clause**
  - **OPTIMIZE FOR n ROWS**
  - **OPTIMIZE FOR ALL ROWS**
- **Set via QAQQINI options file**
  - ***FIRSTIO**
  - ***ALLIO**
- **Default for dynamic interfaces is First I/O**
  - **ODBC, JDBC, STRSQL, dynamic SQL in programs**
  - **CQE - 3% of expected rows**
  - **SQE - 30 rows**
- **Otherwise default is ALL I/O**
  - **Extended dynamic, RUNSQLSTM, INSERT + subSELECT, CLI, static SQL in programs**
  - **All expected rows**
- **Optimization goal will affect the optimizer's decisions**
  - **Use of indexes, SMP, temporary intermediate results like hash tables**
  - **Tell the optimizer as much information as possible**
  - **If the application fetches the entire result set, use *ALLIO**

Copyright 2007 - IBM Corporation

Optimization... the intersection of various factors



V5R2, V5R3 and V5R4 Database Architecture



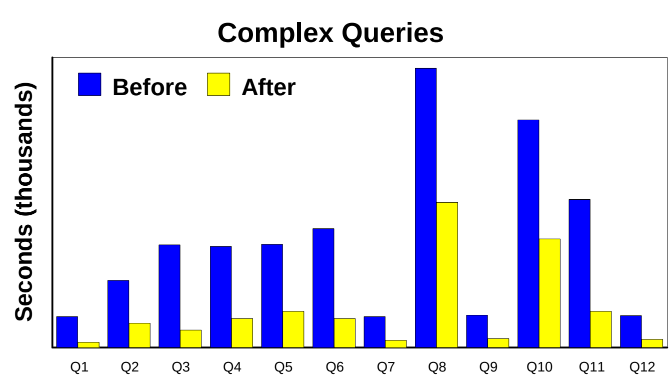
DB2 for i5/OS: Tuning for Performance

CQE and SQE by Release

	V5R2	V5R2	V5R3	V5R3	V5R4	V5R4
	CQE	SQE	CQE	SQE	CQE	SQE
LIKE Predicates	Y		Y			Y
Logical File references	Y		Y		Y	
UDTFs	Y		Y		Y	
LOB columns	Y		Y			Y
LOWER, TRANSLATE or UPPER scalar function	Y		Y		Y	
CHARACTER_LENGTH, POSITION, or SUBSTRING scalar function using UTF-8/16	Y		Y		Y	
Alternate sort sequences	Y		Y		Y	
Derived Logical Files over Physical (S/O)	Y		Y		Y	
Non-SQL queries (QQQQry API, Query/400, OPNQRYF)	Y		Y		Y	
ALWCPYDTA(*NO)	Y		Y			Y
Sensitive Cursor	Y		Y			Y
VIEWS, UNIONS, SubQueries	Y			Y		Y
INSERT, UPDATE, DELETE	Y			Y		Y
Star Schema Join queries	Y			Y		Y
Derived key and Select/Omit Logical Files on the table queried	Y		Y		Y	

The Query Dispatcher → SQE

- Only SQE optimizes
  - INTERSECT
  - EXCEPT
- QAQQINI parameter to ignore unsupported logical files
  - Ignore_Derived_Index = *YES



Selectivity Statistics and Data Skew

- Data Skew relates to the how VALUES are distributed in the DATA
- Ex: US State Column - 0.5% = "**North Dakota**", 50% = "**California**"

Table Size	Column Value	Actual Number of Rows	Estimated number of rows based on equal distribution
200,000	"North Dakota"	1,000	4,000
	"California"	100,000	4,000
200,000,000	"North Dakota"	1,000,000	4,000,000
	"California"	100,000,000	4,000,000

Choosing the "best" access plan is based on understanding the data

- Maintaining statistics real time in Tables and Indexes helps the optimizer select the "best" access method

SELECT CUSTNAME, CUSTID
FROM CUST_DIM
WHERE STATE = "**North Dakota**"

Probe the index and table

SELECT CUSTNAME, CUSTID
FROM CUST_DIM
WHERE STATE = "**California**"

Scan the table

Selectivity Statistics

- SQL to find the rows that contain the color purple, within a 1 million row DB table, when...
  - 300,000 rows contain the color purple

SELECT ORDER, COLOUR, QUANTITY

FROM ITEM_TABLE

WHERE COLOR = 'PURPLE'

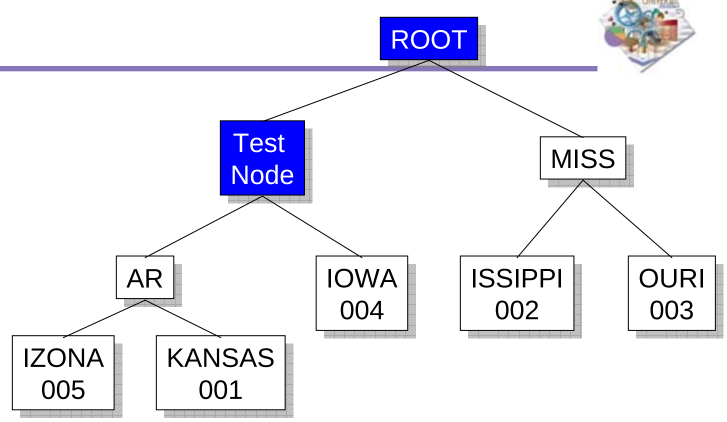
- Without index over color, assume 100,000 rows (10% default from =)
- With radix index over color, estimate 291,357 rows (read n keys)
- With EVI over color, actual 300,000 rows (read symbol table)
- With column stat over color, might be actual, might not...

DB2 for i5/OS

- Two types of indexing technologies are supported
  - ***Radix Index***
  - ***Encoded Vector Index***
- Each type of index has specific uses and advantages
- Respective indexing technologies compliment each other
- Indexes can provide RRNs and/or data
- The goals of creating indexes are:
  - Provide the optimizer the statistics needed to understand the data, based on the query
  - Provide the optimizer implementation choices, based on the selectivity of the query

Radix Index

Database Table	
001	**ARKANSAS**
002	**MISSISSIPPI**
003	**MISSOURI**
004	**IOWA**
005	**ARIZONA**
...	**...**



ADVANTAGES:
- Very fast access to a single key value
- Also fast for small, selected range of key values (low cardinality)
- Provides order

DISADVANTAGES:
- Table rows retrieved in order of key values (not physical order) which equates to random I/O's
- No way to predict which physical index pages are next when traversing the index for large number of key values

DB2 for i5/OS: Tuning for Performance

Encoded Vector Indexing (EVIs)

Indexing technology that can significantly improve performance, especially for star schema

- 10% to 30% faster index builds
- 1/3 to 1/16 the size
- 1/2 the time for index scans
- 1/3 the time for bit map generation

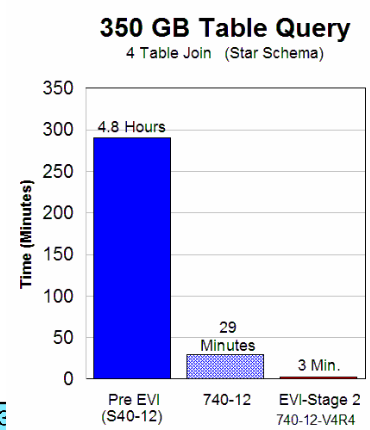
Symbol Table

Key Value	Code	First Row	Last Row	Count
Arizona	1	1	80005	5000
Arkansas	2	5	99760	7300
......				
Virginia	37	1222	30111	340
Wyoming	38	7	83000	2760

Vector

1	13	12	28	2	17	38	2	26	33

Row 1 Row 2 ....



Encoded Vector Indexes

- Create an EVI when
  - Local selection with selectivity of 20-70%
  - Mixed multiple local selection
    - Very good for ANDing and ORing
    - ie colour =x and size = y
    - colour= n and weight=10
  - Key columns with a relatively static set of values
- Create an EVI over
  - Single columns with low cardinality
  - Foreign key columns (star schema)
  - Columns should have low volatility

DB2 for i5/OS: Tuning for Performance

Indexing Strategy - Basic Approach

In general…

- A radix index is best when accessing a small set of rows and the key cardinality is high
- An encoded vector index is best when accessing a set of rows and the key cardinality is low

Radix Indexes

- Local selection columns
- Join columns

} Minimum

- Local selection columns + join columns
- Local selection columns + grouping columns
- Local selection columns + ordering columns
- Ordering columns + local selection columns

Encoded Vector Indexes

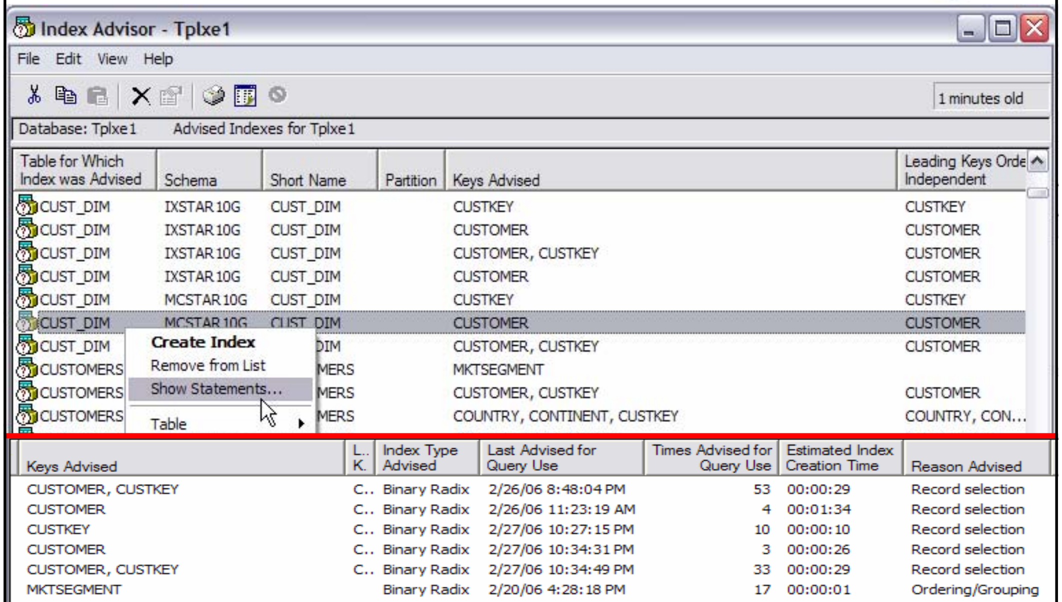
- Local selection column (single key)
- Join column (data warehouse - star or snowflake schema)

Index Advised – System wide

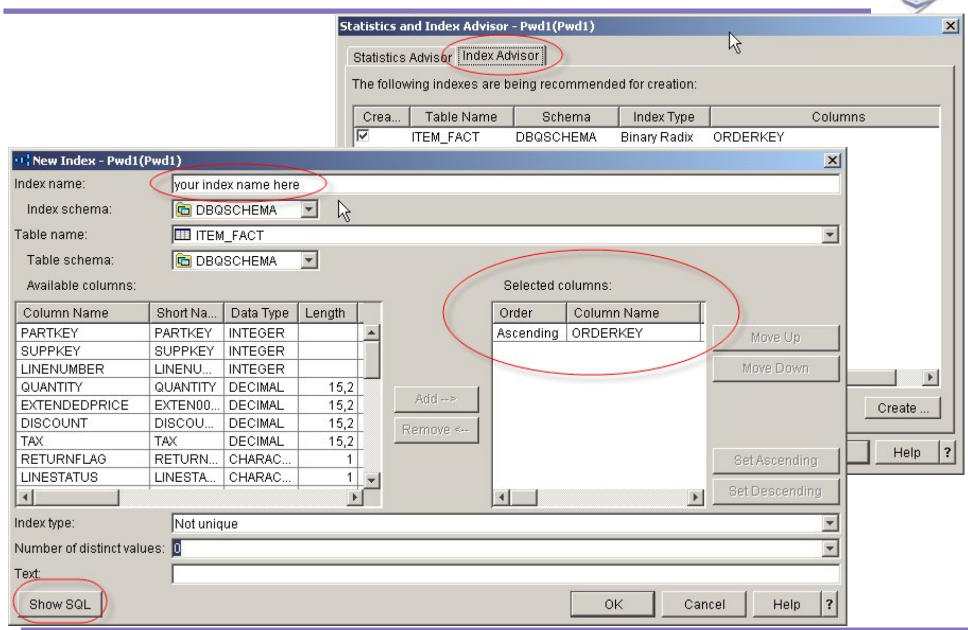
Wow!

- New V5R4 feature
- System wide index advice
  - Data is placed into a DB2 table (QSYS2/SYSIXADV)
  - Autonomic
  - No overhead
- CQE
  - Basic advice
  - Radix index only
  - Based on table scan and local selection columns only
  - Temporary index creation information also provides insight
  - CQE Visual Explain will try and tie pieces together to advice a better index
- SQE
  - Not complete, but much better
  - Radix and EVI indexes
  - Based on all parts of the query
  - Multiple indexes can be advised for the same query
- GUI interface via iSeries Navigator
  - Advice for System, or Schema, or Table
- Can create indexes directly from GUI

Index Advised – System wide



Visual Explain - Index & Stats Advisor



Materialized Query Tables (MQTs)

- Automatic Summary Tables
- Precomputing and Storing the Results of a Query
- Queries directed to base table(s) and optimizer will evaluate use of existing MQTs
- MQTs can be single table queries or inner-joins
- Not automatically updated with base table updates
- Require tuning and indexing just like base tables
- Require V5R3 **AND** latest DB Group PTFs
- Turn on via options in QAQQINI file

Parallel Processing

Allows a user to specify that queries should be able to use either I/O or CPU parallel processing as determined by the optimizer.

- Parallel processing is set on a per-job basis:
  - The parameter DEGREE on the **CHGQRYA CL** command.
  - The parmeter PARALLEL_DEGREE in the **QAQQINI** file.
  - The system value **QQRYDEGREE**.
  - Each job will default to the system value (*NONE is the default).
- I/O parallelism utilizes shared memory and disk resources by pre-fetching or pre-loading the data, in parallel, into memory.
- CPU parallelism utilizes one (or all) of the system processors in conjunction with the shared memory and disk resources in order to reduce the overall elapsed time of a query.
  - CPU parallelism is only available when DB2 Symmetric Multiprocessing is installed
  - CPU parallelism does not necessarily require multiple processors

Degree Parameter Values

- ***NONE**
  - No parallel processing is allowed for database query processing.
- *IO
  - Any number of tasks may be used when the database query optimizer chooses to use I/O parallel processing for queries. CPU parallel processing is not allowed. SQE always considers IO parallelism.
- *OPTIMIZE
  - The query optimizer can choose to use any number of tasks or threads for either I/O or CPU parallel processing to process the query. Use of parallel processing and the number of tasks or threads used will be determined with respect to the number of processors available in the system, this job's share of the amount of active memory available in the pool which the job is run, and whether the expected elapsed time for the query is limited by CPU processing or I/O resources.
- *MAX
  - The query optimizer can choose to use either I/O or CPU parallel processing to process the query. The optimizer will assume that all active memory in the pool can be used to process the query.
- *SYSVAL
  - Use current value of the system value QQRYDEGREE.
- *NBRTASKS nn
  - Specifies the number of tasks or threads to be used when the query optimizer chooses to use CPU parallel processing to process a query. I/O parallelism will also be allowed.
  - Used to manually control the degree value

SMP Considerations

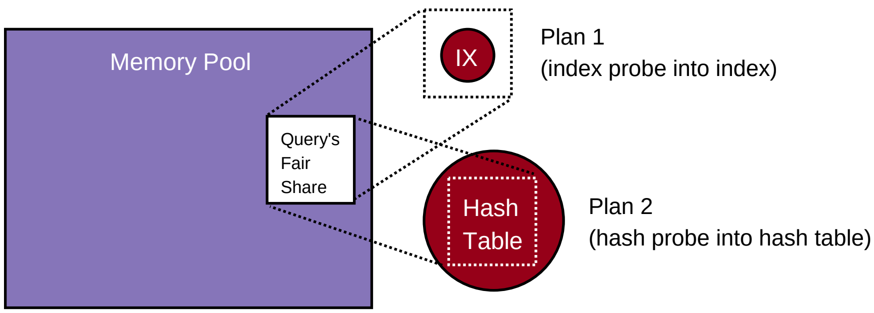
When and where to consider using database parallelism and SMP

- Application environments that can use and benefit from parallelism
  - SQL requests that use methods that are parallel enabled
  - Longer running or complex SQL queries
  - Longer running requests like index creation
  - Few or no concurrent users running in the same memory pool
  - Willing to dedicate most or all the resources to the specific SQL request(s)
- Computing resources
  - > 1 (physical) CPUs
  - 4-8GB memory per CPU
  - 10-20 disk units per CPU
  - 60% or less average CPU utilization during the time interval of the request
- Start with *OPTIMIZE and adjust the MAX ACTIVE number of the job's memory pool
- For single running jobs try *OPTIMIZE first, then try *MAX
- Run jobs in memory pools with paging option set to *CALC
- The optimization goal "ALL I/O" tends to allow SMP, while "FIRST I/O" does not

Beware of conflicts between the need for a high MAX ACTIVE setting for application processing, and the need for a low MAX ACTIVE setting for larger fair share of memory

Fair Share of Memory

- During optimization, the optimizer calculates an expected fair share of memory
- This keeps the optimizer from over commiting memory for a given query
- This allows the optimizer to consider more memory intensive methods
- The fair share value will affect what query plans are choosen



Fair Share of Memory

- CQE fair share = memory pool size / max-active value
- SQE fair share = memory pool size / min(max-active, max(avg-active, 5))
  - Average Active is:
  - 15 minute rolling average number of users when paging option set to *CALC
  - The number of unique users when paging option set to *FIXED
- If query degree is set to *MAX then fair share = entire pool size
- Max active value can be viewed and changed via:
  - WRKSYSSTS command
  - iSeries Navigator - Work Management - Memory Pools

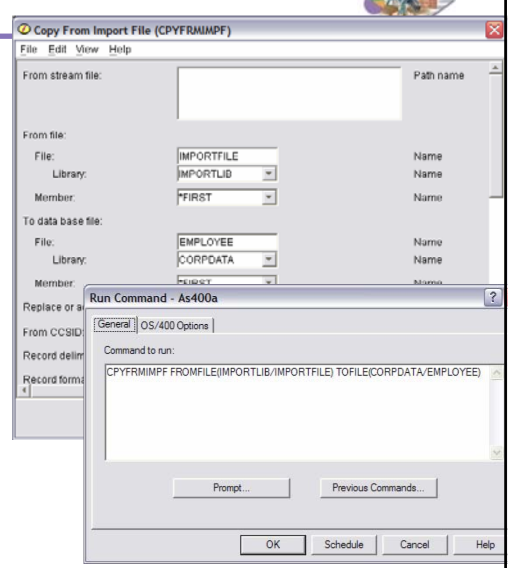
DB2 for i5/OS: Tuning for Performance

ODBC Performance Tips

- Lazy Close
  - Reuse open connections
  - Good for applications such as MS Access
- Data Compression
  - Enabled by default
  - For clients not CPU bound
- Block with a fetch of 1 row
  - Advanced option
  - Test, incompatible with some applications
- Record blocking
  - Default 32kb
  - For read only increase dramatically
- Query Optimization Goal (V5R4)
  - *ALLIO or *FIRSTIO
- Extended Dynamic
  - For subsequent requests of the same query

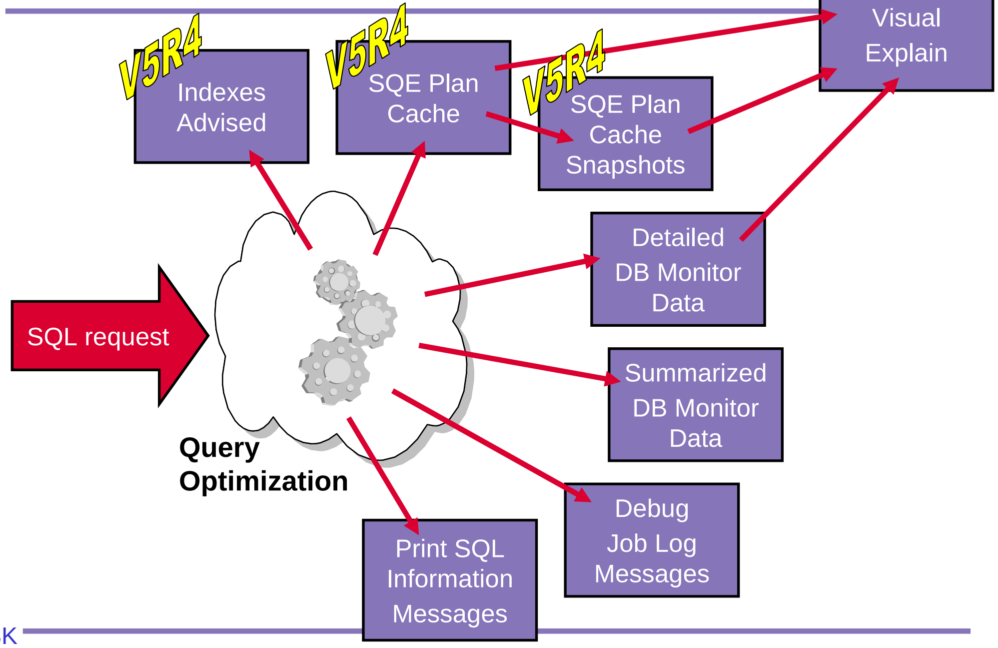
Database Loading

- Parallel Data load
  - Fully utilizes SMP capabilities
- CPYFRMIMPF and CPYTOIMPF CL commands
- Works with fixed format and delimited files
- Import from stream files (IFS), source files, tape files and more



CPYFRMIMPF FROMSTMF('~mydir/myimport.txt') TOFILE(MYLIB/MYTABLE) DTAFMT(*DLM) FLDDLM(',')

Query Optimization Feedback

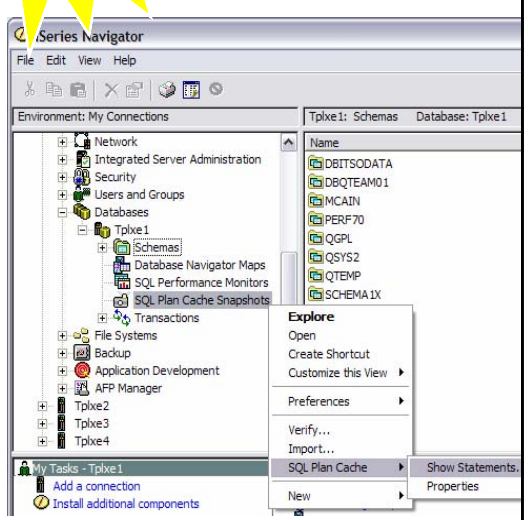


SK

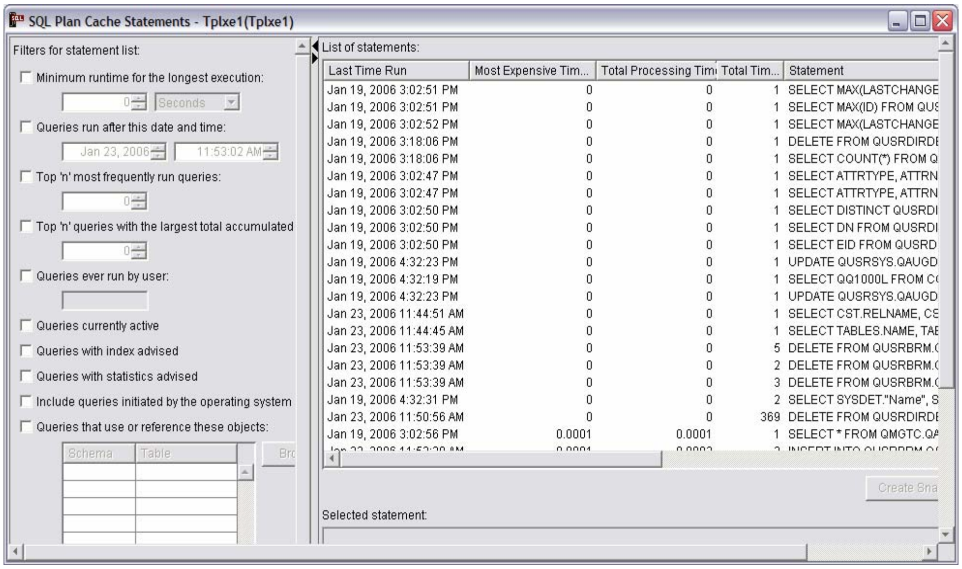
SQE Plan Cache

Cool!

- New V5R4 feature
- System wide information from the SQE Plan Cache
  - Automatic
  - No overhead
- SQE support only
- GUI interface via iSeries Navigator
  - Access
  - Filtering
  - Analysis by time, user, job, statement, etc.
  - Visual Explain
- Data is volatile
  - Information in the SQE Plan Cache is “live” and changing
  - SQE Plan Cache is cleared at IPL
- SQE Plan Cache is always available
  - No need to “start and stop” a tool or utility

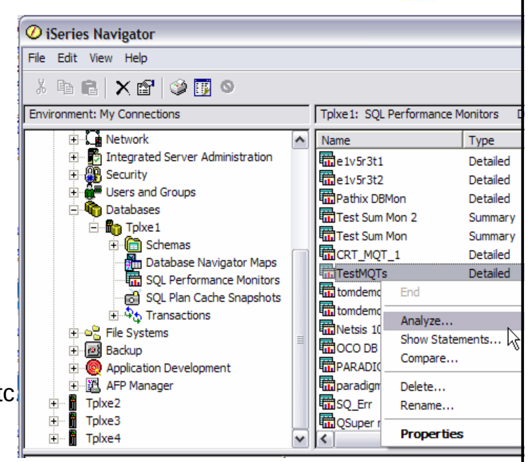


SQE Plan Cache – Show Statements



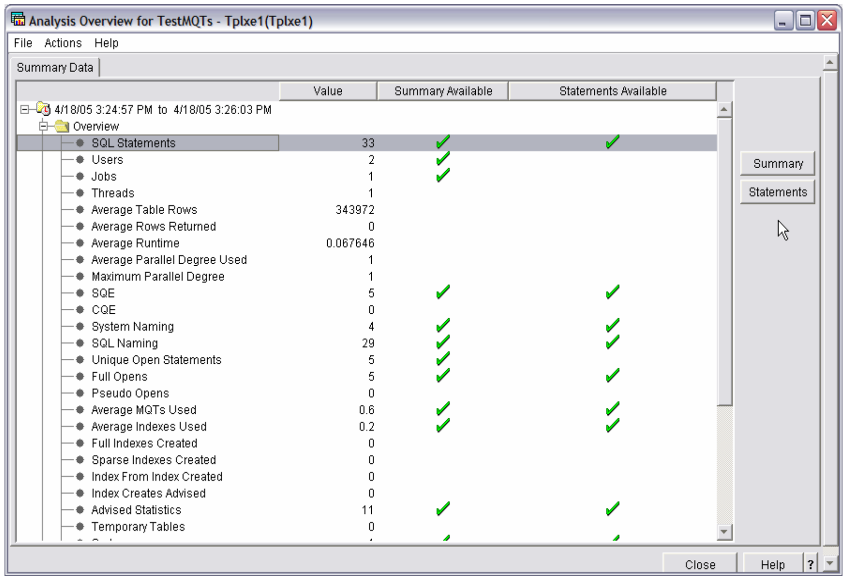
Detailed Database Monitor – SQL Trace

- Enhanced in V5R4
- Detailed information collected by the SQL "tracing" facility
  – Data is placed into a single DB2 table
  – Potentially high overhead
- CQE and SQE support
- Command interface – STRDBMON / ENDDBMON
- Connection attributes interface
- GUI interface via iSeries Navigator
  – Access
  – Pre-filtering and Post-filtering
  – Analysis by time, user, job, statement, etc.
  – Summary information via "dashboard"
  – Visual Explain



- Data is not volatile
  – Information from the optimizer and engine is "captured" at a point in time
- Additional analysis methods available like "before and after" comparisons

Detailed Database Monitor – SQL Trace



Visual Explain

The query access plan is diagrammed for the selected SQL statement

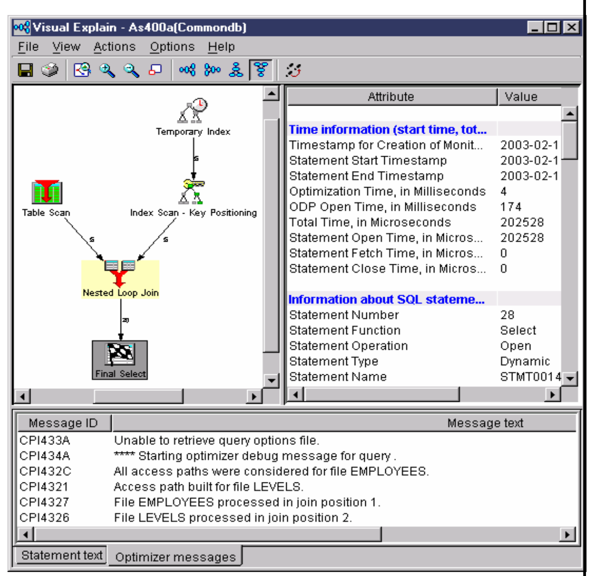
Stages of the access plan are shown as icons

- Detailed information for each stage
- Flyover help available

Several diagram customization Options

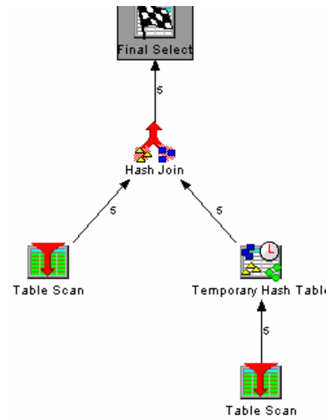
Highlight expensive icons and Paths

Optimizer messages shown



Visual Explain

- Enhanced in V5R4
- Graphical representation of query plan
  - Representation of the DB objects and data structures
  - Representation of the methods and strategy
  - Associated environmental information
  - **Advice on indexes and column statistics**
  - Highlighting of specific query rewrites
  - Highlighting of expensive methods
- CQE and SQE support
- GUI interface via iSeries Navigator
- Based on detailed optimizer information
  - SQE Plan Cache
  - SQE Plan Cache Snapshots
  - Detailed Database Monitor Data



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
- Remember what happens at IPL!

DB2 for i5/OS SQL and Query Performance Monitoring and Tuning Workshop

- The science of query optimization.
  - This topic covers the data access methods available to the DB2 for i5/OS Query Optimizer and the conditions in which the cost based optimizer chooses these methods.
- The art of query optimization.
  - Knowing how the query optimizer works, and what the database engine can do are the first steps in getting the most out of DB2 for i5/OS. This topic covers indexing strategies including Encoded Vector Indexes (EVI), join, sub query and view optimization techniques, etc.
- SQL performance techniques and considerations.
  - A must for the SQL application developer. Topics include understanding SQL Access Plans and Open Data Paths (ODP), effective use of blocking, optimal program compiler settings, etc.
- SQL Performance Tools and Analytical Methods.
  - These topics include in depth discussions of the Database Monitors, DB2 SMP (Symmetrical Multiprocessing) feature and parallelism, Query Governor, Index Advisor and others.
- In addition to the presentations above, several labs have been created to emphasize and demonstrate the concepts introduced in each topic. This course is intended for System i database designers, performance analysts, and application developers who are concerned about SQL and query performance. It is also highly recommended for individuals interested in SQL and query performance on the System i (AS/400).

http://www-03.ibm.com/servers/eserver/iseries/service/igs/db2performance.html

Questions & Answers