Common Luxemburg – March 22, 2007

DB2 for i5/OS
State of the Union for V5R4

Jos Vermaere
Systems Architect

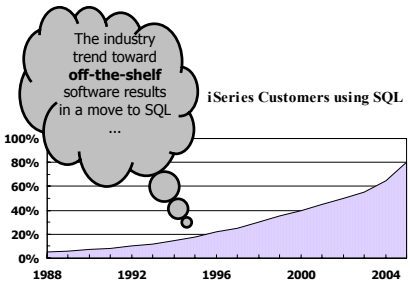
© 2007 IBM Corporation

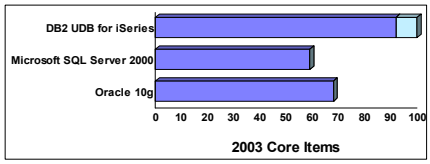
DB2 UDB for iSeries Strategic Initiatives

Openness - Industry Standard Support
- Accomodate ISVs
- Portability/Compatibility
- Flexibility

Continued LEADERSHIP in database technologies
- Consistency across DB2 family
- Shared R & D across IBM Labs

Continued Leveraging of iSeries Strengths
- Availability
- Scalability
- Usability - Total Cost of Ownership
- Application Flexibility





SQL 2003 Core Standard 100% Complete with V5R4!!

© 2007 IBM Corporation

Indexing Technology Revisited

© 2007 IBM Corporation

Indexing and Beyond

- **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 be used for statistics and implementation**
- **Indexes can provide RRNs and/or data**
- **Indexes are scanned or probed**
  - Probe can only occur on contiguous, leading key columns
  - Scan can occur on any key column
  - Probe and scan can be used together

© 2007 IBM Corporation

Index Probe vs. Scan

- **Probe** (key positioning) with leading, n contiguous key columns
  1
  1+2
  1+2+3

- **Scan** (test) with any other key columns
  2
  3
  2+3

Index Key Columns (ITEM_NO, COLOR, SIZE)

ITEM_NO	COLOR	SIZE
001	BLUE	SMALL
001	RED	LARGE
003	BLACK	SMALL
004	GREEN	MEDIUM

…WHERE COLOR = 'BLACK' AND ITEM_NO = 003

...WHERE SIZE = 'MEDIUM'

...WHERE ITEM_NO = 001 AND SIZE = 'LARGE'

© 2007 IBM Corporation

Radix Index

- **Index "tree" structure**
- **Key values are compressed**
  - Common patterns are stored once
  - Unique portion stored in "leaf" pages
  - Positive impact on size and depth of the index tree
- **Algorithm used to find values**
  - Binary search
  - Modified to fit the data structure
- **Maintenance**
  - Index data is automatically spread across all available disk units
  - Tree is automatically rebalanced to maintain an efficient structure
- **Temporary indexes**
  - Considered a temporary data structure to assist the DB engine
  - Maintained temporary indexes available in SQE V5R4

© 2007 IBM Corporation

Index Probe Example

Given an index on table EMPLOYEE keyed on STATE...

SELECT *
FROM EMPLOYEE
WHERE STATE = 'IOWA'

EMPLOYEE Index

STATE
...
IOWA (004)
IOWA (007)
IOWA (010)
IOWA (017)
KANSAS (011)
MISSISSIPPI (002)
MISSISSIPPI (013)
MISSOURI (003)
...

RRN

EMPLOYEE Table

RRN	STATE
001	ARKANSAS
002	MISSISSIPPI
003	MISSOURI
004	IOWA
005	ARIZONA
006	MONTANA
007	IOWA
008	NEBRASKA
009	NEBRASKA
010	IOWA
011	KANSAS
012	WISCONSIN
013	MISSISSIPPI
014	WISCONSIN
015	WISCONSIN
016	ARKANSAS
017	IOWA

Perform a probe into the range using the local selection value(s)

© 2007 IBM Corporation

Encoded Vector Index

- **Index for delivering fast data access in analytical and reporting environments**
  - Advanced technology from IBM Research
  - Used to produce dynamic bitmaps and RRN lists
  - Fast access to statistics to improve query optimizer decision making
- **Not a "tree" structure**
- **Can only be created through an SQL interface or iSeries Navigator GUI**

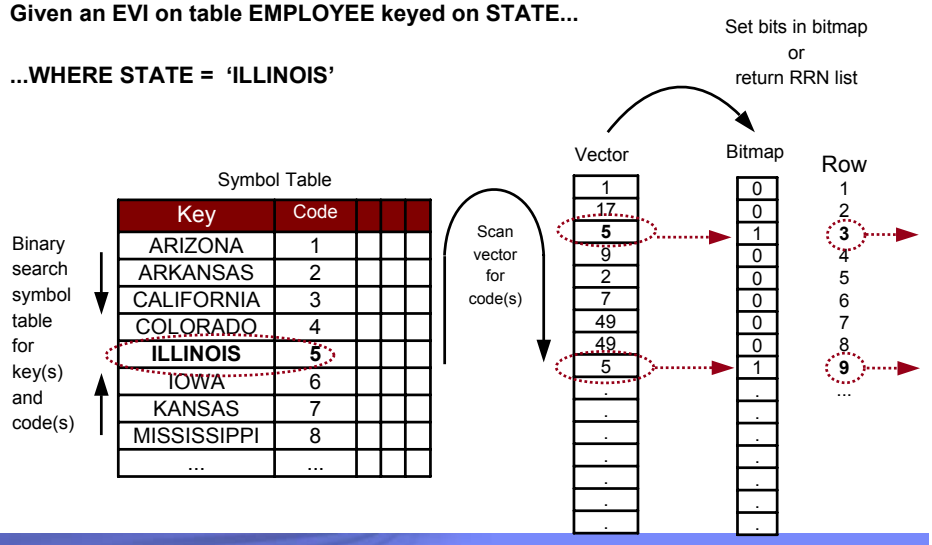
```
CREATE ENCODED VECTOR INDEX
   SchemaName/IndexName ON SchemaName/TableName
   (ColumnName)
   WITH n DISTINCT VALUES;
```


© 2007 IBM Corporation

Bitmap – RRN List Example

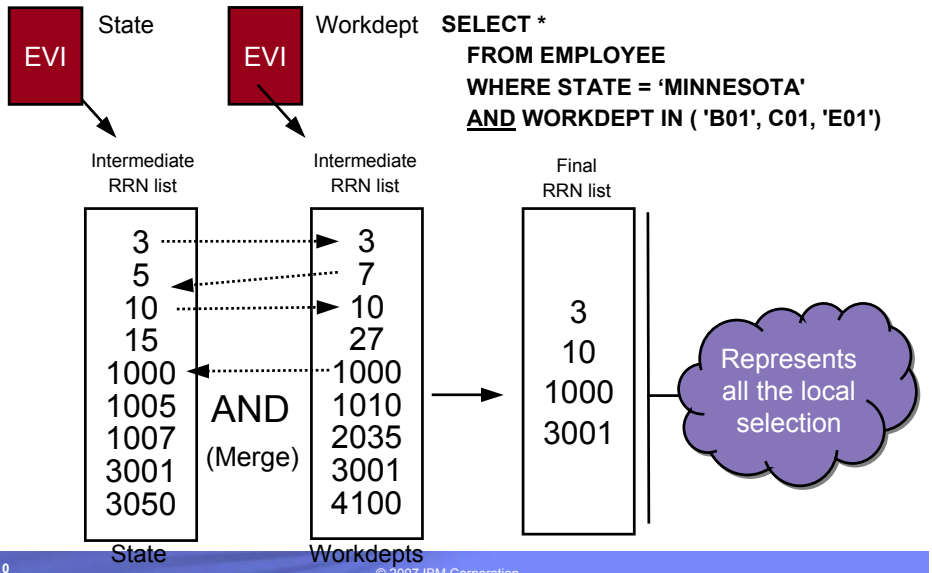
Given an EVI on table EMPLOYEE keyed on STATE...

...WHERE STATE = 'ILLINOIS'



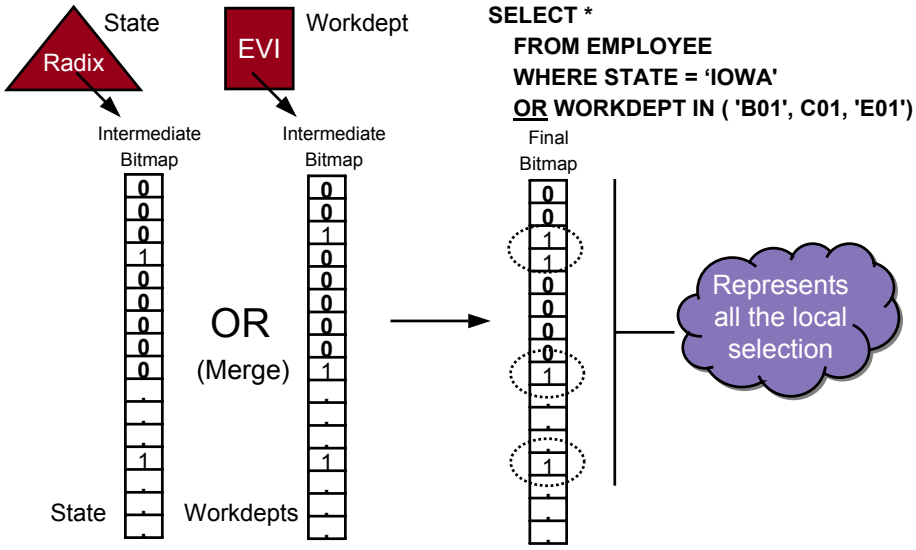
© 2007 IBM Corporation

Index ANDing/ORing



© 2007 IBM Corporation

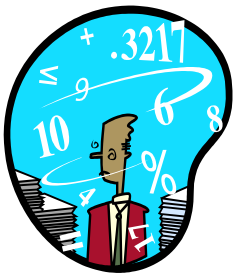
Index ANDing/ORing



© 2007 IBM Corporation

Cardinality – Radix Index or EVI?

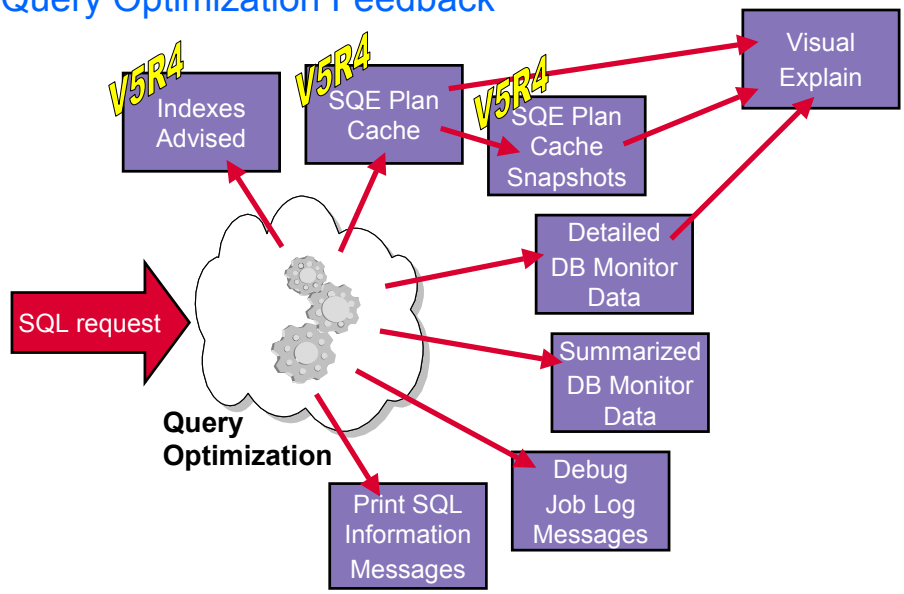
- *Cardinality*: **the number of elements in a set.**
  - High cardinality = large distinct number of values
  - Low cardinality = small distinct number of values



- **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
  - Understanding the data and query are key

© 2007 IBM Corporation

Query Optimization Feedback



© 2007 IBM Corporation

The Index Advise from the Optimizer

- **Both CQE and SQE provide index creation advice**
- **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**
  - Robust advice
  - Radix and EVI indexes
  - Based on all parts of the query
  - Multiple indexes can be advised for the same query
  - Some limitations

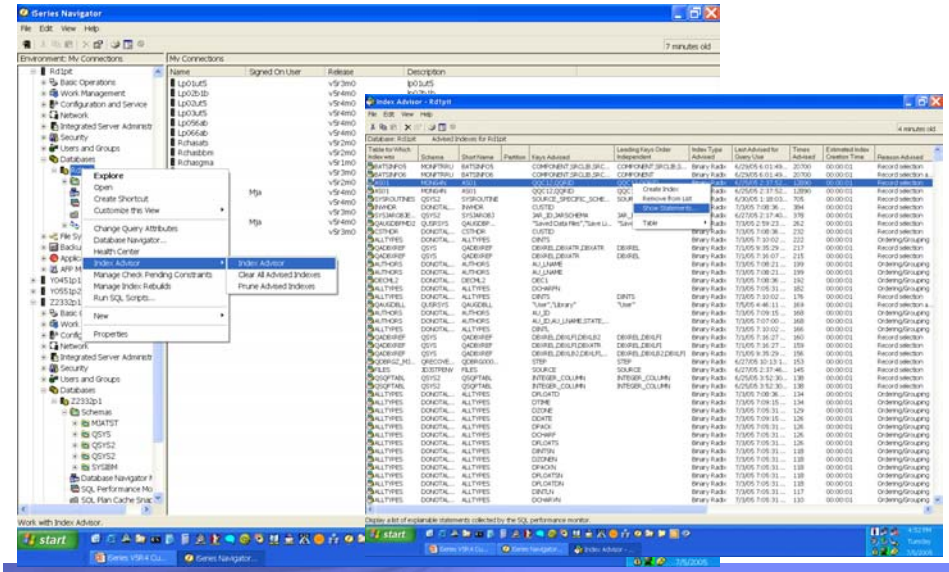
© 2007 IBM Corporation

System-wide Index Advise

- **New V5R4 feature**
  - Data is placed into a DB2 table (QSYS2/SYSIXADV)
  - Autonomic
  - No overhead
- **CQE and SQE support**
  - CQE only provides basic advice based on local selection predicates
  - SQE provides complex advice based on all parts of the query
    - Not complete, but much better
- **GUI interface via iSeries Navigator**
  - Advice for System, or Schema, or Table
- **System only adds (summary) rows, user must manage the data**
  - Options to clear or prune
- **Can create indexes directly from GUI**
  - Additional indexing analysis might be required to determine the optimal index

© 2007 IBM Corporation

Index Advisor



© 2007 IBM Corporation

Visual Explain

- **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

© 2007 IBM Corporation

Index Advise from Other Sources

- **SQE Plan Cache (V5R4)**
  - No direct index advice
  - Index advice via Snapshot data or Visual Explain
- **SQE Plan Cache Snapshot (V5R4)**
  - Enhanced SQE index advised
  - "3020" records to show multiple indexes for same table
  - Temporary index created
- **Detailed Database Monitor (V5R4)**
  - Enhanced SQE index advised
  - "3020" records to show multiple indexes for same table
  - Temporary index created
- **Summary Database Monitor**
  - No enhanced SQE index advised
  - Basic index advice
  - Temporary index created
- **Debug Messages in Job Log**
  - No enhanced SQE index advised
  - Basic index advice
  - Temporary index created
- **Print SQL Information**
  - No index advice
  - Temporary index created

© 2007 IBM Corporation

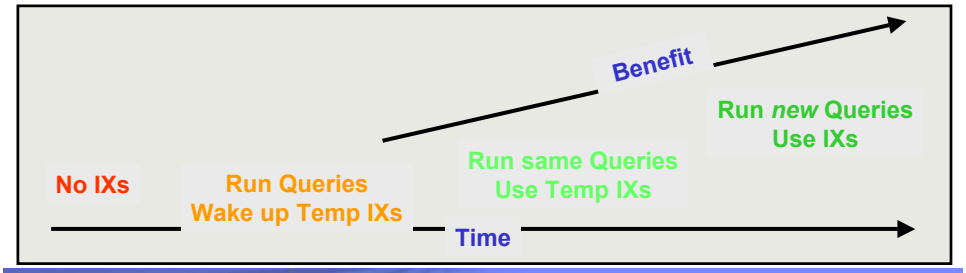
Autonomic Index Creation

- **Optimizer can have the DB Engine create a temporary index**
- **Both full and sparse indexes can be created**
- **Temporary indexes are not used for statistics**
- **Temporary indexes are *maintained***
- **CQE**
  - Temporary indexes are not reused and not shared
  - Usually a bottleneck in query performance
  - Can impact overall system performance
  - Can increase the amount of temporary storage used
- **SQE**
  - New feature in V5R4
  - Temporary indexes are reused and shared across jobs and queries
  - Creation is based on "watching" the query requests over time
  - Creation is based on optimizer's own index advice
  - Temporary index maintenance is delayed when all associated cursors closed

© 2007 IBM Corporation

Autonomic Index Creation

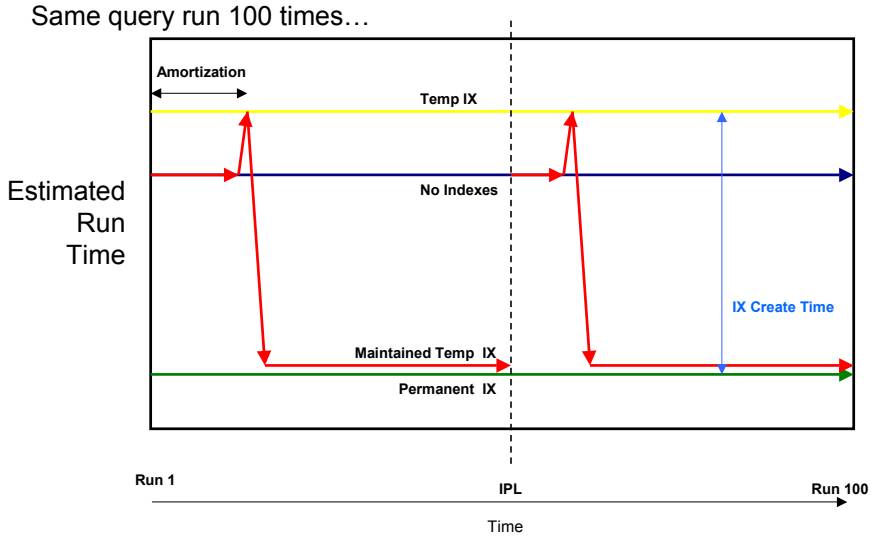
- **CQE temporary indexes represent a good opportunity for tuning**
  - Temp indexes are not shared or reused
- **SQE temporary indexes represent DB2 self tuning**
  - For the same set of queries, temp indexes are about the same as hash tables
  - Temp indexes are shared and reused, providing more benefit



© 2007 IBM Corporation

Autonomic Index Creation in Action

Same query run 100 times…



© 2007 IBM Corporation

Indexing Strategies

© 2007 IBM Corporation

Why Create Indexes?

- **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

✓**Accurate statistics means accurate costing**

✓**Accurate costing means optimal query plan**

✓**Optimal query plans means best performance**

© 2007 IBM Corporation

The Process of Identifying Indexes

- **Proactive method**
  - Analyze the data model, application and SQL requests
- **Reactive method**
  - Rely on optimizer feedback and actual implementation methods
  - Rely on SQE's ability to auto tune using temporary indexes
- **Understand the data being queried**
  - Column selectivity
  - Column cardinality
- **Separating complex queries into individual parts by table**
  - Selecting
  - Joining
  - Grouping
  - Ordering
  - Subquery
  - View

© 2007 IBM Corporation

Identifying Indexes: Basic Approach

- **Radix Indexes**
  - Local selection columns
  - Join columns
  - Local selection columns + join columns
  - Local selection columns + grouping columns
  - Local selection columns + ordering columns
  - Ordering columns + local selection columns

Minimum

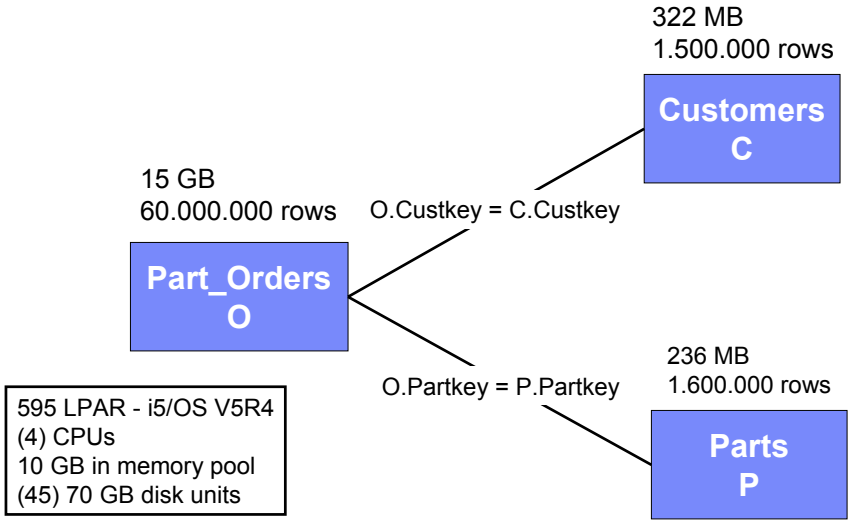
- **Encoded Vector Indexes**
  - Local selection column (single key)
  - Join column (data warehouse - star or snowflake schema)

© 2007 IBM Corporation

Indexing: A Case Study

© 2007 IBM Corporation

Three Tables, one System



© 2007 IBM Corporation

The Test Scenario

- **80 SQL requests from a single JDBC connection…**
  - 2 SETs
  - 53 SELECTs
  - 15 INSERTs
  - 5 UPDATEs
  - 15 DELETEs
  - 73 via SQE
  - 5 via CQE

- **Scenarios…**
  1. No indexes
  2. Indexes on join columns only
     - 4 radix indexes
  3. Indexes for selecting, joining, grouping, ordering
     - 13 radix indexes
     - 2 encoded vector indexes

© 2007 IBM Corporation

The Indexes

- **Indexes on join columns only**
  - ✓create index part_orders_ix1 on part_orders (custkey);
  - ✓create index part_orders_ix2 on part_orders (partkey);
  - ✓create index customers_ix1 on customers (custkey);
  - ✓create index parts_ix1 on parts (partkey);
- **Index for selecting, joining, grouping, ordering**
  - ✓create index part_orders_ix3 on part_orders (returnflag, custkey);
  - ✓create index part_orders_ix4 on part_orders (shipmode, custkey);
  - ✓create index part_orders_ix5 on part_orders (orderkey, linenumber, custkey);
  - ✓create index part_orders_ix6 on part_orders (orderkey, custkey);
  - ✓create index part_orders_ix7 on part_orders (returnflag, partkey);
  - ✓create index part_orders_ix8 on part_orders (shipmode, partkey);
  - ✓create index part_orders_ix9 on part_orders (orderkey, linenumber, partkey);
  - ✓create index customers_ix2 on customers (customer, custkey);
  - ✓create index parts_ix2 on parts (part, partkey);

  - ✓create encoded vector index part_orders_evi1 on part_orders (returnflag);
  - ✓create encoded vector index part_orders_evi2 on part_orders (shipmode);

© 2007 IBM Corporation

The Results

	Total Time	Max Time	Avg Time
All Indexes	23,547	2,493	0,076
Join Indexes	5.138,851	1.249,081	20,975
No Indexes	6.302,275	1.533,910	20,265

	Table Scans	Hash Group By	Hash Join	Temp Indexes
All Indexes	15	0	0	0
Join Indexes	42	6	4	4
No Indexes	97	19	17	12

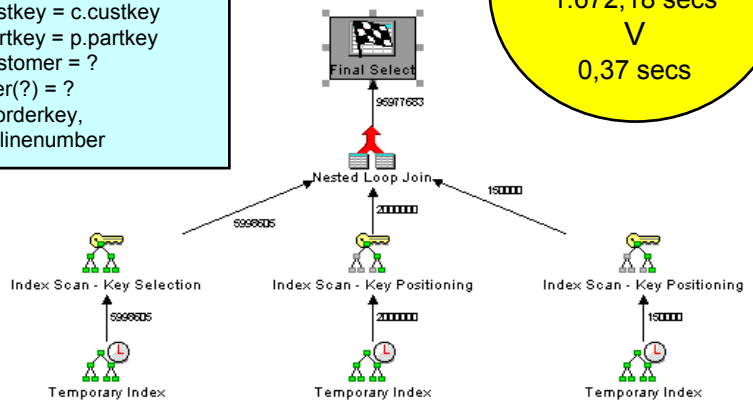
	Avg Async Reads	Avg Sync Reads
All Indexes	15	0
Join Indexes	42	6
No Indexes	97	19

© 2007 IBM Corporation

Indexing Strategy – Results: No Indexes

```
select *
from part_orders o,
        customers c,
        parts p
where o.custkey = c.custkey
and     o.partkey = p.partkey
and     c.customer = ?
and     upper(?) = ?
order by o.orderkey,
            o.linenumber
```

1.940,87 secs
V
1.672,18 secs
V
0,37 secs

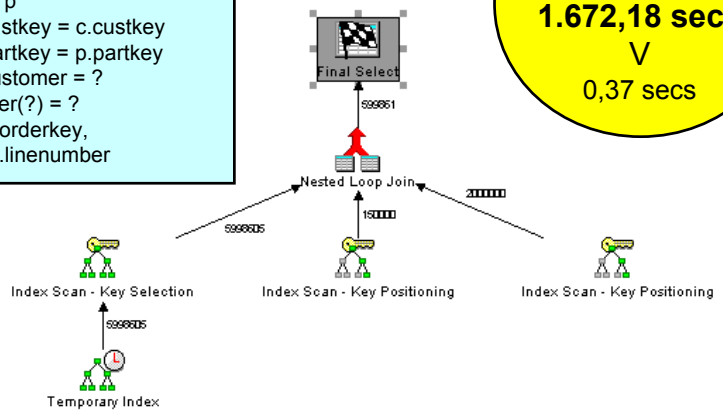


© 2007 IBM Corporation

Indexing Strategy – Results: Join Indexes

```
select *
from part_orders o,
        customers c,
        parts p
where o.custkey = c.custkey
and     o.partkey = p.partkey
and     c.customer = ?
and     upper(?) = ?
order by o.orderkey,
            o.linenumber
```

1.940,87 secs
V
1.672,18 secs
V
0,37 secs

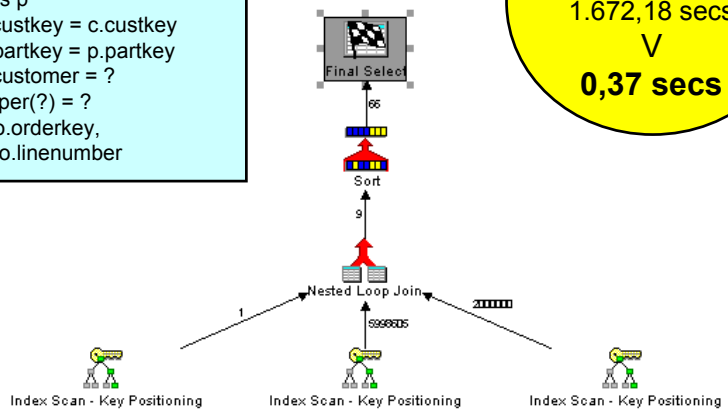


© 2007 IBM Corporation

Indexing Strategy – Results: All Indexes

```
select *
from part_orders o,
        customers c,
        parts p
where o.custkey = c.custkey
and     o.partkey = p.partkey
and     c.customer = ?
and     upper(?) = ?
order by o.orderkey,
            o.linenumber
```

1.940,87 secs
V
1.672,18 secs
V
0,37 secs



© 2007 IBM Corporation

Trade-off – Index Maintenance

- **For best query performance, create the appropriate indexes**
- **Eliminating table scans and temporary data structures will more than make up for index maintenance overhead**
- **Consider the number of indexes when doing *high* volume batch operations**
- **Consider parallel index maintenance for INSERTs**
  - DB2 SMP feature installed and enabled
- **Drop indexes when inserting into an empty table**
- **Consider dropping indexes when adding, changing or deleting more than 50% of the rows**
  - Use SMP to create indexes in parallel
  - (INSERT + INDEX CREATION) < (INSERT + INDEX MAINT)
- **Consider and watch out for access path protection (SMAPP)**

© 2007 IBM Corporation

Application Development

© 2007 IBM Corporation

Free-Format SQL in RPG

```
d getOrderCount      pi         10 i 0
d inCustNumber                   9 b 0 const
d outSqlState                    5 a
d outSqlMsg                    256 a
d ordercount         s          10 i 0
  /free
       outSqlMsg = *blanks;
       exec sql SELECT count(*) into :orderCount FROM orders
                     WHERE cust_no = :inCustNumber;
       outSqlState = sqlState;
  if %subst(sqlState:1:2) <> '00';
       exec sql GET DIAGNOSTICS CONDITION 1
       :outSqlMsg = MESSAGE_TEXT;
       orderCount = 0;
  endif;
  return orderCount;
  /end-free
```


© 2007 IBM Corporation

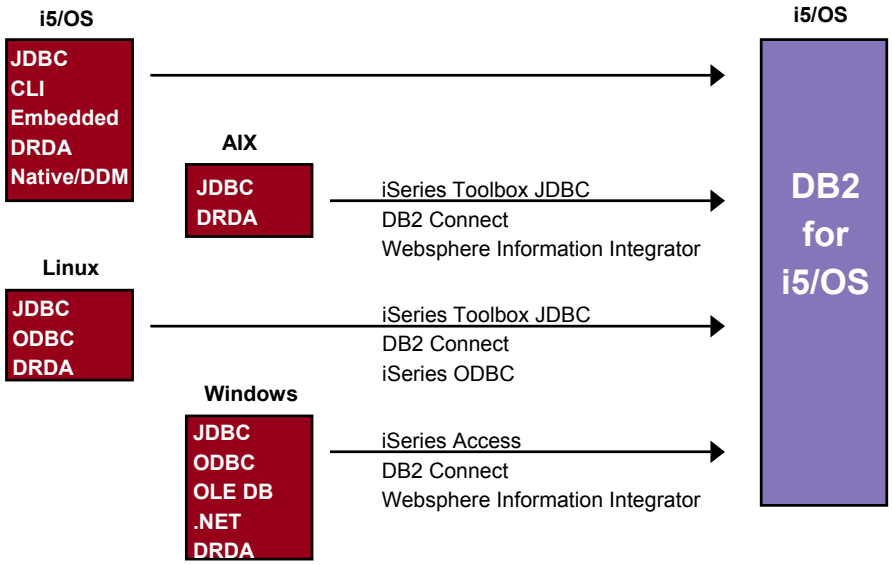
iSeries Access for Windows

- **.NET Provider Enhancements**
  - LOB column support**
  - System Naming & Library List support**
  - MS FW 2.0 Compatibility**
  - Intellisense support to aid programmers
  - Multiple active result sets on a connection
  - Customizable string processing

  **Available with latest V5R3 Service Pack
- **OLE DB Driver**
  - System Naming & Library List support
- **ODBC Driver**
  - Optimization Goal connection attribute
- **JDBC – Version 3 currency & performance**
  - Optimization Goal connection attribute
- **IBM EWLM support added to CLI, DRDA, .NET, ODBC, JDBC**
- **Driver support for Windows Vista (depending on availability)**

© 2007 IBM Corporation

DB2 for i5/OS as the Database Server



© 2007 IBM Corporation

Choosing the JDBC Driver for System i Access

- **Native JDBC Driver is the best choice to access DB2 for i5/OS**
  - Optimized for DB2 access - best overall performance
  - Use when running directly on i5/OS
  - Supports a range of specific properties
- **IBM Toolbox for Java**
  - Uses native i5/OS protocol - more efficient than non-native protocols
  - Use when accessing DB2 from a separate system/partition
  - Runs on any JVM
  - Supports extended dynamic SQL and other i5/OS properties
- **DB2 Universal JDBC Driver**
  - Considered when access to different DB2 platform is required
  - Requires a DRDA (DB2 Connect) connection to System i
  - Not as efficient as IBM Toolbox for Java driver

© 2007 IBM Corporation

More, Faster, Better

- **PHP DB2 data access with ibm_db2 extension in Zend Core for i5/OS**
  - http://devzone.zend.com/manual/view/page/ref.ibm-db2.html
- **CLI Enhancements**
  - Maximum Handles limit doubled to 160,000
  - SQLFetchScroll block-fetch & column-wise binding
  - Column-wise blocked insert binding
  - Optimization Goal connection attribute
  - Cursor Sensitivity Statement attribute
  - Improved XA documentation
  - New SQLGetInfo and SQLColAttributes options
  - Max rows attribute (SQL_ATTR_MAX_ROWS)
- **XA over DRDA**
- **Redesigned SQL Descriptor Area (SQLDA)**
  - Support for longer column names
  - Faster internal processing

© 2007 IBM Corporation

More, Faster, Better, Bigger …

- **Bigger & Badder Limits**
  - 2 MB SQL statement maximum
  - 1000 tables per query (DML Only)
  - 128 byte Column Names
  - 1024 parameters for Stored Procedure
  - 32K index keys & ORDER BY
- **New SET SESSION Authorization statement**
  - Better SQL auditing with the ability to supply actual user of remote connections
  - New SESSION USER & SYSTEM USER special registers
  - Considerations
    - *ALLOBJ authority required to execute
    - DB2 resources are freed, so performance can be impacted
    - Other settings such as SQL Path may need to be reset

© 2007 IBM Corporation

Stored Procedures

- **DB2 Expression Evaluator for faster SQL Procedural language, existing objects must be recreated**
- **Easier authority setup with support for DFTRDBCOL & DYNDFTCOL attributes**
  - Useful when porting from other DBMS's
  - Also available on V5R2 & V5R3 with latest Database Group PTF
- **Simpler maintenance with new ALTER PROCEDURE statement**

```
ALTER PROCEDURE Increase_Level
REPLACE
BEGIN
    DECLARE CurLvl INT;
    SELECT edlevel INTO CurLvl FROM emptbl
        WHERE empno=Emp#;
    IF NwLvl > CurLvl THEN
        UPDATE emptbl SET edlevel=NwLvl, salary=salary + (salary*0.10)
                WHERE empno=Emp#;
    END IF
END
```

- **Plans to support debug environment for DB2 Development Center**

© 2007 IBM Corporation

DB2 and SQL

© 2007 IBM Corporation

OLAP Expressions – ROW_NUMBER

- **ROW_NUMBER computes a sequential number for the rows in the final result set**

```
SELECT ROW_NUMBER() OVER (
       ORDER BY workdept, lastname) AS  lastname, salary
       FROM employee ORDER BY workdept, lastname
```

NBR	LASTNAME	SALARY
1	HAAS	52750.00
2	HEMMINGER	46500.00
3	LUCCHESSI	46500.00
4	O'CONNELL	29250.00
5	ORLANDO	29250.00

```
SELECT workdept, lastname, hiredate,
  ROW_NUMBER() OVER
                 workdept
  ORDER BY hiredate) AS Nbr
  FROM employee
  ORDER BY workdept, hiredate
```

WORKDEPT	LASTNAME	HIREDATE	NBR
A00	LUCCHESSI	1958-05-16	1
A00	O'CONNELL	1963-12-05	2
A00	HAAS	1965-01-01	3
A00	HEMMINGER	1965-01-01	4
A00	ORLANDO	1972-05-05	5
B01	THOMPSON	1973-10-10	1
C01	QUINTANA	1971-07-28	1
C01	KWAN	1975-04-05	2

© 2007 IBM Corporation

OLAP Expressions: RANK and DENSE_RANK

- **RANK & DENSE_RANK for highlighting data attribute – independent of the result set sorting**

```
SELECT empno, lastname, salary+bonus AS TOTAL_SALARY,
      RANK() OVER (ORDER BY salary+bonus DESC) AS Salary_Rank
      FROM employee WHERE salary+bonus > 30000 ORDER BY lastname
```

SELECT empno, lastname, salary+bonus AS TOT...

EMPNO	LASTNAME	TOTAL_SALARY	SALARY_RANK
000050	GEYER	40975.00	5
000010	HAAS	53750.00	1
200010	HEMMINGER	47500.00	2
000090	HENDERSON	30350.00	
200220	JOHN	30440.00	
000030	KWAN	39050.00	6
000110	LUCCHESSI	47400.00	3
000220	LUTZ	30440.00	
000070	PULASKI	36870.00	7
000060	STERN	32750.00	8
000020	THOMPSON	42050.00	4

Dense_Rank() Output

SALARY_RANK
5
1
2
6
3
7
8
4

© 2007 IBM Corporation

Recursive SQL Support

- **Recursive Common Table Expressions**
  - Useful for navigating tables where rows are inherently related to other rows in same table - Bill of Materials, Organizational Hierarchies, …
  - Example:

```
WITH emp_list (level, empid, name) AS
    (SELECT 1, empid, name  FROM emp
      WHERE name = 'Carfino'
    UNION ALL
     SELECT o.level + 1, next_layer.empid, next_layer.name
      FROM emp as next_layer, emp_list o
       WHERE o.empid = next_layer.mgrid )
SELECT level, name FROM emp_list
```

1 - Initializing the query

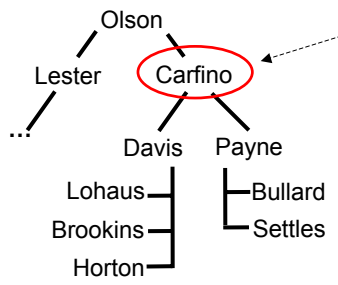
2 – Recursive reference to the next level

3 – Start the query & return final results

© 2007 IBM Corporation

Recursive SQL Support - Initialization

SELECT 1, empid, name FROM emp
WHERE name = 'Carfino'

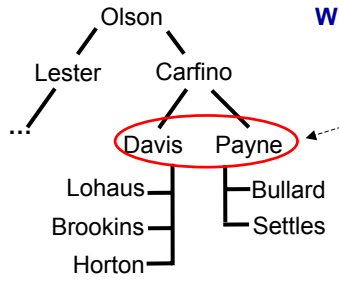



```
CREATE TABLE emp(
        empid INTEGER PRIMARY KEY,
        name  VARCHAR(10),
        salary DECIMAL(9, 2),
        mgrid INTEGER)
```


© 2007 IBM Corporation

Recursive SQL Support – Pass # 1

SELECT o.level + 1, next_layer.empid,
next_layer.name
FROM emp as next_layer, emp_list o
WHERE o.empid = next_layer.mgrid

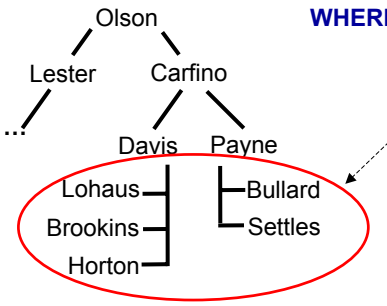



```
CREATE TABLE emp(
        empid INTEGER PRIMARY KEY,
        name  VARCHAR(10),
        salary DECIMAL(9, 2),
        mgrid INTEGER)
```


© 2007 IBM Corporation

Recursive SQL Support – Pass # 2

```
SELECT o.level + 1, next_layer.empid,
      next_layer.name
  FROM emp as next_layer, emp_list o
 WHERE o.empid = next_layer.mgrid
```

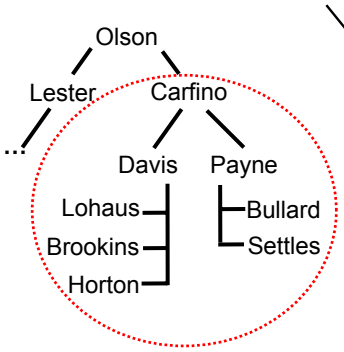



```
CREATE TABLE emp(
        empid INTEGER PRIMARY KEY,
        name  VARCHAR(10),
        salary DECIMAL(9, 2),
        mgrid INTEGER)
```


© 2007 IBM Corporation

Recursive SQL Support – Final Results

```
WITH emp_list (level, empid, name) AS
  (SELECT 1, empid, name  FROM emp
      WHERE name = 'Carfino'
   UNION ALL
      SELECT o.level + 1, next_layer.empid, next_layer.name
                FROM emp as next_layer, emp_list o
                WHERE o.empid = next_layer.mgrid )
SELECT level, name FROM emp_list
```



WITH emp_list(...

00001	NAME
1	Carfino
2	Payne
2	Davis
3	Settles
3	Bullard
3	Lohaus
3	Horton
3	Brookins

© 2007 IBM Corporation

Recursive Support for Bill of Materials

with rpl (level, parent, pdesc, child, cdesc,childqty) as (select 1, root.par...

Level	Parent	Description	Component	Description	Quantity in Assembly
1	11L5441	Power Assembly ...	09H4112	Power Supply ...	1
1	11L5441	Power Assembly ...	54K6942	Shipping Instruction...	1
2	09H4112	Power Supply ...	1789225	Fuse 12 AMP ...	2
2	09H4112	Power Supply ...	1586237	Power Cord 6 ft. ...	2
2	09H4112	Power Supply ...	18H1588	Support Bracket ...	1
2	09H4112	Power Supply ...	15H1200	CIF Tool	1
2	09H4112	Power Supply ...	578K211	Slider Long sleeve ...	2
2	09H4112	Power Supply ...	57G3289	Rail	4
3	1586237	Power Cord 6 ft. ...	1584451	Power Cord Retaine...	2
3	18H1588	Support Bracket ...	47P5281	Screw 8 mm ...	12
3	18H1588	Support Bracket ...	47J5213	Nut 8 mm	12
3	15H1200	CIF Tool ...	15H2900	Language Selector ...	1
3	578K211	Slider Long sleeve...	15H1200	CIF Tool	1
3	578K211	Slider Long sleeve...	09H2278	Filler Panel	1
3	578K211	Slider Long sleeve...	09K2557	Filler Panel Spring ...	4
4	15H1200	CIF Tool ...	15H2900	Language Selector ...	1
4	09H2278	Filler Panel ...	21L5554	Front Cover	1
4	09H2278	Filler Panel ...	29L5454	Front Cover Clip ...	4

```
WITH rpl (level, parent, pdesc, child, cdesc,childqty) AS
 (SELECT 1, root.parent, root.pdesc, root.child, root.cdesc, root.childqty
   FROM explprod as root
          WHERE root.parent = '11L5441'
      UNION ALL
        SELECT level+1, low.parent, low.pdesc, low.child, low.cdesc, low.childqty
          FROM rpl high, explprod low
          WHERE high.child = low.parent)
SELECT level as "Level", parent as "Parent", pdesc as "Description", child as "Component",
cdesc as "Description", childqty as "Quantity in Assembly"
         FROM rpl;
```


© 2007 IBM Corporation

Recursive SQL Support - Considerations

- **Breadth First is default processing order, Search clause can be used to change processing order – if the Order By clause also specified**
  - Search clause adds extra overhead, only use as required

```
WITH emp_list(level,empid, name) AS
   (SELECT 1,empid, name FROM emp
     WHERE name = 'Carfino'
   UNION ALL
     SELECT level+1, next_layer.empid, next_layer.name
      FROM emp as next_layer, emp_list p
      WHERE p.empid = next_layer.mgrid)
SEARCH DEPTH FIRST BY empid SET seqcol
SELECT level,name FROM emp_list ORDER BY seqcol
```

Depth Search Output

WITH emp_list(lev...

00001	NAME
1	Carfino
2	Davis
3	Brookins
3	Lohaus
3	Horton
2	Payne
3	Bullard
3	Settles

© 2007 IBM Corporation

Recursive SQL Support - Considerations

- **If table data has cyclic relationships, need use CYCLE clause to prevent never ending queries**

```
WITH destinations (departure, arrival, itinerary) AS
 (SELECT f.departure, f.arrival, CAST (f.departure ||'->'|| f.arrival AS VARCHAR(200))
    FROM flights f
    WHERE f.departure ='Rochester'
  UNION ALL
   SELECT r.departure, b.arrival, CAST (r.itinerary ||'->'|| b.arrival AS VARCHAR(200))
    FROM destinations r,flights b
    WHERE r.arrival =b.departure )
   CYCLE arrival SET cyclic_data TO '1 ' DEFAULT '0 '
SELECT departure, arrival, itinerary, cyclic_data FROM destinations
```

WITH destinations (departure, arrival, itinerary) AS (SELECT f.departure, f.arrival, CAST.....

DEPARTURE	ARRIVAL	ITINERARY				CYCLIC_DATA
Rochester	Chicago	Rochester	->Chicago			0
Rochester	Minneapolis	Rochester	->Minneapolis			0
Rochester	Brussels	Rochester	->Chicago	->Brussels		0
Rochester	New York	Rochester	->Chicago	->New York		0
Rochester	Minneapolis	Rochester	->Chicago	->Minneapolis		0
Rochester	Seatle	Rochester	->Chicago	->Seatle		0
Rochester	London	Rochester	->Chicago	->Brussels	->London	0
Rochester	Chicago	Rochester	->Chicago	->Brussels	->Chicago	1
Rochester	Luxembourg	Rochester	->Chicago	->Brussels	->Luxembourg	0

© 2007 IBM Corporation

SELECT Enhancements

- **Subquery and Scalar Fullselect**

```
UPDATE MyExchangeRates m SET conversion_rate=
        (SELECT rate FROM EuropeRates e WHERE e.ctryid = m.ctry_id
      UNION
        SELECT rate FROM AsiaRates a WHERE a.ctrycode = m.ctry_id)/100
```

- **Multi-column predicates**
  - WHERE (c1,c2) IN (SELECT a,b FROM…)
  - WHERE (c1,c2) = (SELECT a,b FROM…)
- **Exclusive Lock with SELECT**

```
SELECT * FROM orders
   WHERE order_id = 'E5100'
   WITH RS USE AND KEEP EXCLUSIVE LOCKS
```


© 2007 IBM Corporation

Richer Toolbox of SQL Functions

- **Triple DES Data Encryption – ENCRYPT_TDES**
- **Date/Time Processing**
  - LAST_DAY, NEXT_DAY, ADD_MONTHS, VARCHAR_FORMAT
- **Statistical Processing**
  - STDDEV_SAMP & VARIANCE_SAMP
- **Miscellaneous**
  - GENERATE_UNIQUE

```
CREATE TABLE EMP_UPDATE
    (UNIQUE_ID VARCHAR(13) FOR BIT DATA,
    EMPNO CHAR(6),
    TEXT VARCHAR(1000))
INSERT INTO EMP_UPDATE VALUES (GENERATE_UNIQUE(), '000020', 'Update entry
1...')
```

  - RAISE_ERROR

```
SELECT emp_name,
  CASE job_type
   WHEN 1 THEN 'Programmer'
   WHEN 2 THEN 'Administrator'
   WHEN 3 THEN 'Project Manager'
   WHEN 4 THEN 'Manager'
  ELSE RAISE_ERROR('70001', 'Invalid JobType') END
FROM employee
```


© 2007 IBM Corporation

INSTEAD OF Triggers

- **New Trigger Type can be used to change the semantics of INSERTs, UPDATEs, & DELETE operations against a view – only can be defined over an SQL view**
  - Many Views are read-only due to derivations, joins, grouping, etc
  - IOTs useful in setting up encryption to be semi-transparent
  - V5R4 support builds on base IOT support delivered via V5R3 PTF (ibm.com/iseries/db2/iot.html) by providing support for join views, etc.

```
CREATE VIEW empdept AS
      SELECT empno, firstname, lastname, deptname FROM employee, department
      WHERE workdept=deptno

CREATE TRIGGER UpdateJoin
 INSTEAD OF UPDATE ON empdept
 REFERENCING OLD ROW AS o NEW ROW AS n
 FOR EACH ROW
BEGIN
  UPDATE employee
     SET empno=n.empno, firstname=n.firstname, lastname=n.lastname
     WHERE empno=o.empno;
  UPDATE department SET deptname = n.deptname WHERE deptname=o.deptname;
END
```


© 2007 IBM Corporation

Miscellaneous Enhancements

- **ANS Timestamp Format - '2005-06-13 01:22:03.000444'**
- **Support for LABEL ON INDEX**
- **Optional Journaling for SQL Tables – NOT LOGGED INITIALLY**

```
CREATE TABLE new_materials
 AS (SELECT * FROM materials WHERE YEAR(item_date)=2006)
 WITH DATA NOT LOGGED INITIALLY
```

- **Ability to control i5/OS Record Format Name on SQL Create Table**

```
CREATE TABLE registeredUser (userid VARCHAR(50), active CHAR(1),
  registration_date DATE)
   RCDFMT reguser
```

- **Simpler SQL & CL Integration with new qcmdexc stored procedure**

```
CALL qcmdexc('ADDLIBLE DBLIB2', 15)
```

instead of `CALL qcmdexc ('ADDLIBLE DBLIB2', 0000000015.00000)`

- **Syntax Flagger for SQL Core Standard**

© 2007 IBM Corporation

Performance enhancements

- **SQL Query Engine (SQE) Enhancements**
  - Support for LIKE, LOB columns, SUBSTR, and Sensitive Cursors
  - Enhanced Partitioned Table optimization
  - Autonomic Indexes
- **Faster SQL Procedural language with Expression Evaluator**
  - More details - http://ibm.com/servers/enable/site/education/wp/113c2/113c2.pdf
- **Faster XML Extenders & New Redbook**
  - The Ins & Outs of XML and DB2 for i5/OS
    http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247258.html?Open
- **Transaction Manager Enhancements**
  - Parallel Rollback Unlock
  - Soft Commit - Reduce disk forcing with QIBM_TN_COMMIT_DURABLE environment variable

© 2007 IBM Corporation

Performance Tooling

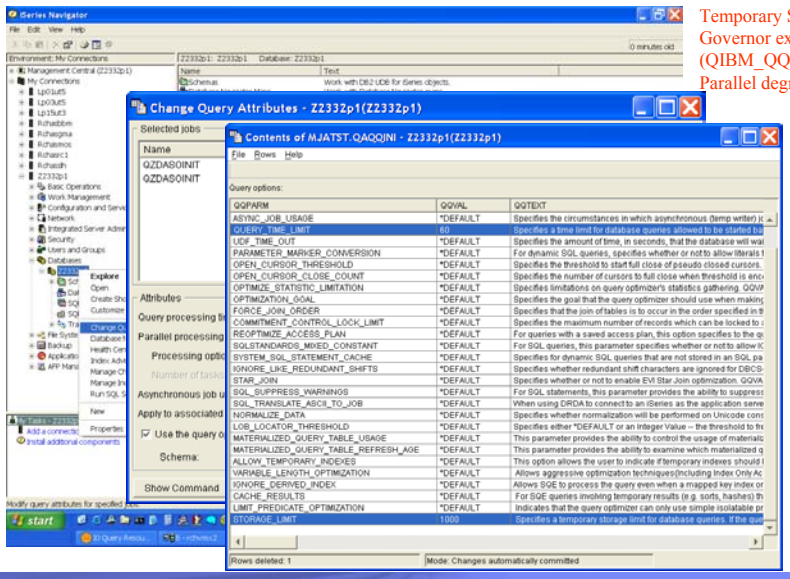
- **Predictive Temporary Storage Governor**
  - STORAGE_LIMIT QAQQINI option
  - QRYSTGLMT parameter on CHGQRYA command
- **Governor exit point for easier administration, QIBM_QQQ_QUERY_GOVR**
- **CURRENT DEGREE special register for controlling DB2 SMP parallelism via SQL**
- **VOLATILE Table Support**

  CREATE TABLE worktable (id INT, name CHAR(10), current_total DEC(8,2)) VOLATILE
- **New filters on STRDBMON command**
  - JOB wildcarding (QZDAS*)
  - RUNTHLD, FTRFILE, FTRUSER, FTRINTNETA
- **New Redbook on Database Monitor:**

  Diagnosing SQL Performance on DB2 UDB for iSeries (SG24-6654)

© 2007 IBM Corporation

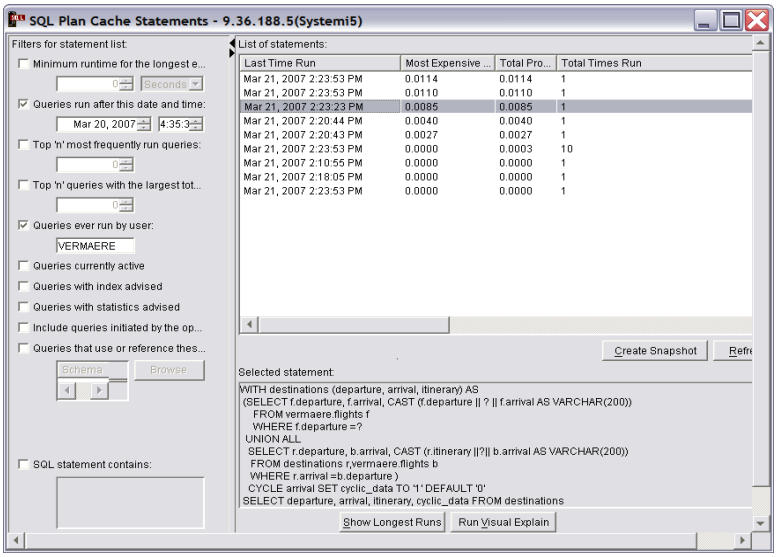
Governor Exit Point



Temporary Storage Governor
Governor exit point
(QIBM_QQQ_QUERY_GOVR)
Parallel degree granularity

© 2007 IBM Corporation

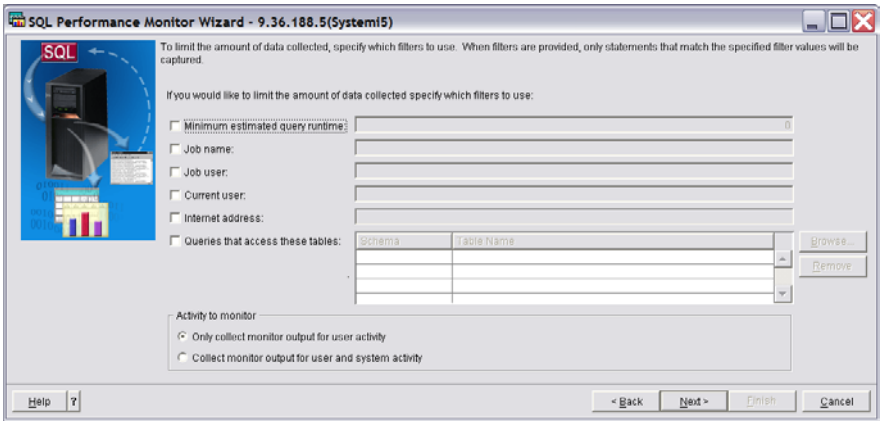
Live DB2 Performance Analysis via SQL Plan Cache



Always On - no database monitor overhead

© 2007 IBM Corporation

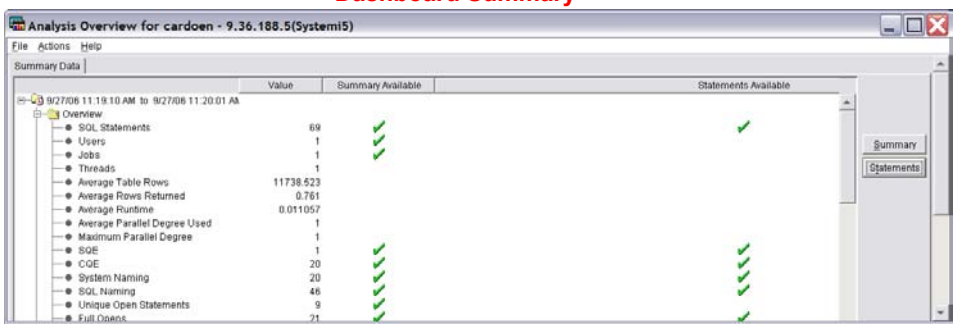
Advanced Database Monitor Filtering



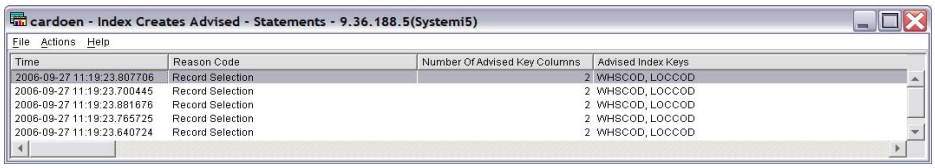
© 2007 IBM Corporation

Database Monitor Analysis Simplified

Dashboard Summary

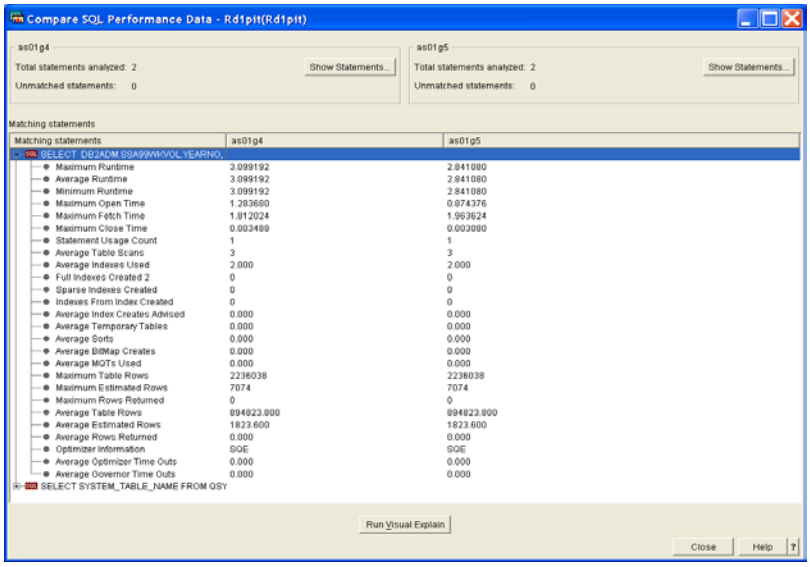


Drill-Thru Analysis



© 2007 IBM Corporation

Database Monitor Comparison



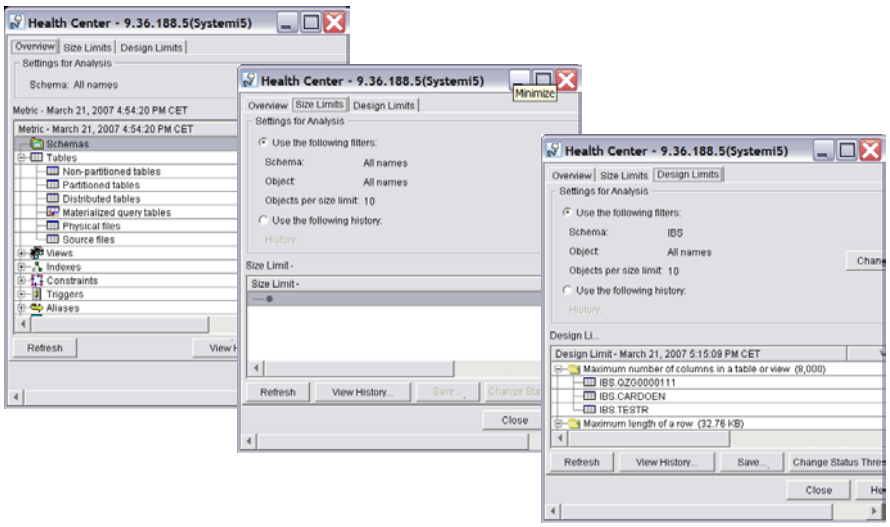
© 2007 IBM Corporation

Additional Capabilities

- **DB2 Health Center**
  - Monitor database metrics
  - Check which objects are nearing DB2 limits
- **Index Evaluator (first available on V5R3)**
  - "Native" last-used date added in V5R4
- **MQT Evaluator**
- **OnDemand SQL Analysis – enhanced "Show Current SQL"**
  - Statement name
  - Program or package name
  - Open Information

© 2007 IBM Corporation

DB2 Health Center



© 2007 IBM Corporation

Materialized Query Tables (MQT)

- **MQT Example:**

```
CREATE TABLE Example_MQT AS
       (SELECT Geography, Region, Year, Month,
         SUM(Revenue) AS Total_Revenue, SUM(Quantity) AS Total_Quantity,
         COUNT(*) AS Rows_per_Group
           FROM Example_Table
            GROUP BY Geography,Region,Year,Month)
      DATA INITIALLY IMMEDIATE
      REFRESH DEFERRED
      ENABLE QUERY OPTIMIZATION
      MAINTAINED BY USER
```

- **Optimizer could use the MQT instead of fully executing the following query:**

```
SELECT Geography, Year,
     SUM(Revenue) AS Total_Revenue, SUM(Quantity) AS Total_Quantity,
       FROM Example_Table WHERE Year IN (2004, 2005)
           GROUP BY Geography, Year;
```

white paper: **ibm.com/servers/enable/site/education/abstracts/438a_abs.html**

- **User responsible for keeping MQT data current and activating optimizer MQT awareness with QAQQINI options**

© 2007 IBM Corporation

Journaling Enhancements

- **Improved automatic journal configuration**
  - Support for SEQUENCE object
  - Automatic SQL journaling with QDFTJRN data area after CrtDupObj & Restore commands

    Data Area layout…
    Bytes 1-10: Journal Library
    Bytes 11-20: Journal Name
    Bytes 21-30: Object Type (*ALL,*FILE,*NONE, *DTAARA, *DTAQ)
    Bytes 31-40: Option (*ALLOPR,*CREATE, *MOVE, *RESTORE, *RSTOVRJRN)
- **SMAPP and journaling support for EVIs & ICU indexes**
- **Ability to Journal Minimal Data journal entries with *FLDBDY option**
- **Enhanced remote journal error reporting**
- **Customizable Journal Recovery Count**
- **Enhanced CHGJRN & WRKJRN commands**
- **Single Journal support for 10,000,000 objects**
- **New F-IT entry for identity columns**

© 2007 IBM Corporation

DB2 Cross References and Catalogs Recovery

- **More Robust DB2 Catalogs & Cross-Reference Files**
  - Queue protection
  - RCLSTG SELECT(*DBXREF) progress indicator
  - Library-level reclaim with RCLDBXREF command
  - Automatic rebuild QSYS2 & SYSIBM catalog views when cross-reference files recreated

© 2007 IBM Corporation

Additional Information

- **DB2 for i5/OS home page - ibm.com/iseries/db2**
- **Newsgroups**
  - USENET: comp.sys.ibm.as400.misc, comp.databases.ibm-db2
  - System i Network DB2 Forum - http://systeminetwork.com/isnetforums/forumdisplay.php
- **Education Resources - Classroom & Online**
  - ibm.com/iseries/db2/gettingstarted.html
  - ibm.com/servers/enable/site/education/ibo/view.html?oc#db2
- **DB2 for i5/OS Publications**
  - White Papers: ibm.com/servers/enable/site/education/ibo/view.html?wp#db2
  - Online Manuals: ibm.com/iseries/db2/books.html
  - Porting Help: ibm.com/servers/enable/site/db2/porting.html
  - DB2 for i5/OS Redbooks (http://ibm.com/redbooks)
    - Stored Procedures, Triggers, and UDFs on DB2 UDB for iSeries (SG24-6503-02)
    - Preparing for and Tuning the SQL Query Engine on DB2 for i5/OS (SG24-6598-01)
    - Modernizing iSeries Application Data Access (SG24-6393)
    - SQL Performance Diagnosis on DB2 for i5/OS (SG24-6654)

© 2007 IBM Corporation