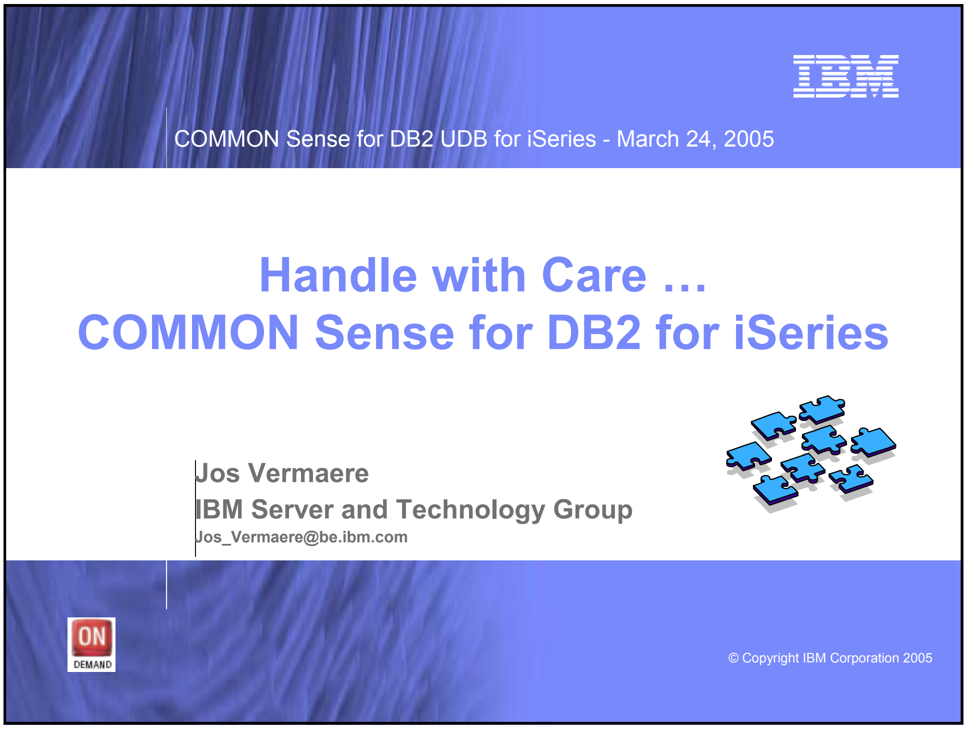
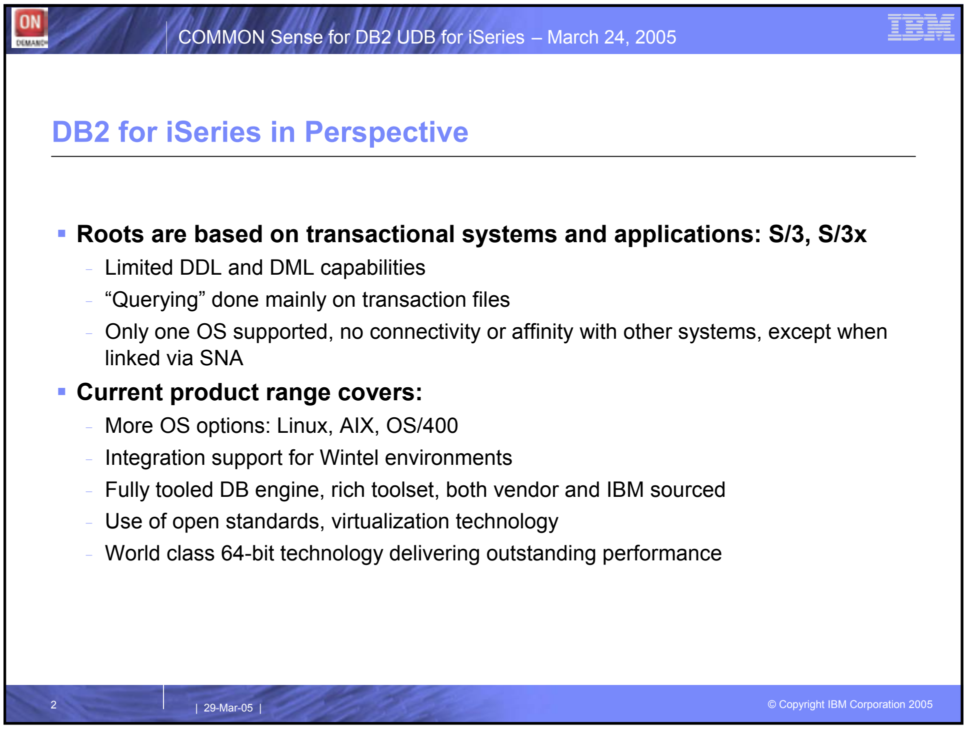


COMMON Sense for DB2 UDB for iSeries - March 24, 2005

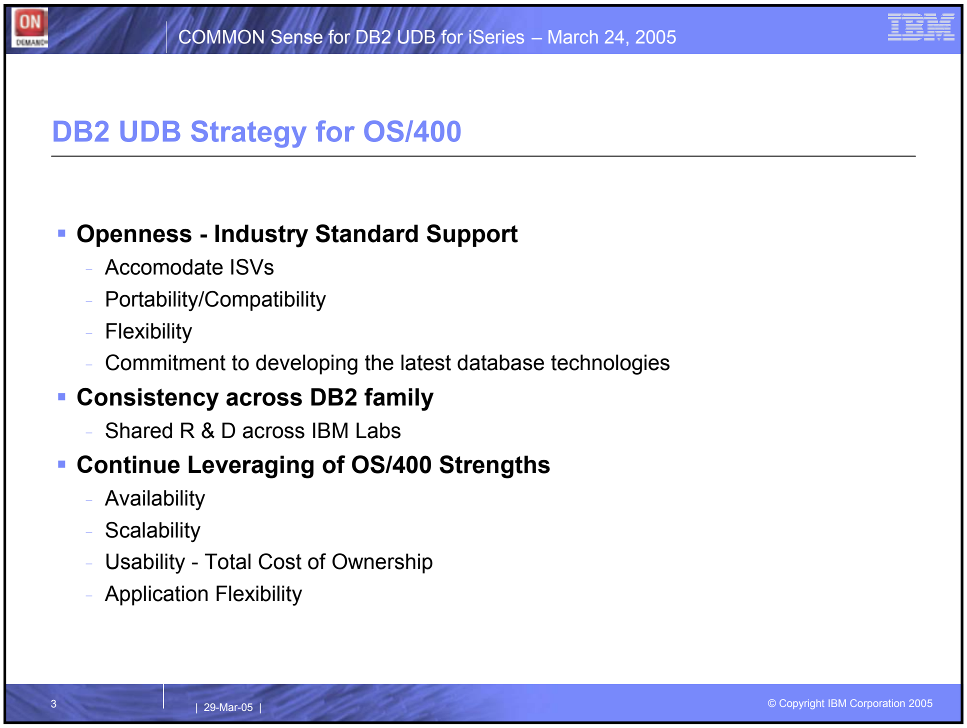
Handle with Care …
COMMON Sense for DB2 for iSeries

Jos Vermaere
IBM Server and Technology Group
Jos_Vermaere@be.ibm.com



DB2 for iSeries in Perspective

- **Roots are based on transactional systems and applications: S/3, S/3x**
  - Limited DDL and DML capabilities
  - “Querying” done mainly on transaction files
  - Only one OS supported, no connectivity or affinity with other systems, except when linked via SNA
- **Current product range covers:**
  - More OS options: Linux, AIX, OS/400
  - Integration support for Wintel environments
  - Fully tooled DB engine, rich toolset, both vendor and IBM sourced
  - Use of open standards, virtualization technology
  - World class 64-bit technology delivering outstanding performance

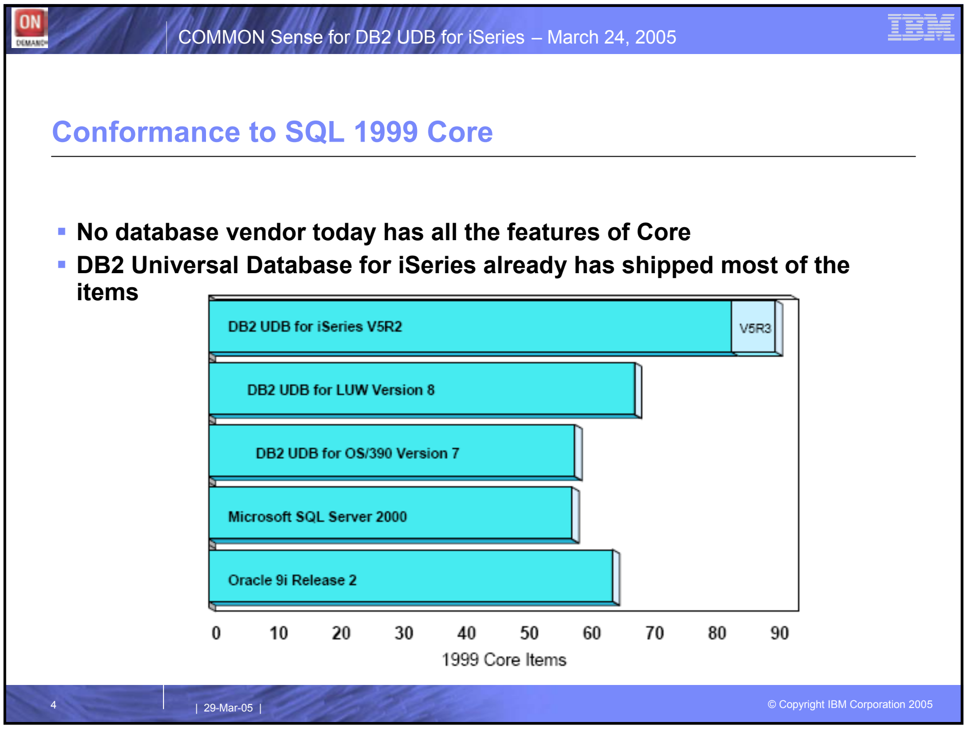


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

DB2 UDB Strategy for OS/400

- **Openness - Industry Standard Support**
  - Accomodate ISVs
  - Portability/Compatibility
  - Flexibility
  - Commitment to developing the latest database technologies
- **Consistency across DB2 family**
  - Shared R & D across IBM Labs
- **Continue Leveraging of OS/400 Strengths**
  - Availability
  - Scalability
  - Usability - Total Cost of Ownership
  - Application Flexibility

3 | 29-Mar-05 |

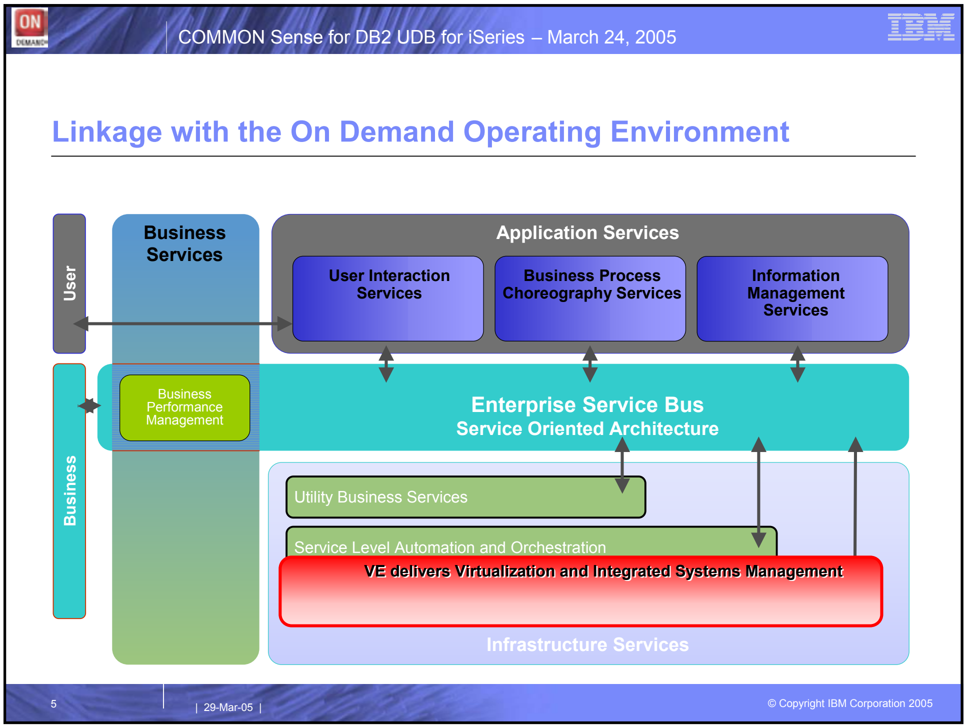


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Conformance to SQL 1999 Core

- **No database vendor today has all the features of Core**
- **DB2 Universal Database for iSeries already has shipped most of the items**

4 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Linkage with the On Demand Operating Environment

User

Business

Business Services

Business Performance Management

Application Services

User Interaction Services

Business Process Choreography Services

Information Management Services

Enterprise Service Bus

Service Oriented Architecture

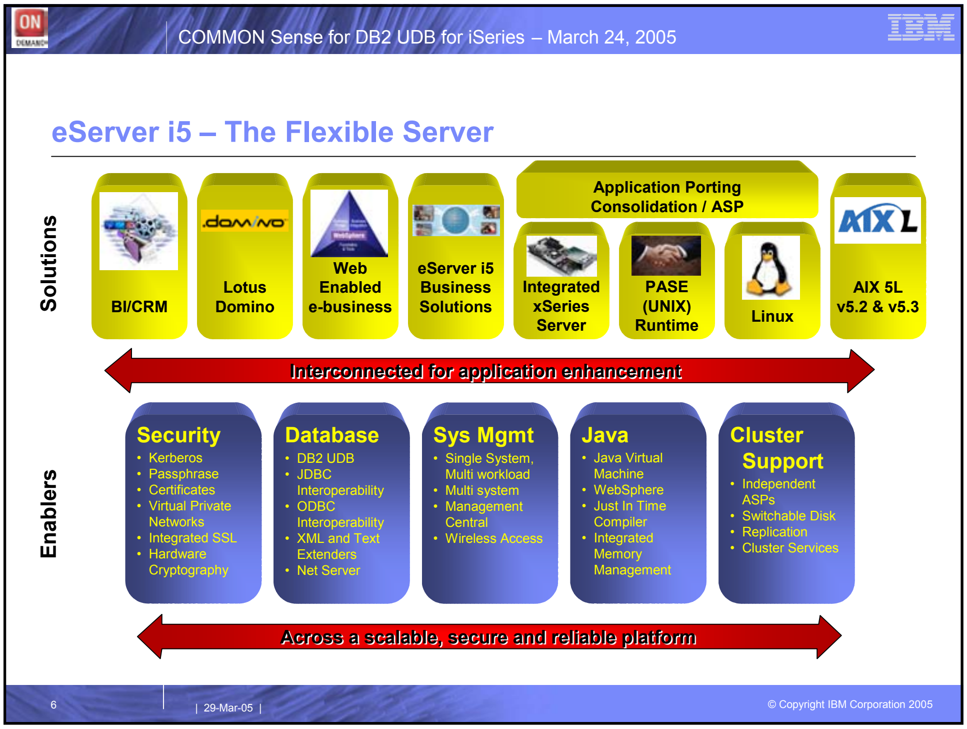
Utility Business Services

Service Level Automation and Orchestration

VE delivers Virtualization and Integrated Systems Management

Infrastructure Services

5 | 29-Mar-05 | © Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

eServer i5 – The Flexible Server

Solutions

BI/CRM

Lotus Domino

Web Enabled e-business

eServer i5 Business Solutions

Application Porting Consolidation / ASP

Integrated xSeries Server

PASE (UNIX) Runtime

Linux

AIX 5L v5.2 & v5.3

Interconnected for application enhancement

Enablers

Security
- Kerberos
- Passphrase
- Certificates
- Virtual Private Networks
- Integrated SSL
- Hardware Cryptography

Database
- DB2 UDB
- JDBC Interoperability
- ODBC Interoperability
- XML and Text Extenders
- Net Server

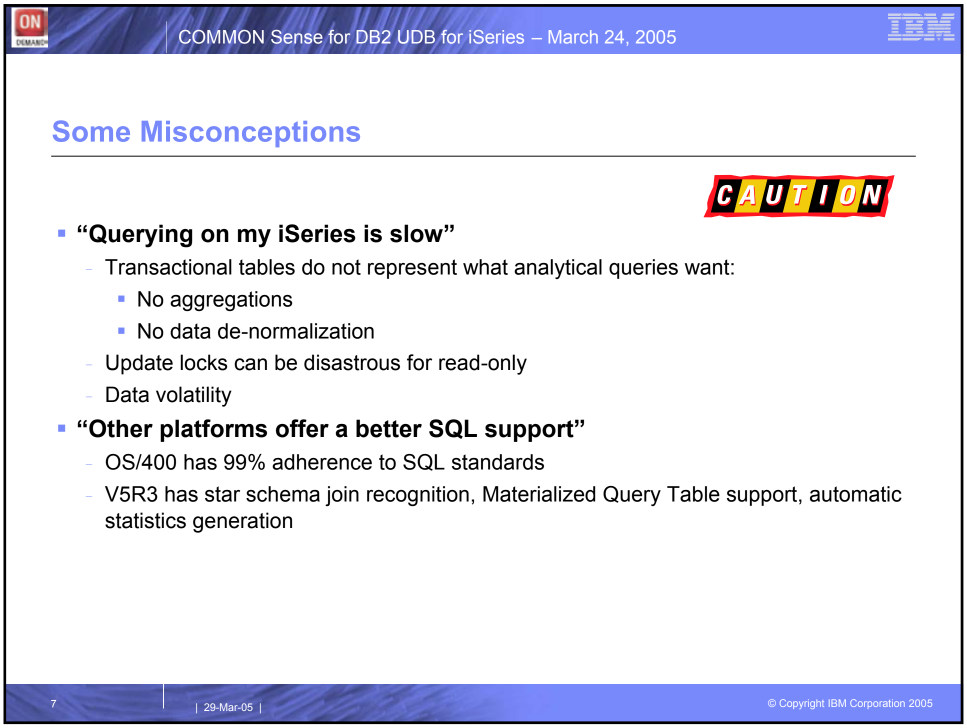
Sys Mgmt
- Single System, Multi workload
- Multi system Management Central
- Wireless Access

Java
- Java Virtual Machine
- WebSphere
- Just In Time Compiler
- Integrated Memory Management

Cluster Support
- Independent ASPs
- Switchable Disk
- Replication
- Cluster Services

Across a scalable, secure and reliable platform

6 | 29-Mar-05 | © Copyright IBM Corporation 2005



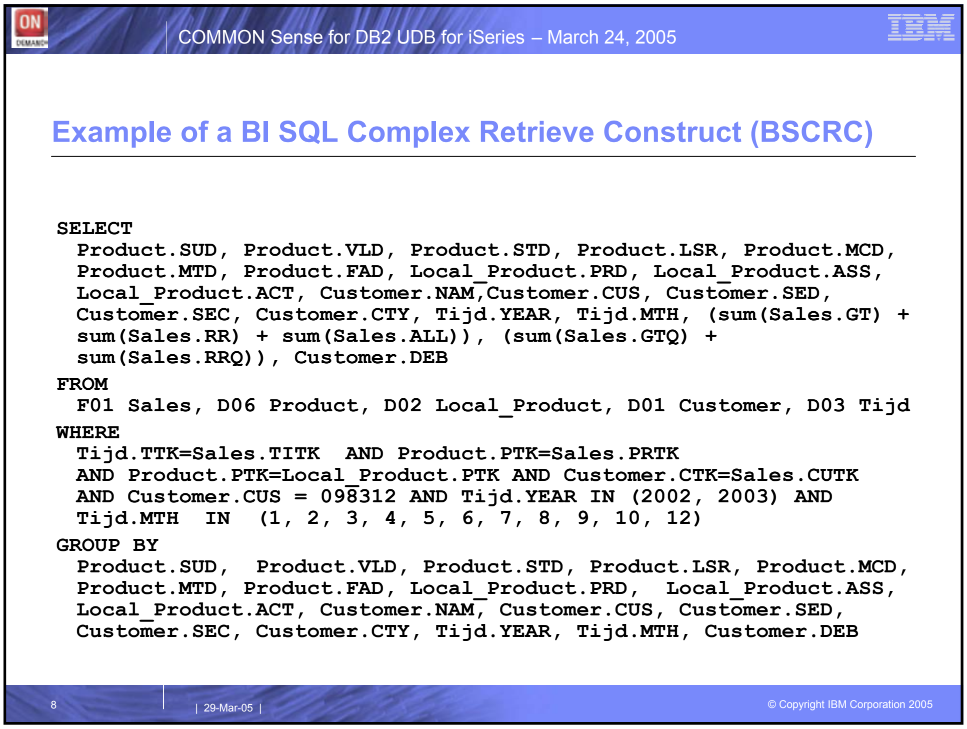
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Some Misconceptions

CAUTION

- **“Querying on my iSeries is slow”**
  - Transactional tables do not represent what analytical queries want:
    - No aggregations
    - No data de-normalization
  - Update locks can be disastrous for read-only
  - Data volatility
- **“Other platforms offer a better SQL support”**
  - OS/400 has 99% adherence to SQL standards
  - V5R3 has star schema join recognition, Materialized Query Table support, automatic statistics generation

7 | 29-Mar-05 |

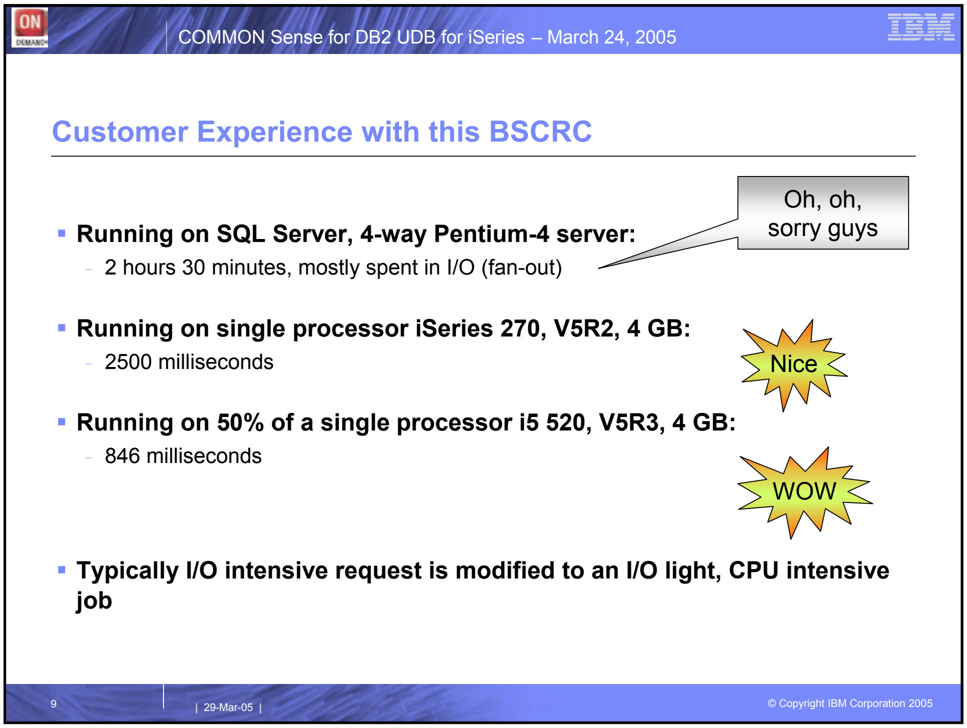


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Example of a BI SQL Complex Retrieve Construct (BSCRC)

```
SELECT
  Product.SUD, Product.VLD, Product.STD, Product.LSR, Product.MCD,
  Product.MTD, Product.FAD, Local_Product.PRD, Local_Product.ASS,
  Local_Product.ACT, Customer.NAM,Customer.CUS, Customer.SED,
  Customer.SEC, Customer.CTY, Tijd.YEAR, Tijd.MTH, (sum(Sales.GT) +
  sum(Sales.RR) + sum(Sales.ALL)), (sum(Sales.GTQ) +
  sum(Sales.RRQ)), Customer.DEB
FROM
  F01 Sales, D06 Product, D02 Local_Product, D01 Customer, D03 Tijd
WHERE
  Tijd.TTK=Sales.TITK  AND Product.PTK=Sales.PRTK
  AND Product.PTK=Local_Product.PTK AND Customer.CTK=Sales.CUTK
  AND Customer.CUS = 098312 AND Tijd.YEAR IN (2002, 2003) AND
  Tijd.MTH  IN  (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12)
GROUP BY
  Product.SUD,  Product.VLD, Product.STD, Product.LSR, Product.MCD,
  Product.MTD, Product.FAD, Local_Product.PRD,  Local_Product.ASS,
  Local_Product.ACT, Customer.NAM, Customer.CUS, Customer.SED,
  Customer.SEC, Customer.CTY, Tijd.YEAR, Tijd.MTH, Customer.DEB
```

8 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Customer Experience with this BSCRC

- **Running on SQL Server, 4-way Pentium-4 server:**
  - 2 hours 30 minutes, mostly spent in I/O (fan-out)

Oh, oh, sorry guys

- **Running on single processor iSeries 270, V5R2, 4 GB:**
  - 2500 milliseconds

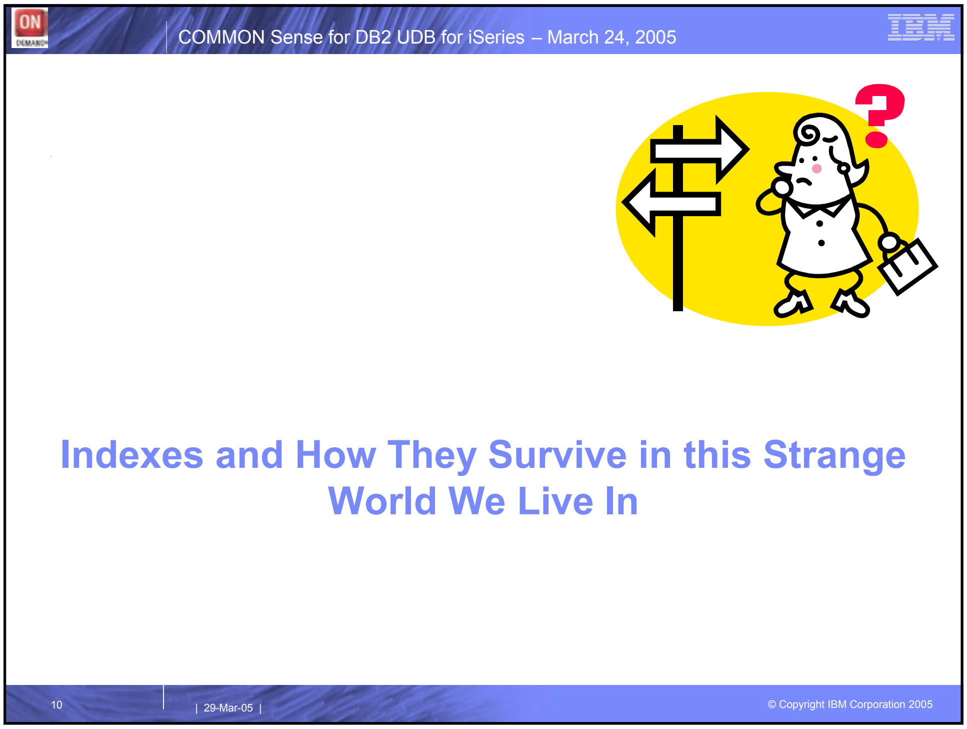
Nice

- **Running on 50% of a single processor i5 520, V5R3, 4 GB:**
  - 846 milliseconds

WOW

- **Typically I/O intensive request is modified to an I/O light, CPU intensive job**

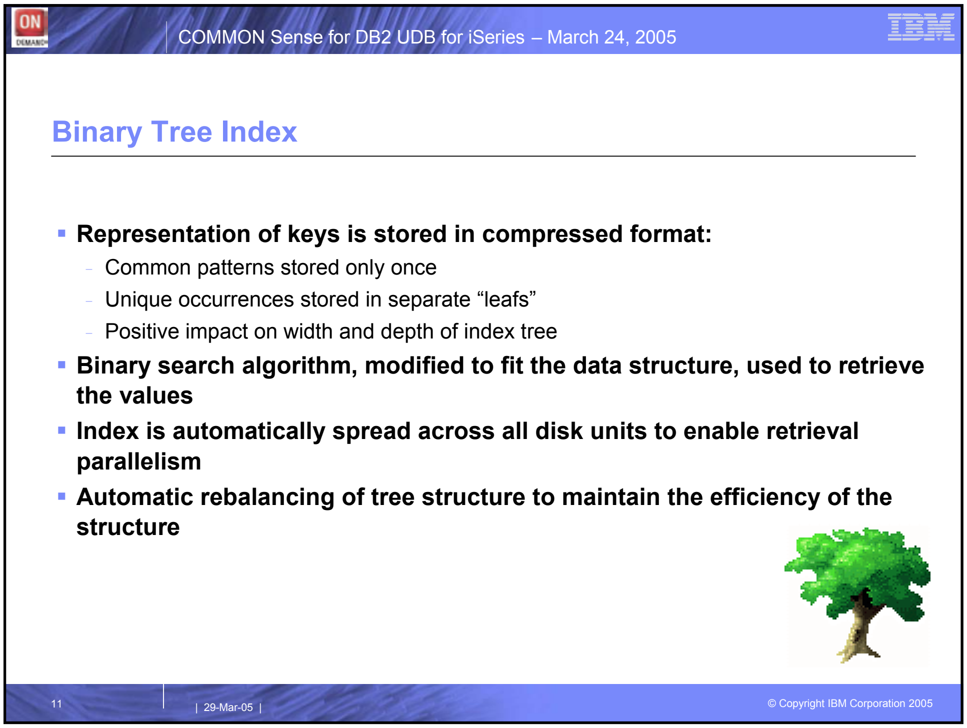
9 | 29-Mar-05 | © Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

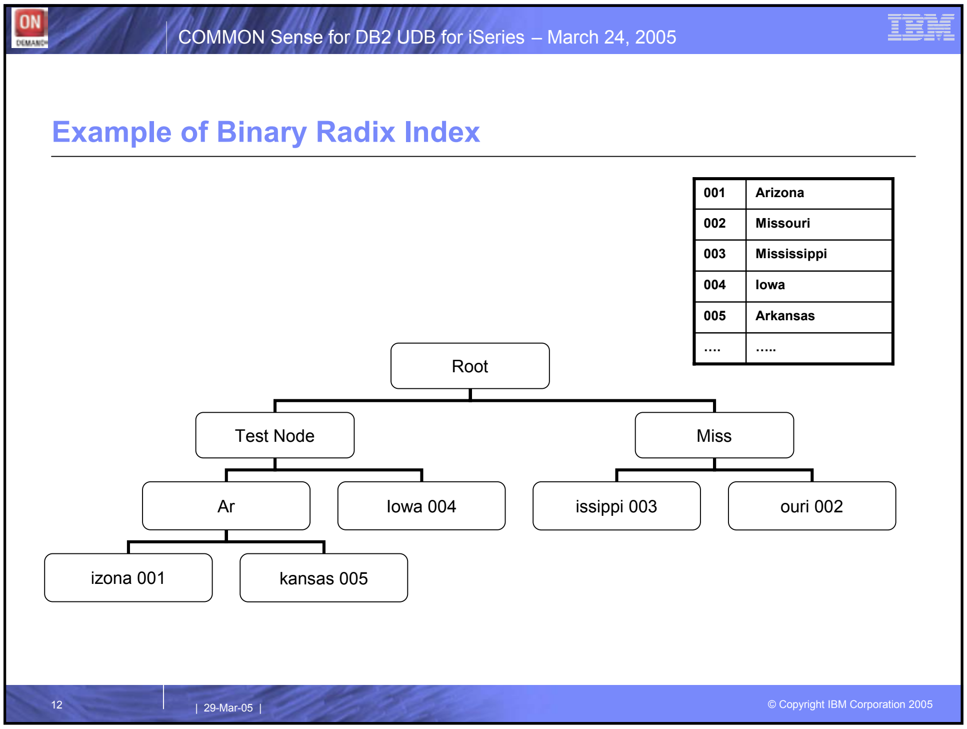
Indexes and How They Survive in this Strange World We Live In

10 | 29-Mar-05 | © Copyright IBM Corporation 2005



Binary Tree Index

- **Representation of keys is stored in compressed format:**
  - Common patterns stored only once
  - Unique occurrences stored in separate "leafs"
  - Positive impact on width and depth of index tree
- **Binary search algorithm, modified to fit the data structure, used to retrieve the values**
- **Index is automatically spread across all disk units to enable retrieval parallelism**
- **Automatic rebalancing of tree structure to maintain the efficiency of the structure**



Example of Binary Radix Index

001	**Arizona**
002	**Missouri**
003	**Mississippi**
004	**Iowa**
005	**Arkansas**
….	**…..**

Root

Test Node

Miss

Ar

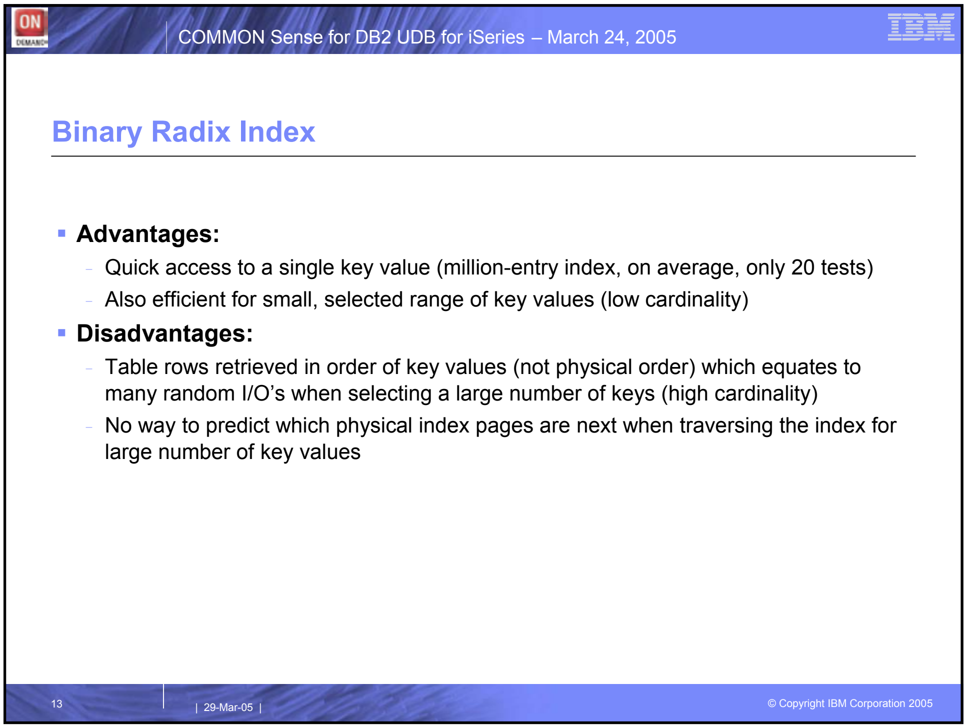
Iowa 004

issippi 003

ouri 002

izona 001

kansas 005

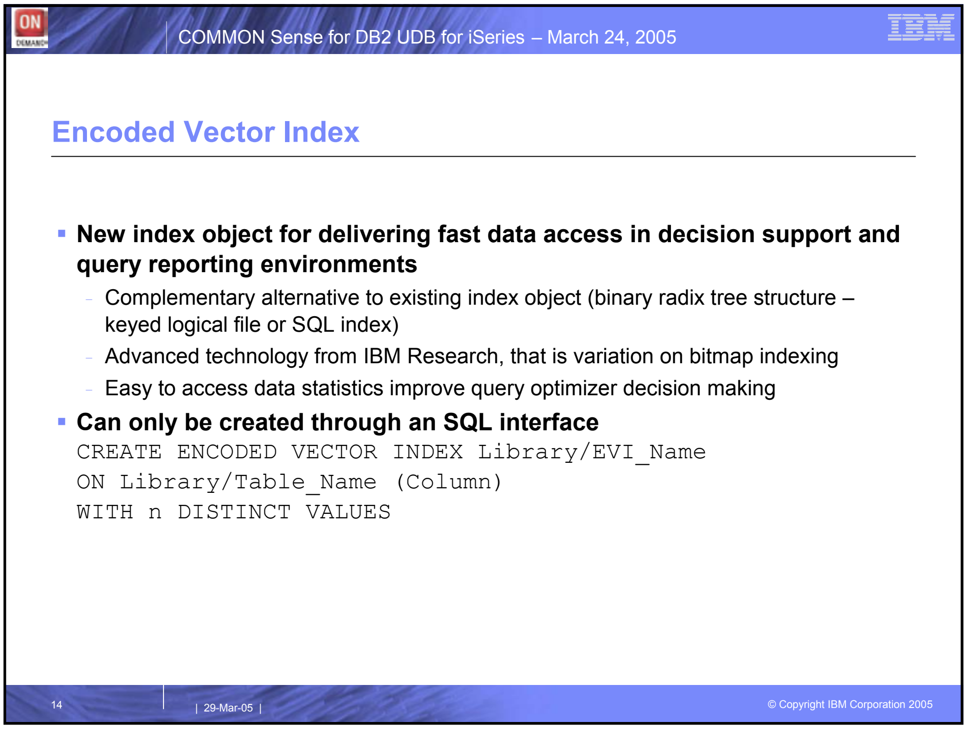


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Binary Radix Index

- **Advantages:**
  - Quick access to a single key value (million-entry index, on average, only 20 tests)
  - Also efficient for small, selected range of key values (low cardinality)
- **Disadvantages:**
  - Table rows retrieved in order of key values (not physical order) which equates to many random I/O's when selecting a large number of keys (high cardinality)
  - No way to predict which physical index pages are next when traversing the index for large number of key values

13 | 29-Mar-05 |



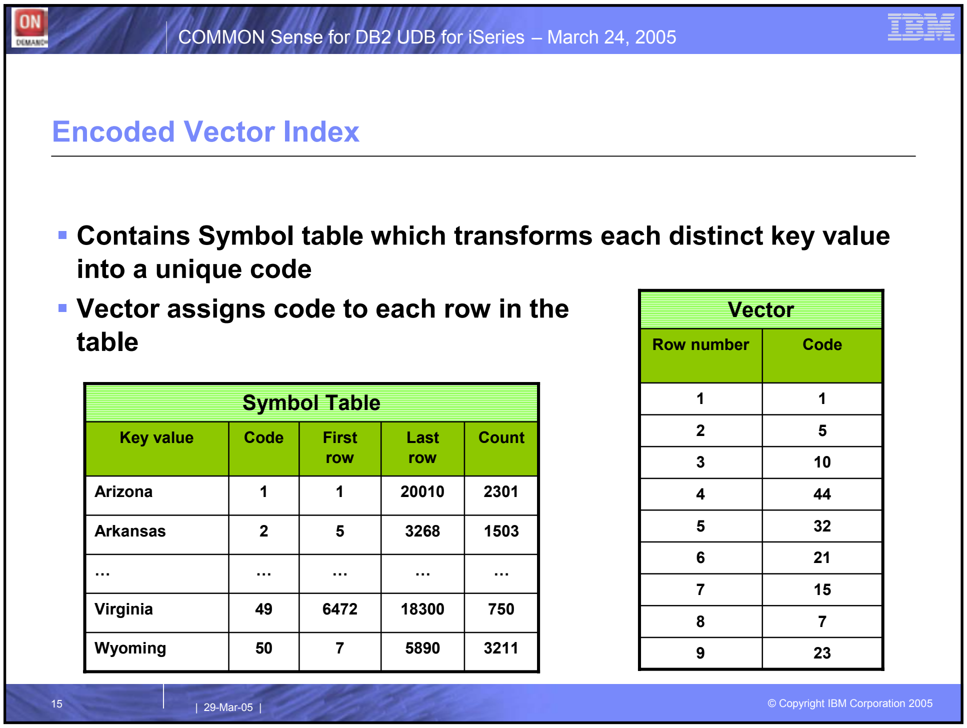
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Encoded Vector Index

- **New index object for delivering fast data access in decision support and query reporting environments**
  - Complementary alternative to existing index object (binary radix tree structure – keyed logical file or SQL index)
  - Advanced technology from IBM Research, that is variation on bitmap indexing
  - Easy to access data statistics improve query optimizer decision making
- **Can only be created through an SQL interface**

```
CREATE ENCODED VECTOR INDEX Library/EVI_Name
ON Library/Table_Name (Column)
WITH n DISTINCT VALUES
```

14 | 29-Mar-05 |

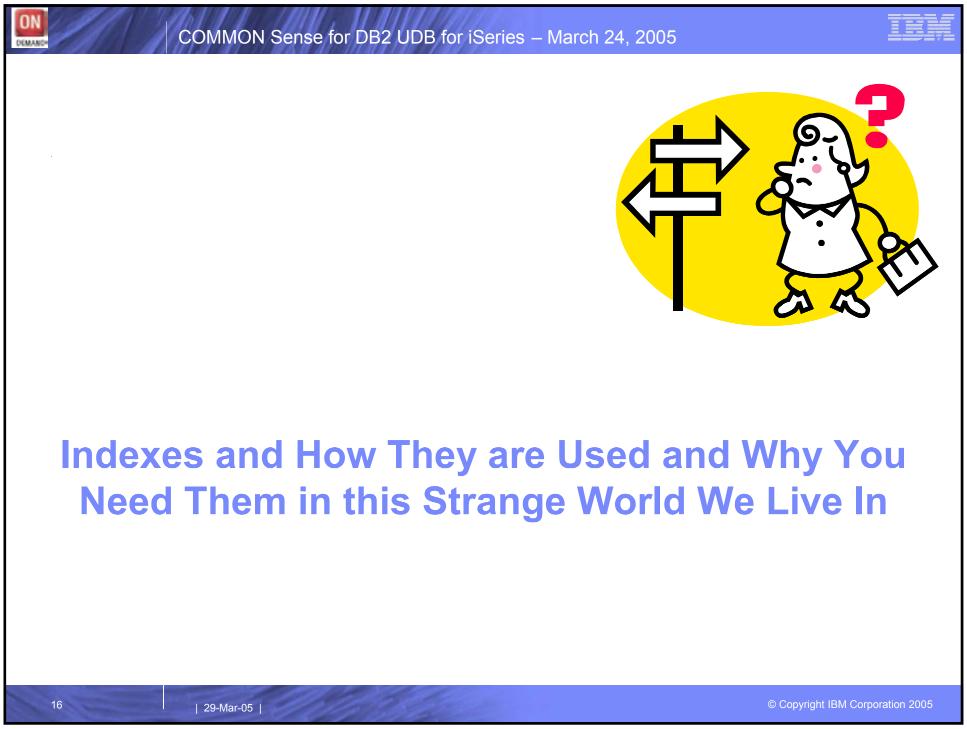


Encoded Vector Index

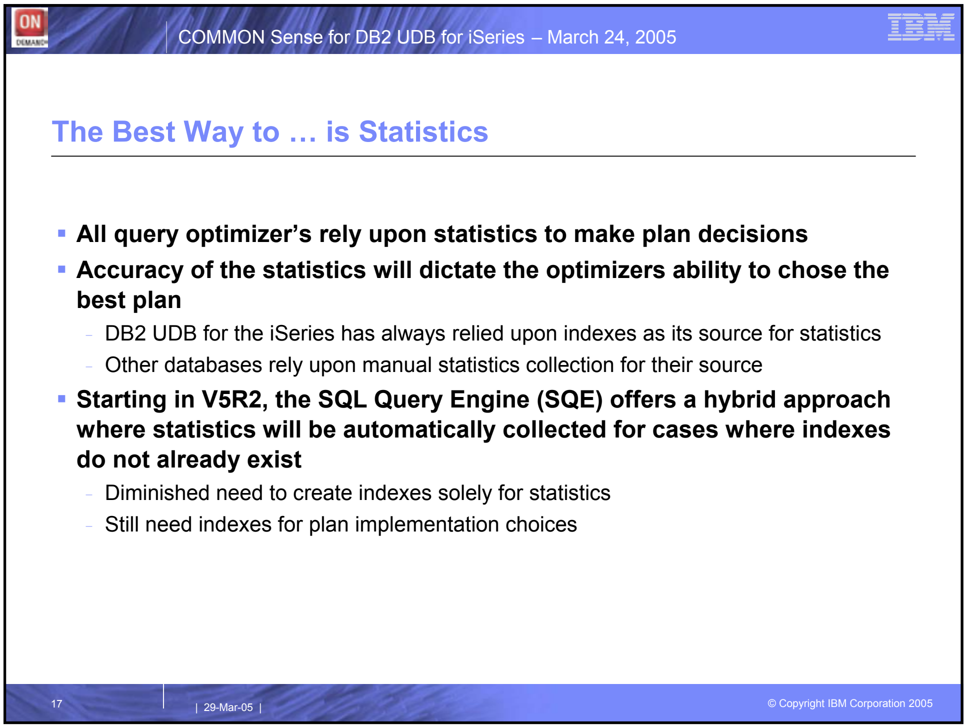
- Contains Symbol table which transforms each distinct key value into a unique code
- Vector assigns code to each row in the table

Symbol Table				
Key value	Code	First row	Last row	Count
Arizona	1	1	20010	2301
Arkansas	2	5	3268	1503
…	…	…	…	…
Virginia	49	6472	18300	750
Wyoming	50	7	5890	3211

Vector	
Row number	Code
1	1
2	5
3	10
4	44
5	32
6	21
7	15
8	7
9	23



Indexes and How They are Used and Why You Need Them in this Strange World We Live In

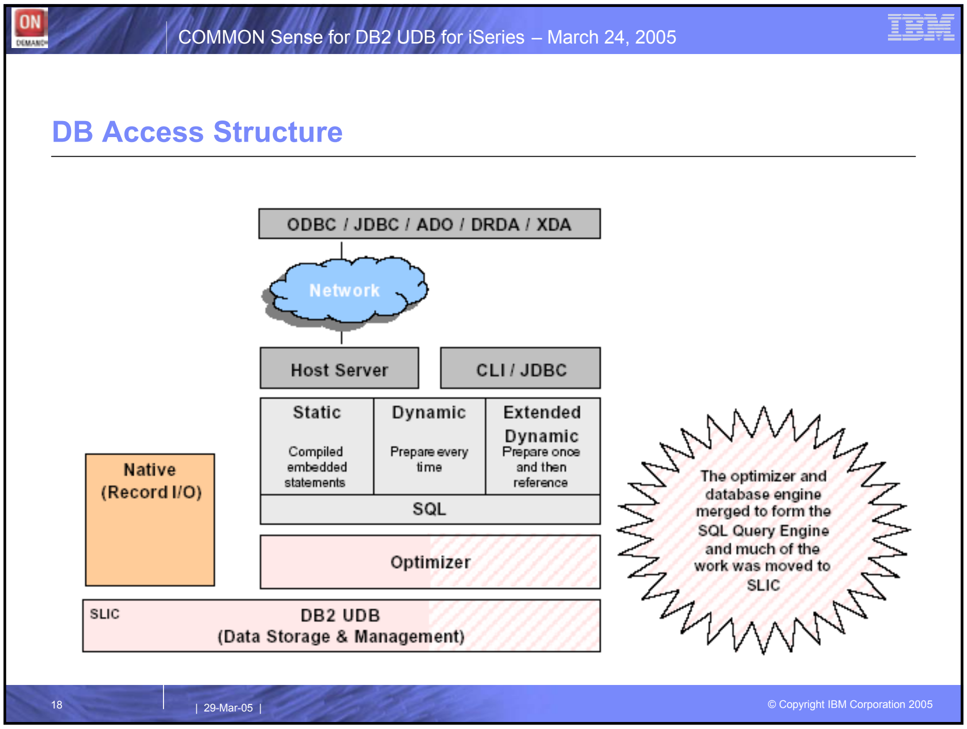


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

The Best Way to … is Statistics

- **All query optimizer's rely upon statistics to make plan decisions**
- **Accuracy of the statistics will dictate the optimizers ability to chose the best plan**
  - DB2 UDB for the iSeries has always relied upon indexes as its source for statistics
  - Other databases rely upon manual statistics collection for their source
- **Starting in V5R2, the SQL Query Engine (SQE) offers a hybrid approach where statistics will be automatically collected for cases where indexes do not already exist**
  - Diminished need to create indexes solely for statistics
  - Still need indexes for plan implementation choices

17 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

DB Access Structure

ODBC / JDBC / ADO / DRDA / XDA

Network

Host Server

CLI / JDBC

Static

Compiled embedded statements

Dynamic

Prepare every time

Extended Dynamic

Prepare once and then reference

SQL

Native (Record I/O)

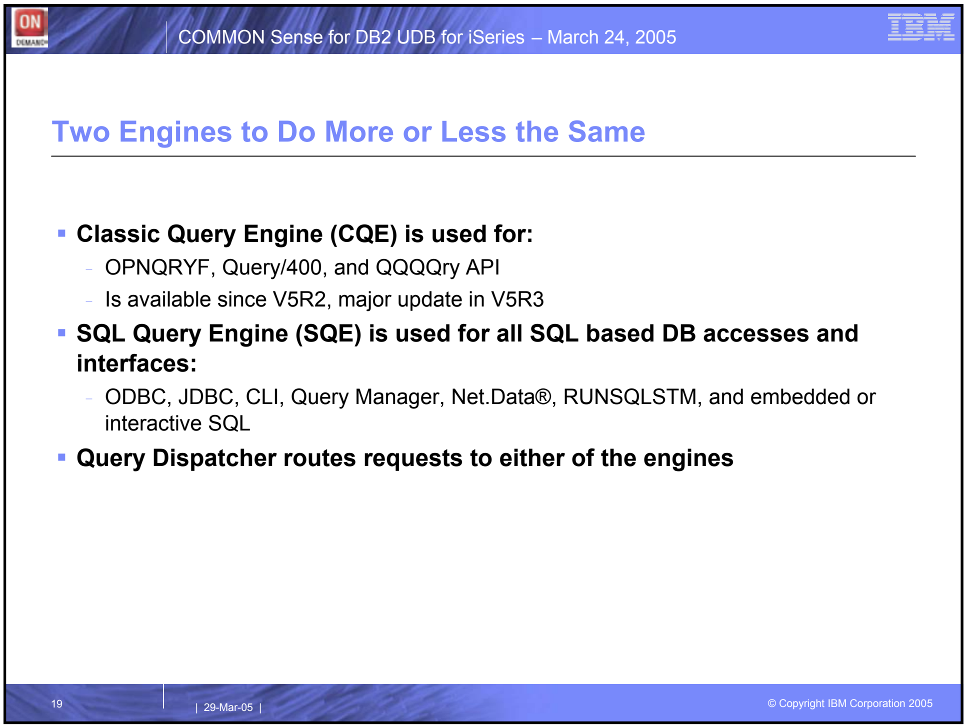
Optimizer

SLIC

DB2 UDB (Data Storage & Management)

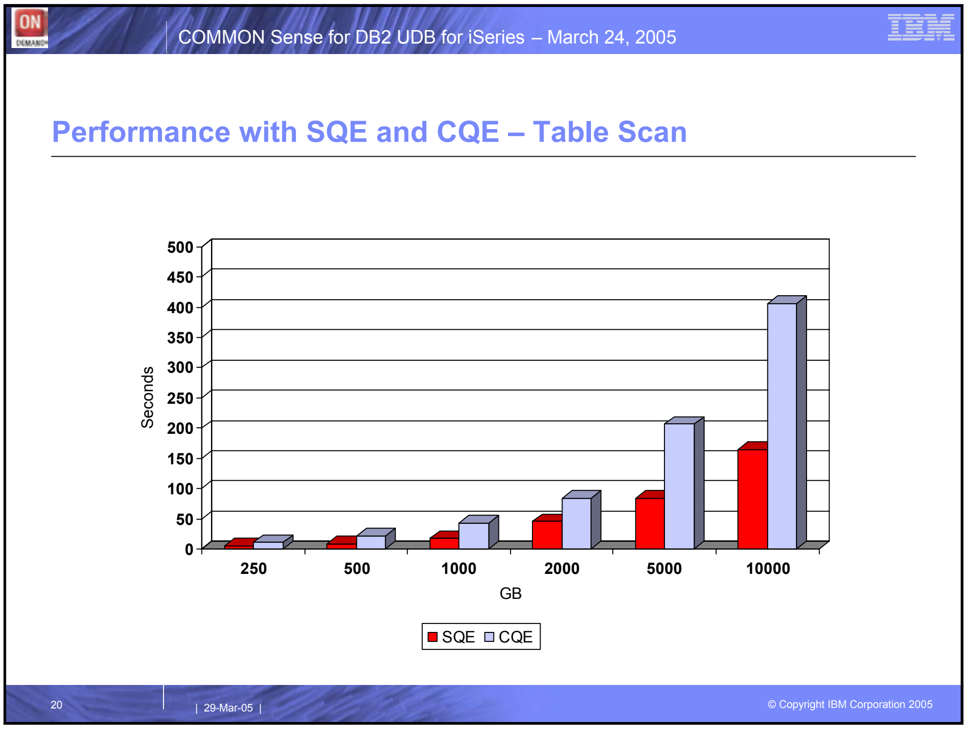
The optimizer and database engine merged to form the SQL Query Engine and much of the work was moved to SLIC

18 | 29-Mar-05 |

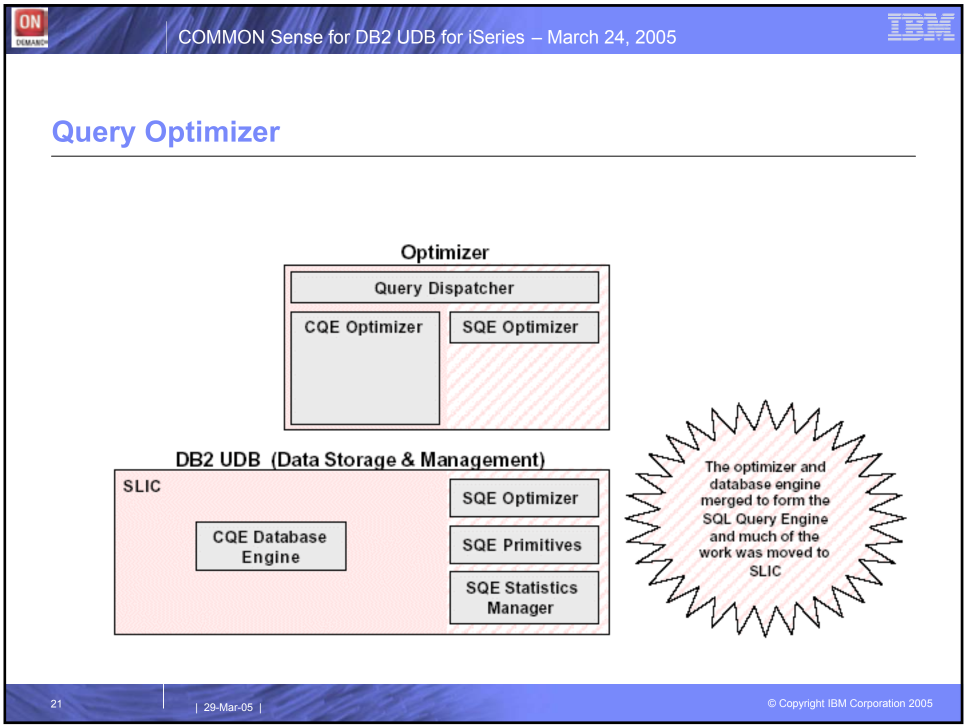


Two Engines to Do More or Less the Same

- **Classic Query Engine (CQE) is used for:**
  - OPNQRYF, Query/400, and QQQQry API
  - Is available since V5R2, major update in V5R3
- **SQL Query Engine (SQE) is used for all SQL based DB accesses and interfaces:**
  - ODBC, JDBC, CLI, Query Manager, Net.Data®, RUNSQLSTM, and embedded or interactive SQL
- **Query Dispatcher routes requests to either of the engines**



Performance with SQE and CQE – Table Scan



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Query Optimizer

Optimizer

Query Dispatcher

CQE Optimizer

SQE Optimizer

DB2 UDB (Data Storage & Management)

SLIC

CQE Database Engine

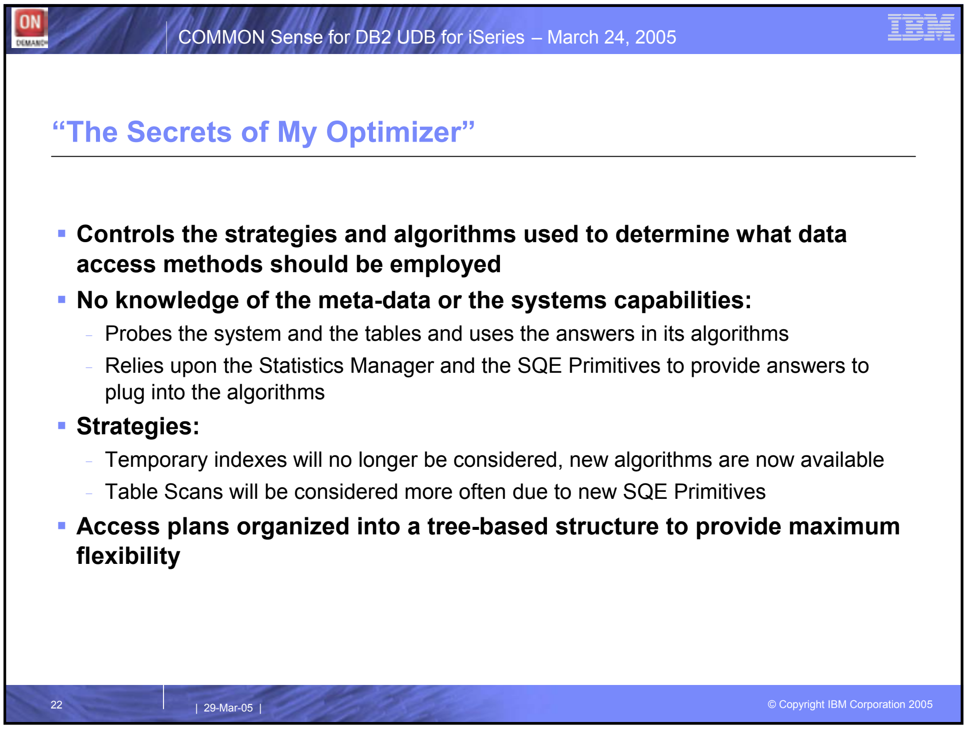
SQE Optimizer

SQE Primitives

SQE Statistics Manager

The optimizer and database engine merged to form the SQL Query Engine and much of the work was moved to SLIC

21 | 29-Mar-05 |

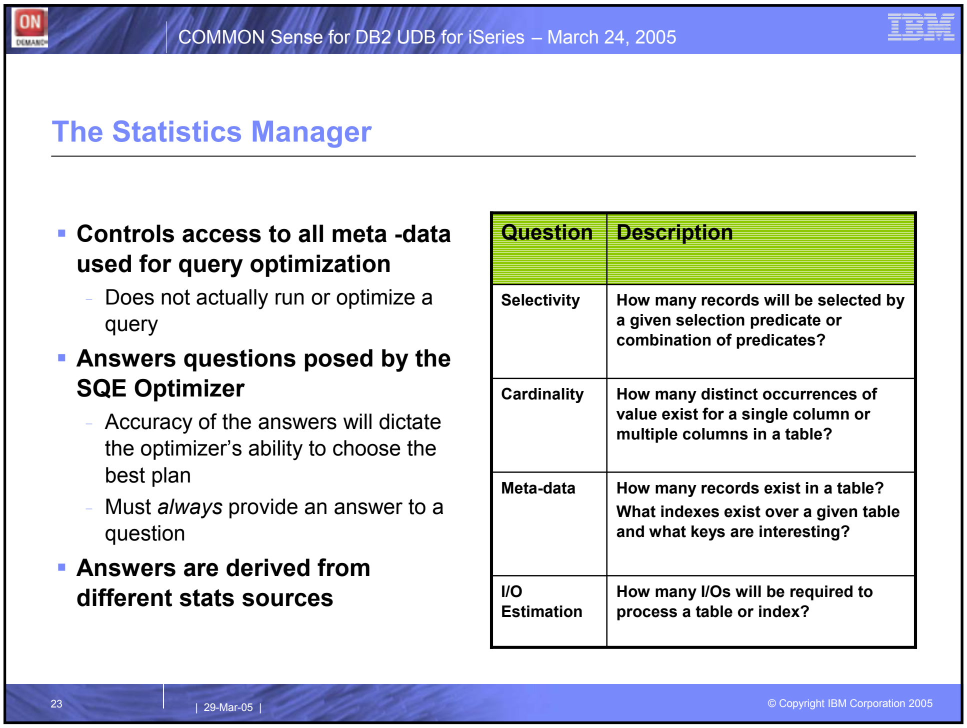


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

“The Secrets of My Optimizer”

- **Controls the strategies and algorithms used to determine what data access methods should be employed**
- **No knowledge of the meta-data or the systems capabilities:**
  - Probes the system and the tables and uses the answers in its algorithms
  - Relies upon the Statistics Manager and the SQE Primitives to provide answers to plug into the algorithms
- **Strategies:**
  - Temporary indexes will no longer be considered, new algorithms are now available
  - Table Scans will be considered more often due to new SQE Primitives
- **Access plans organized into a tree-based structure to provide maximum flexibility**

22 | 29-Mar-05 |



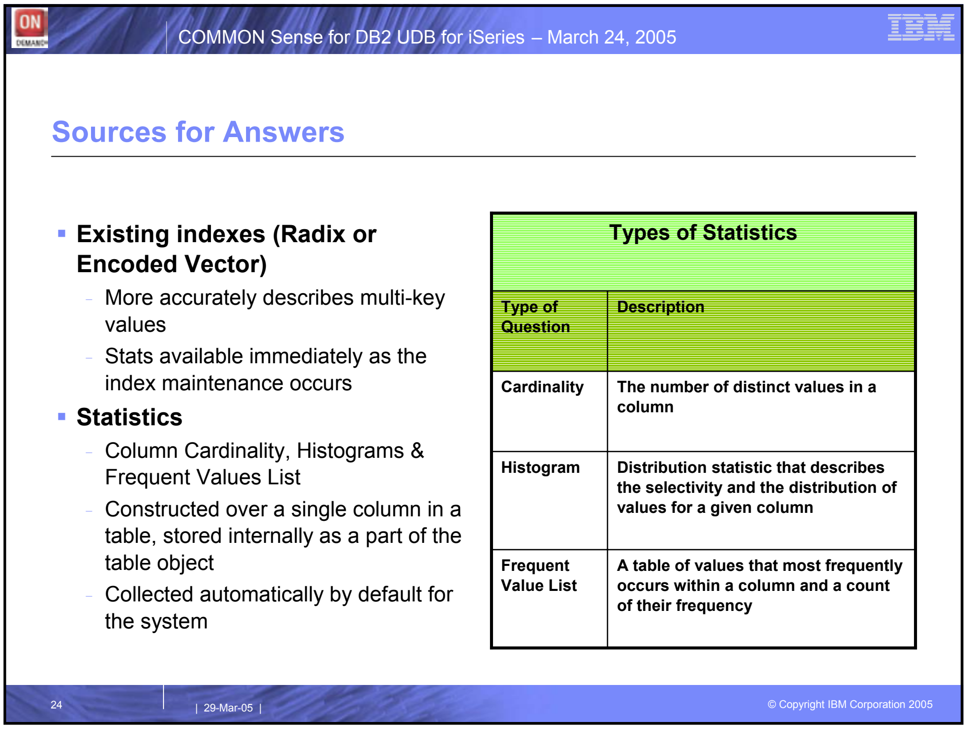
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

The 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**

Question	Description
Selectivity	How many records will be selected by a given selection predicate or combination of predicates?
Cardinality	How many distinct occurrences of value exist for a single column or multiple columns in a table?
Meta-data	How many records exist in a table? What indexes exist over a given table and what keys are interesting?
I/O Estimation	How many I/Os will be required to process a table or index?

23 | 29-Mar-05 | © Copyright IBM Corporation 2005



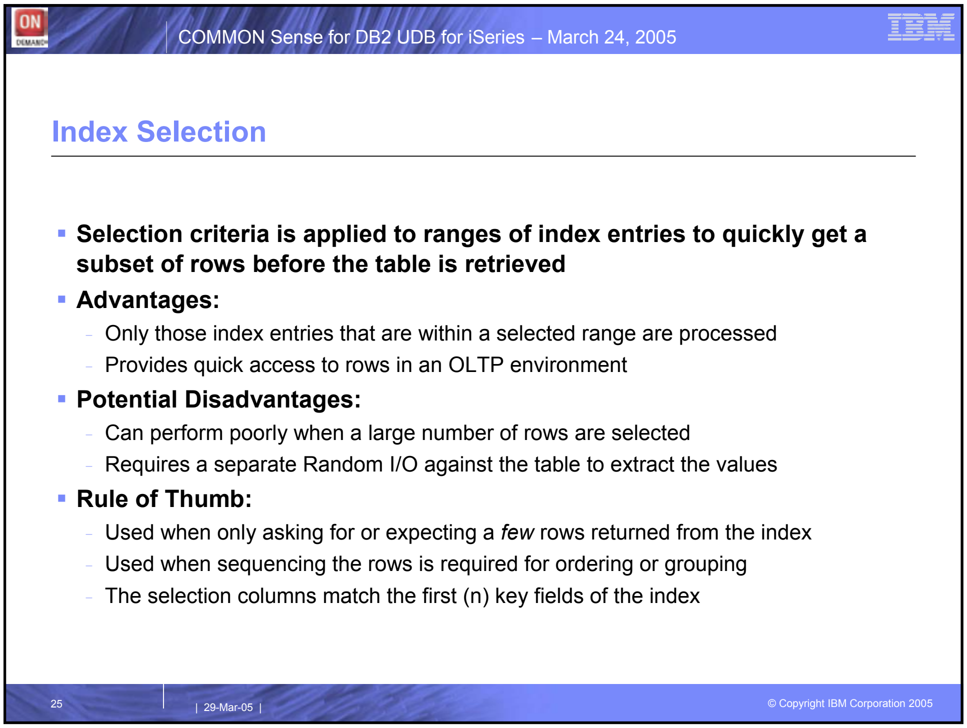
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Sources for Answers

- **Existing indexes (Radix or Encoded Vector)**
  - More accurately describes multi-key values
  - Stats available immediately as the index maintenance occurs
- **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

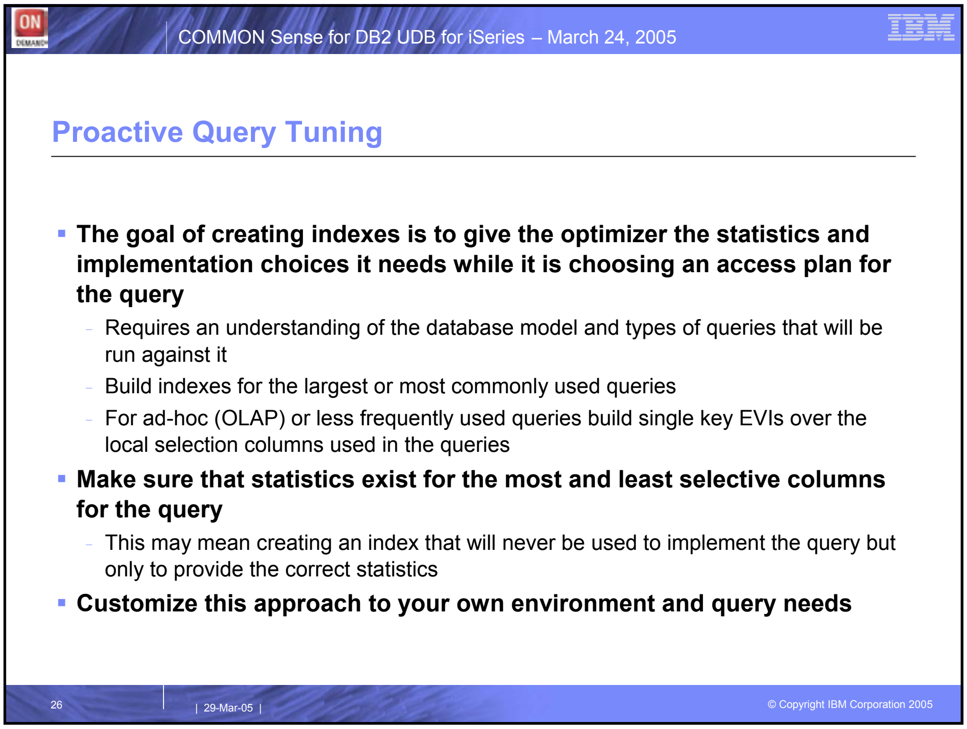
Types of Statistics	
Type of Question	Description
Cardinality	The number of distinct values in a column
Histogram	Distribution statistic that describes the selectivity and the distribution of values for a given column
Frequent Value List	A table of values that most frequently occurs within a column and a count of their frequency

24 | 29-Mar-05 | © Copyright IBM Corporation 2005



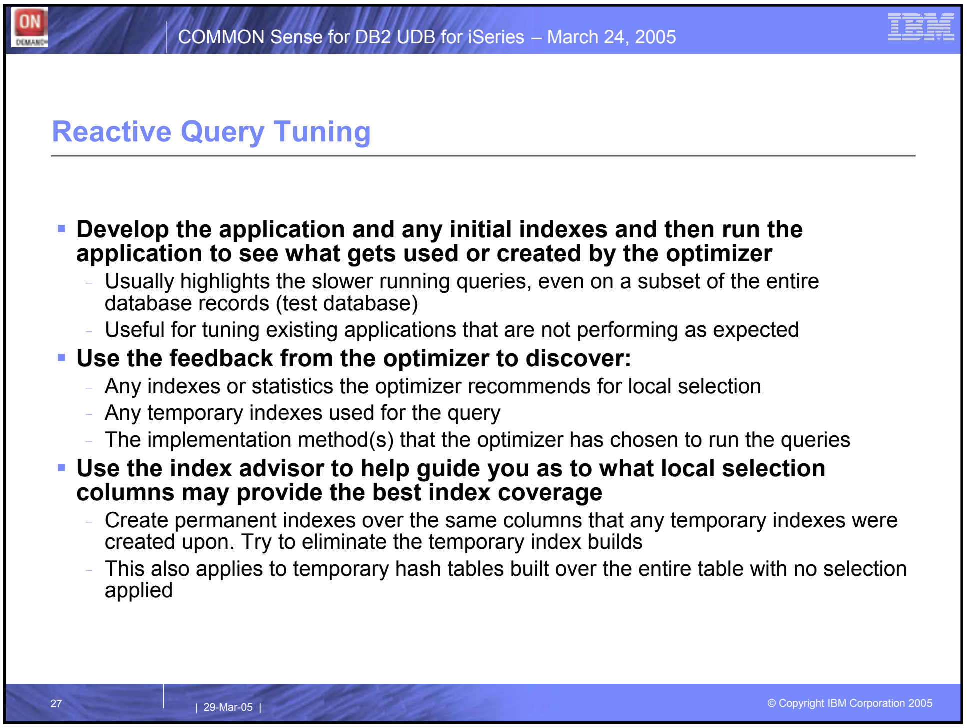
Index Selection

- **Selection criteria is applied to ranges of index entries to quickly get a subset of rows before the table is retrieved**
- **Advantages:**
  - Only those index entries that are within a selected range are processed
  - Provides quick access to rows in an OLTP environment
- **Potential Disadvantages:**
  - Can perform poorly when a large number of rows are selected
  - Requires a separate Random I/O against the table to extract the values
- **Rule of Thumb:**
  - Used when only asking for or expecting a *few* rows returned from the index
  - Used when sequencing the rows is required for ordering or grouping
  - The selection columns match the first (n) key fields of the index



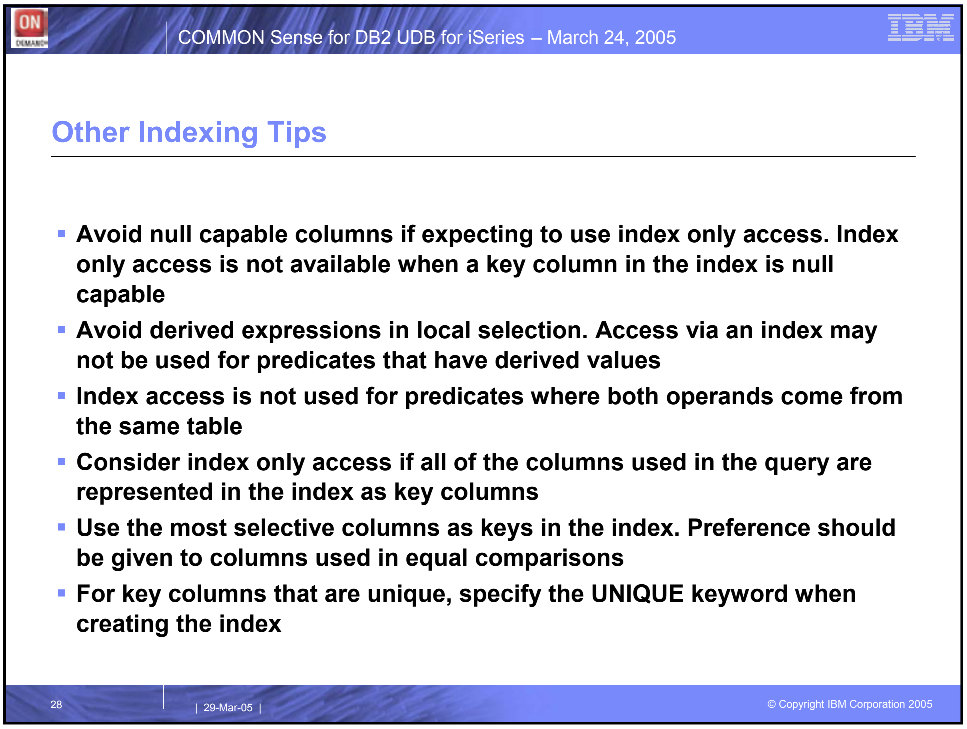
Proactive Query Tuning

- **The goal of creating indexes is to give the optimizer the statistics and implementation choices it needs while it is choosing an access plan for the query**
  - Requires an understanding of the database model and types of queries that will be run against it
  - Build indexes for the largest or most commonly used queries
  - For ad-hoc (OLAP) or less frequently used queries build single key EVIs over the local selection columns used in the queries
- **Make sure that statistics exist for the most and least selective columns for the query**
  - This may mean creating an index that will never be used to implement the query but only to provide the correct statistics
- **Customize this approach to your own environment and query needs**



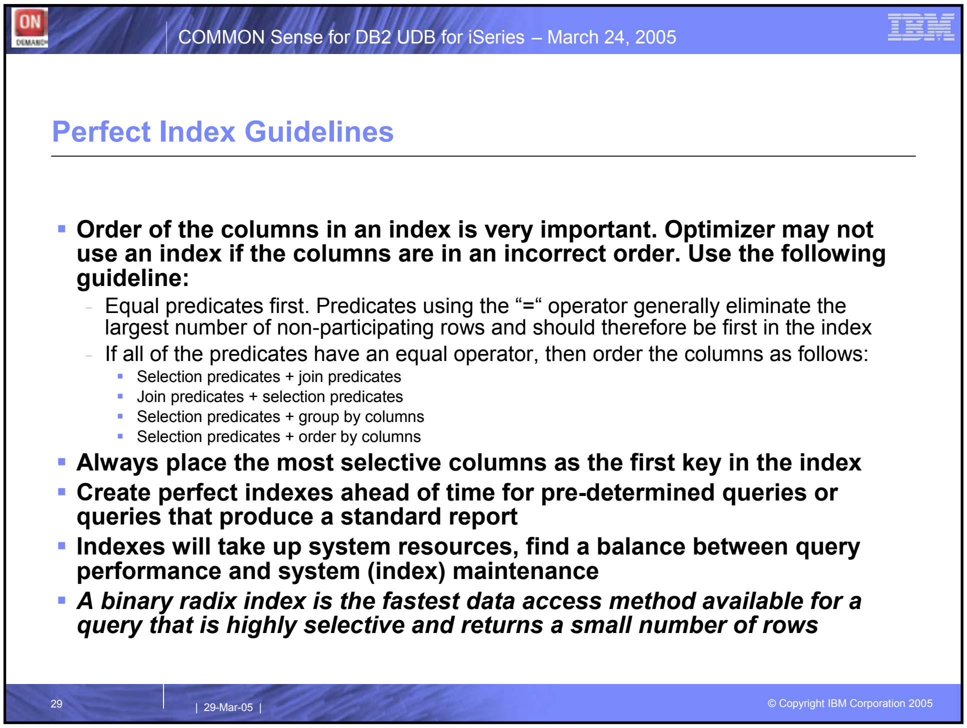
Reactive Query Tuning

- **Develop the application and any initial indexes and then run the application to see what gets used or created by the optimizer**
  - Usually highlights the slower running queries, even on a subset of the entire database records (test database)
  - Useful for tuning existing applications that are not performing as expected
- **Use the feedback from the optimizer to discover:**
  - Any indexes or statistics the optimizer recommends for local selection
  - Any temporary indexes used for the query
  - The implementation method(s) that the optimizer has chosen to run the queries
- **Use the index advisor to help guide you as to what local selection columns may provide the best index coverage**
  - Create permanent indexes over the same columns that any temporary indexes were created upon. Try to eliminate the temporary index builds
  - This also applies to temporary hash tables built over the entire table with no selection applied



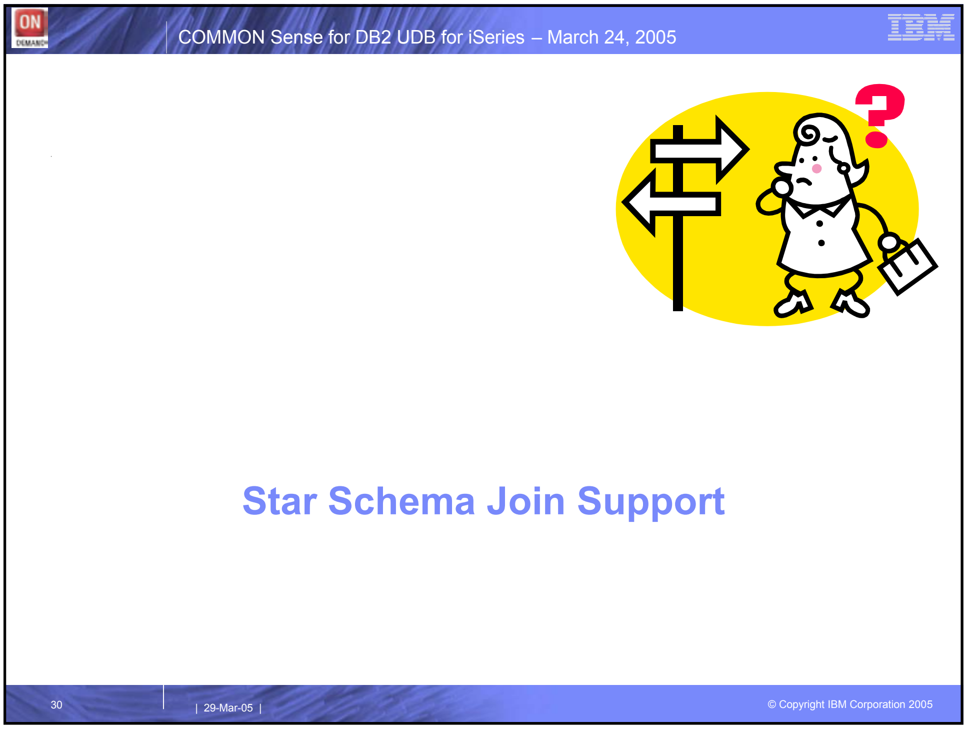
Other Indexing Tips

- **Avoid null capable columns if expecting to use index only access. Index only access is not available when a key column in the index is null capable**
- **Avoid derived expressions in local selection. Access via an index may not be used for predicates that have derived values**
- **Index access is not used for predicates where both operands come from the same table**
- **Consider index only access if all of the columns used in the query are represented in the index as key columns**
- **Use the most selective columns as keys in the index. Preference should be given to columns used in equal comparisons**
- **For key columns that are unique, specify the UNIQUE keyword when creating the index**



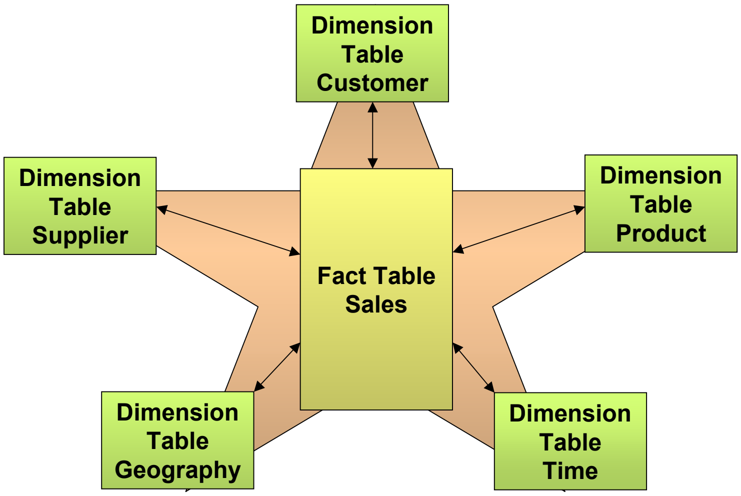
Perfect Index Guidelines

- **Order of the columns in an index is very important. Optimizer may not use an index if the columns are in an incorrect order. Use the following guideline:**
  - Equal predicates first. Predicates using the "=" operator generally eliminate the largest number of non-participating rows and should therefore be first in the index
  - If all of the predicates have an equal operator, then order the columns as follows:
    - Selection predicates + join predicates
    - Join predicates + selection predicates
    - Selection predicates + group by columns
    - Selection predicates + order by columns
- **Always place the most selective columns as the first key in the index**
- **Create perfect indexes ahead of time for pre-determined queries or queries that produce a standard report**
- **Indexes will take up system resources, find a balance between query performance and system (index) maintenance**
- ***A binary radix index is the fastest data access method available for a query that is highly selective and returns a small number of rows***



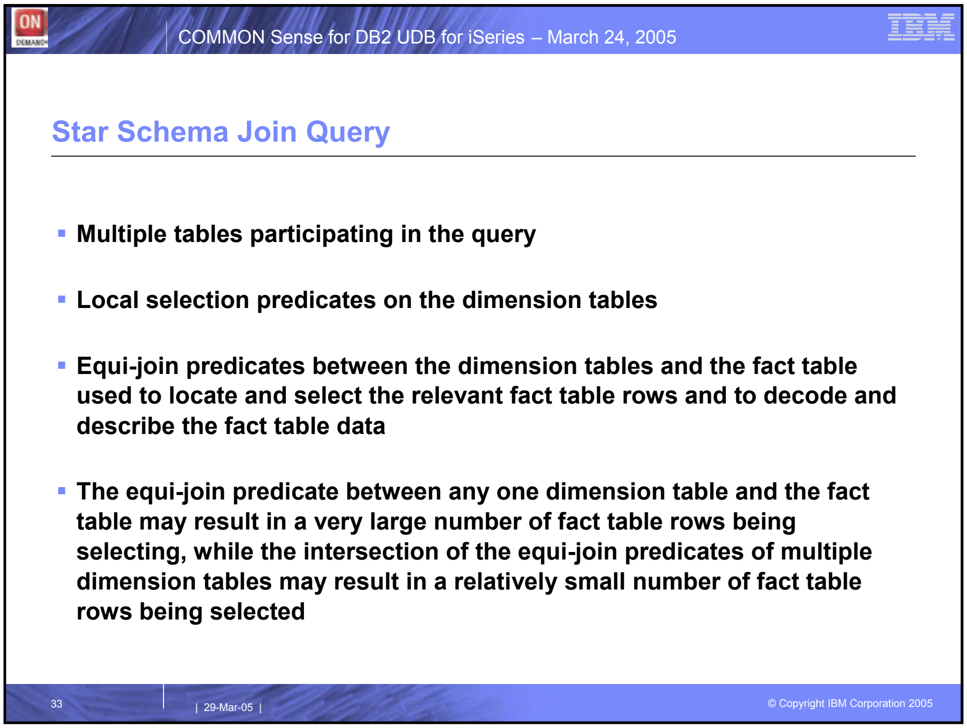
Star Schema Join Support

What is a Star Schema?



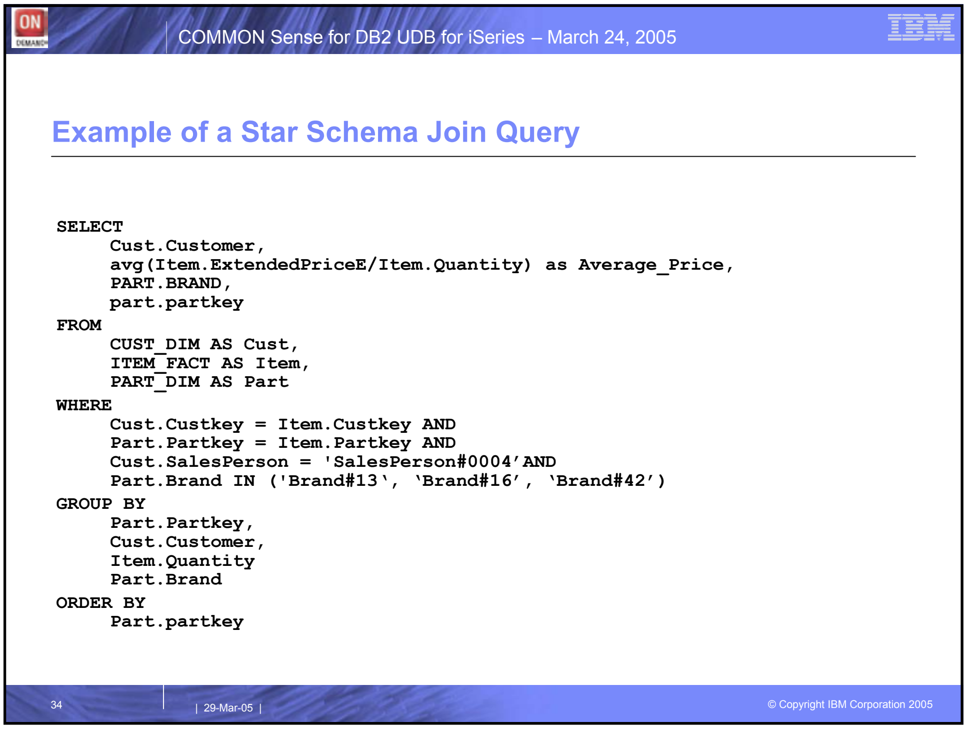
Attributes of a Star Schema

- A relatively large fact table containing millions or billions of rows holding the measurable or additive “facts” such as sales type transactions or events
- Relatively small and highly normalized dimension tables containing descriptive data about the “facts” (in the central fact table), such as customer or location information
- A central fact table which is dependent on the surrounding dimension tables using a parent / child relationship, with the fact table as the child and the dimension tables as the parent.
- If the dimension tables are further normalized, the results are dimensions that may have additional tables supporting them (a “snowflake” schema)



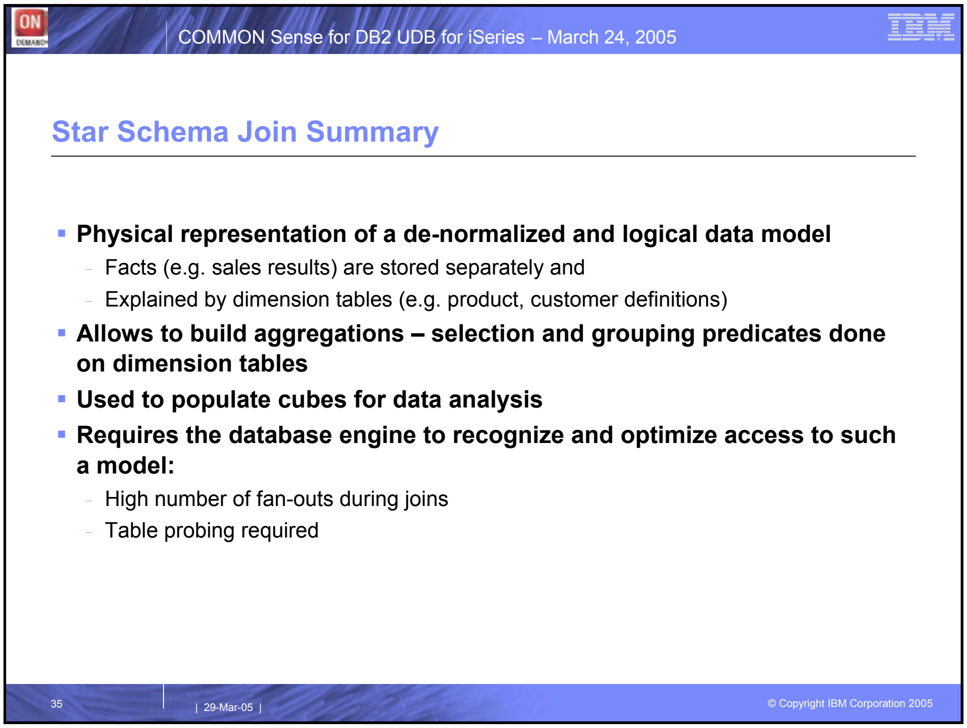
Star Schema Join Query

- Multiple tables participating in the query
- Local selection predicates on the dimension tables
- Equi-join predicates between the dimension tables and the fact table used to locate and select the relevant fact table rows and to decode and describe the fact table data
- The equi-join predicate between any one dimension table and the fact table may result in a very large number of fact table rows being selecting, while the intersection of the equi-join predicates of multiple dimension tables may result in a relatively small number of fact table rows being selected



Example of a Star Schema Join Query

```
SELECT
      Cust.Customer,
      avg(Item.ExtendedPriceE/Item.Quantity) as Average_Price,
      PART.BRAND,
      part.partkey
FROM
      CUST_DIM AS Cust,
      ITEM_FACT AS Item,
      PART_DIM AS Part
WHERE
      Cust.Custkey = Item.Custkey AND
      Part.Partkey = Item.Partkey AND
      Cust.SalesPerson = 'SalesPerson#0004’AND
      Part.Brand IN ('Brand#13‘, ‘Brand#16’, ‘Brand#42’)
GROUP BY
      Part.Partkey,
      Cust.Customer,
      Item.Quantity
      Part.Brand
ORDER BY
      Part.partkey
```

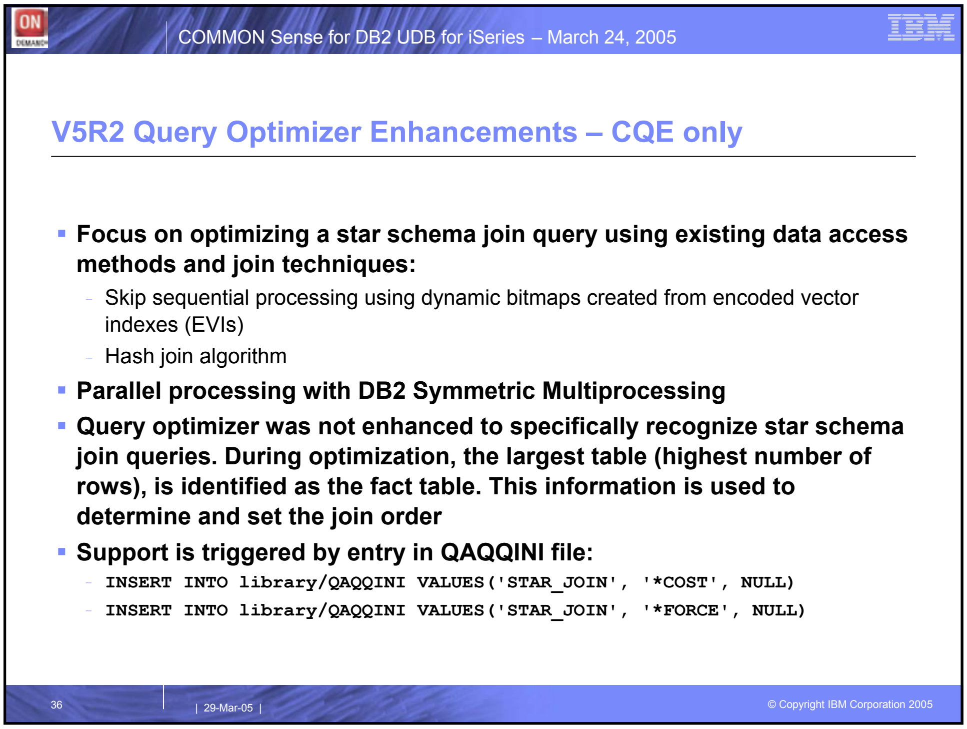


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Star Schema Join Summary

- **Physical representation of a de-normalized and logical data model**
  - Facts (e.g. sales results) are stored separately and
  - Explained by dimension tables (e.g. product, customer definitions)
- **Allows to build aggregations – selection and grouping predicates done on dimension tables**
- **Used to populate cubes for data analysis**
- **Requires the database engine to recognize and optimize access to such a model:**
  - High number of fan-outs during joins
  - Table probing required

35 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

V5R2 Query Optimizer Enhancements – CQE only

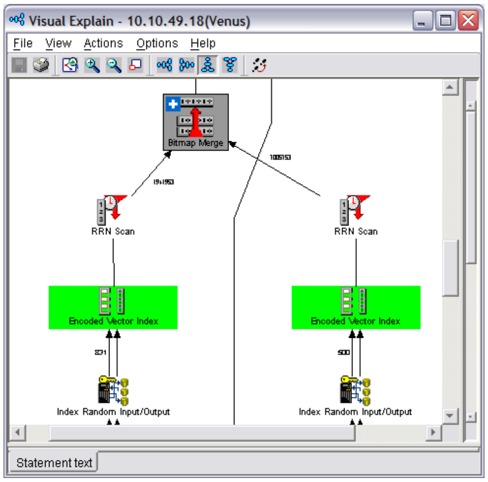
- **Focus on optimizing a star schema join query using existing data access methods and join techniques:**
  - Skip sequential processing using dynamic bitmaps created from encoded vector indexes (EVIs)
  - Hash join algorithm
- **Parallel processing with DB2 Symmetric Multiprocessing**
- **Query optimizer was not enhanced to specifically recognize star schema join queries. During optimization, the largest table (highest number of rows), is identified as the fact table. This information is used to determine and set the join order**
- **Support is triggered by entry in QAQQINI file:**
  - `INSERT INTO library/QAQQINI VALUES('STAR_JOIN', '*COST', NULL)`
  - `INSERT INTO library/QAQQINI VALUES('STAR_JOIN', '*FORCE', NULL)`

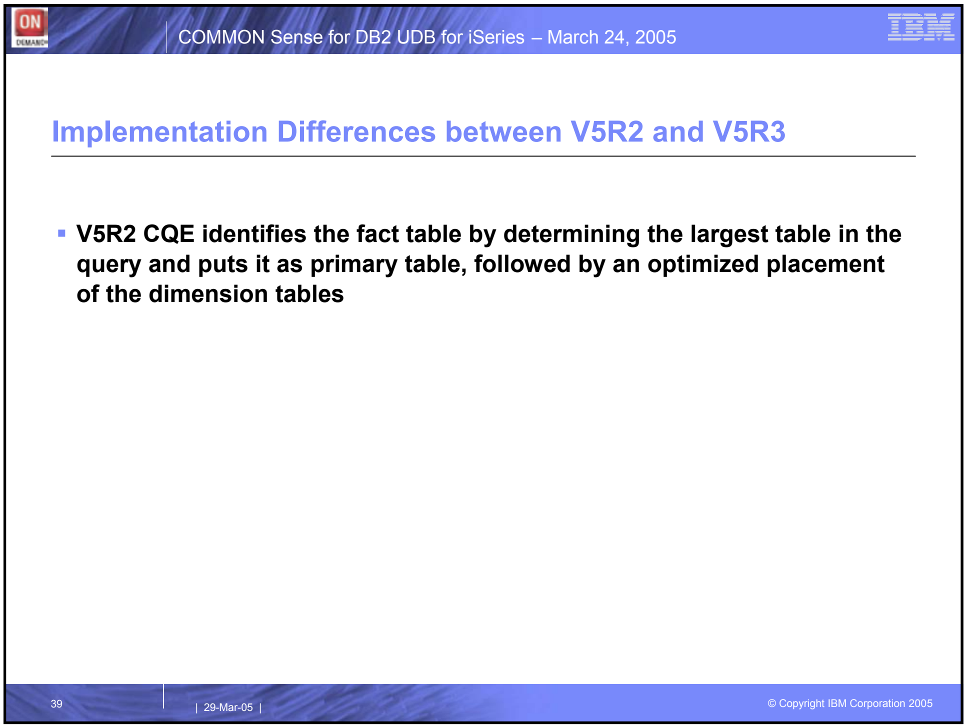
36 | 29-Mar-05 |

V5R3 Query Optimizer Enhancements - SQE

- **Automatic support for optimized star schema join support – no entries in QAQQINI required**
- **Still requires DB2 Symmetric Multiprocessing**
- **Uses "look-ahead predicate generation" (LPG) and implements parallel methods for:**
  - Selecting data from the dimension tables
  - Building the hash tables
  - Scanning the EVIs
  - Building the bitmaps or relative record number (RRN) lists
  - Selecting the data from the fact table
- **Single key EVIs created over the foreign key columns of the fact table are optimal for the SQE optimizer to implement its star schema join techniques**

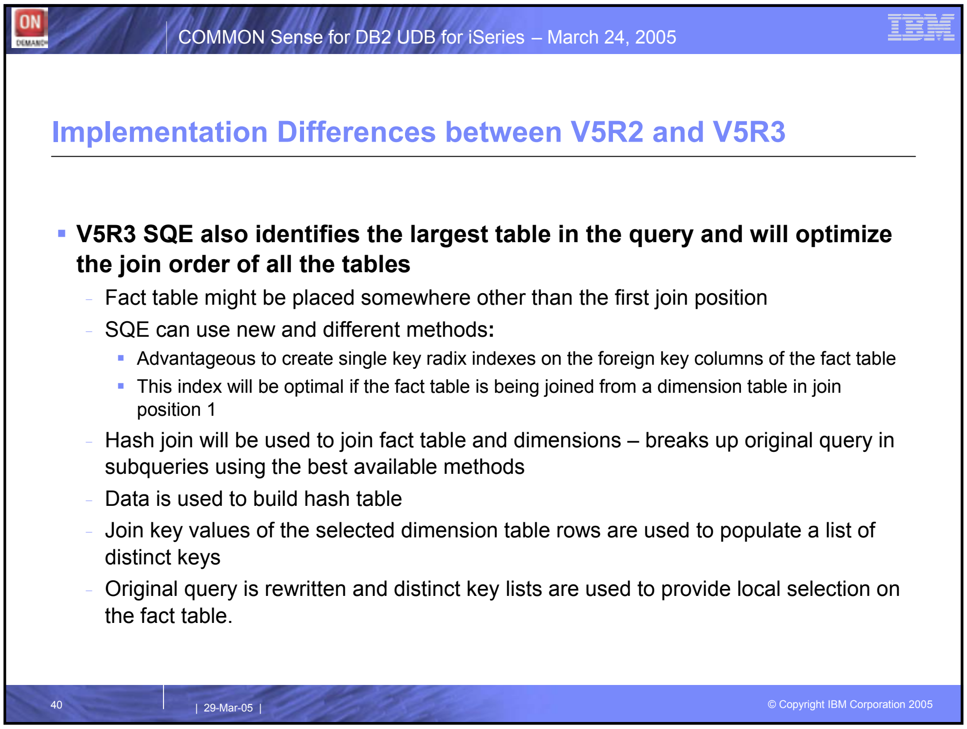
Look-ahead Predicate Generation





Implementation Differences between V5R2 and V5R3

- **V5R2 CQE identifies the fact table by determining the largest table in the query and puts it as primary table, followed by an optimized placement of the dimension tables**

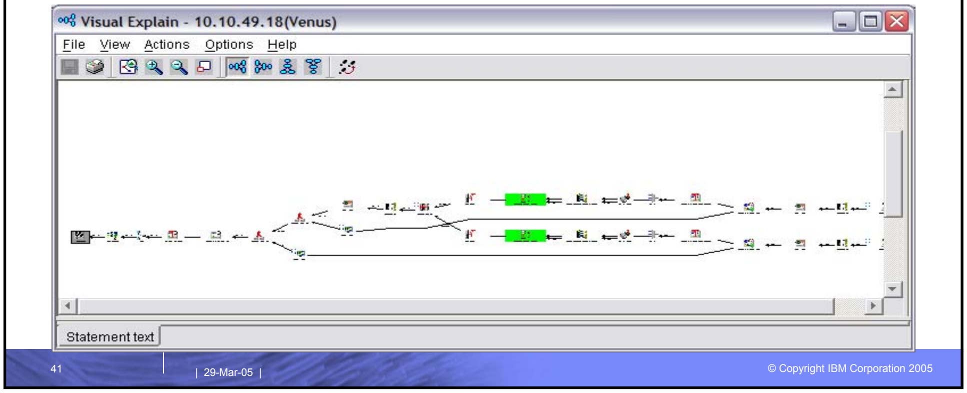


Implementation Differences between V5R2 and V5R3

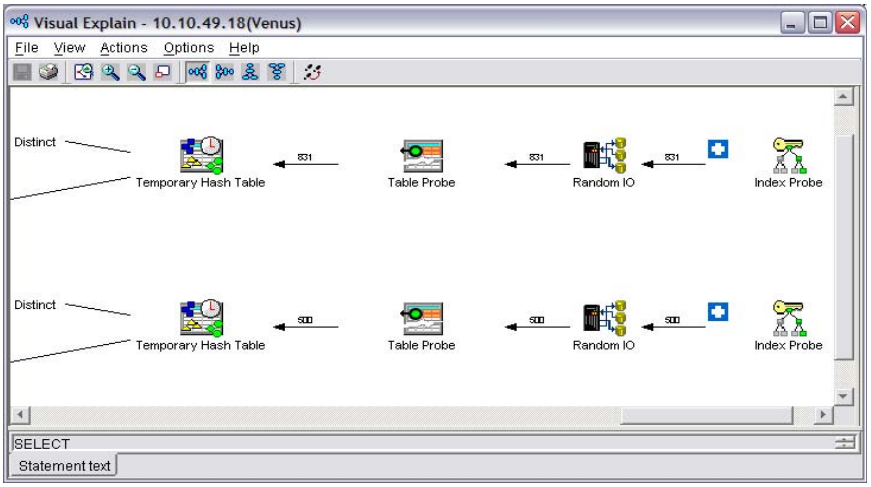
- **V5R3 SQE also identifies the largest table in the query and will optimize the join order of all the tables**
  - Fact table might be placed somewhere other than the first join position
  - SQE can use new and different methods**:**
    - Advantageous to create single key radix indexes on the foreign key columns of the fact table
    - This index will be optimal if the fact table is being joined from a dimension table in join position 1
  - Hash join will be used to join fact table and dimensions – breaks up original query in subqueries using the best available methods
  - Data is used to build hash table
  - Join key values of the selected dimension table rows are used to populate a list of distinct keys
  - Original query is rewritten and distinct key lists are used to provide local selection on the fact table.

V5R3 Implementation

```
SELECT Cust.Customer, avg(Item.ExtendedPriceE/Item.Quantity) as Average_Price, PART.BRAND,
part.partkey
FROM CUST_DIM AS Cust, ITEM_FACT AS Item, PART_DIM AS Part
WHERE Cust.Custkey = Item.Custkey AND Part.Partkey = Item.Partkey AND Cust.SalesPerson =
'SalesPerson#0004'AND Part.Brand IN ('Brand#13‘, ‘Brand#16’, ‘Brand#42’)
GROUP BY Part.Partkey, Cust.Customer, Item.Quantity, Part.Brand
ORDER BY Part.partkey
```

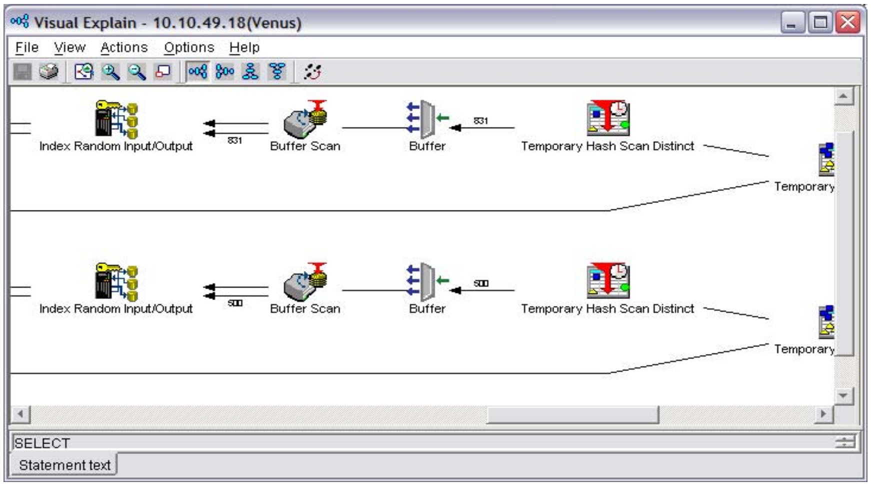

Visual Explain - 10.10.49.18(Venus)
File View Actions Options Help
Statement text

V5R3 Implementation



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

V5R3 Implementation


Visual Explain - 10.10.49.18(Venus)
File
View
Actions
Options
Help
Index Random Input/Output
831
Buffer Scan
Buffer
831
Temporary Hash Scan Distinct
Temporary
Index Random Input/Output
500
Buffer Scan
Buffer
500
Temporary Hash Scan Distinct
Temporary
Statement text

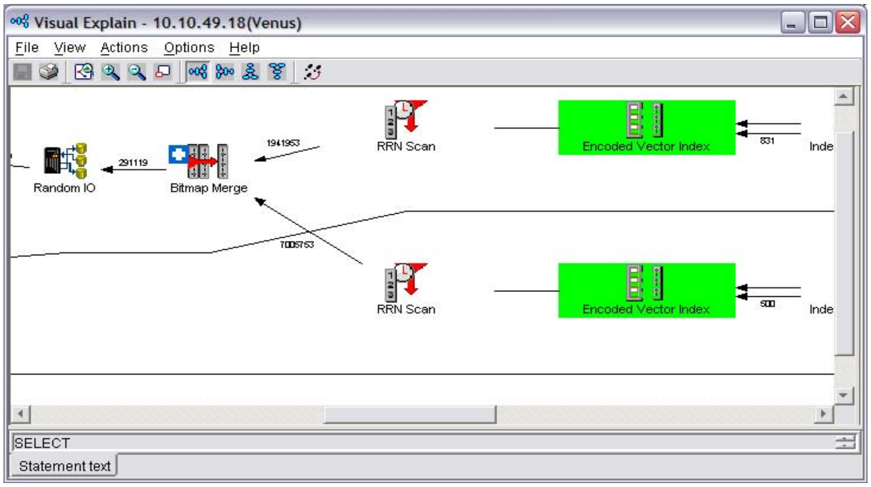
SELECT

43 | 29-Mar-05 |

© Copyright IBM Corporation 2005

COMMON Sense for DB2 UDB for iSeries – March 24, 2005

V5R3 Implementation



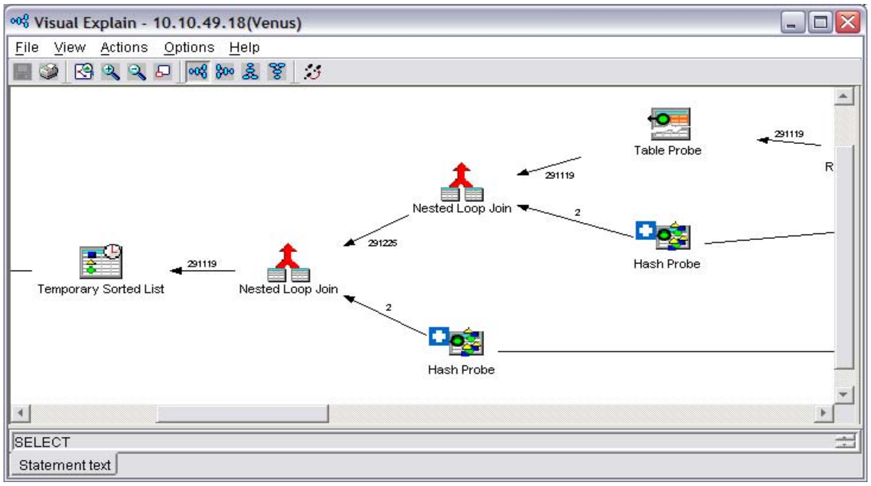
SELECT

44 | 29-Mar-05 |

© Copyright IBM Corporation 2005

COMMON Sense for DB2 UDB for iSeries – March 24, 2005

V5R3 Implementation

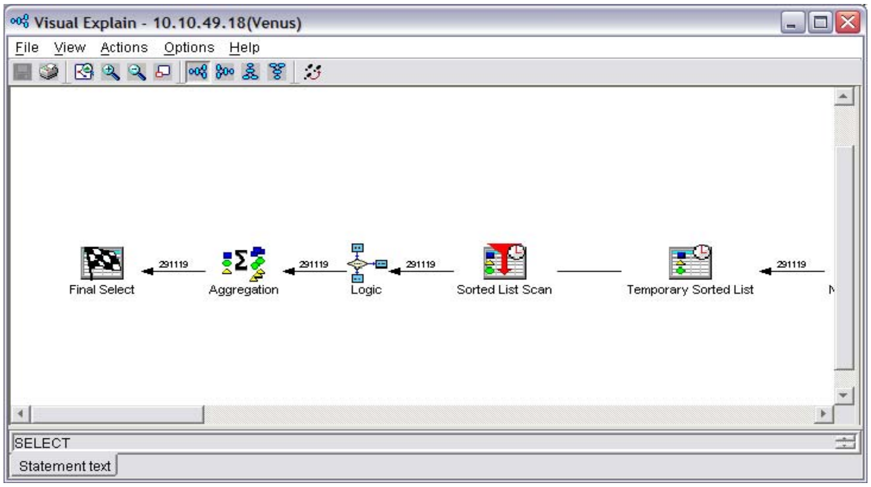


SELECT

45 | 29-Mar-05 |

COMMON Sense for DB2 UDB for iSeries – March 24, 2005

V5R3 Implementation



SELECT

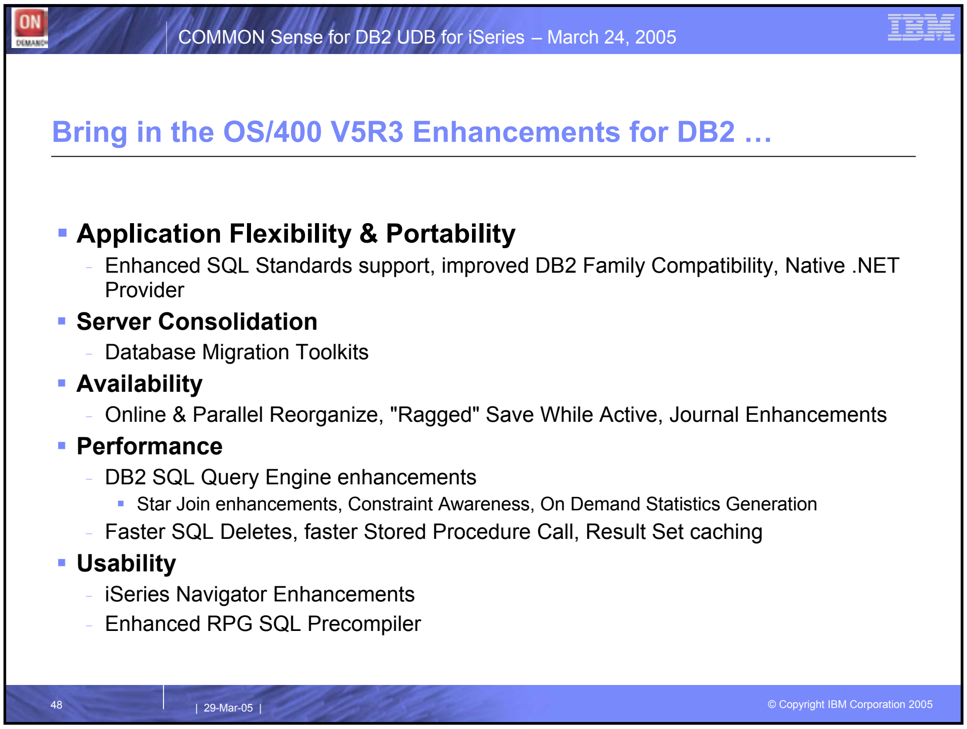
46 | 29-Mar-05 |

COMMON Sense for DB2 UDB for iSeries – March 24, 2005

What's new for DB2 with OS/400 V5R3 in this Strange World We Live In

47 | 29-Mar-05 |

© Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Bring in the OS/400 V5R3 Enhancements for DB2 …

- **Application Flexibility & Portability**
  - Enhanced SQL Standards support, improved DB2 Family Compatibility, Native .NET Provider
- **Server Consolidation**
  - Database Migration Toolkits
- **Availability**
  - Online & Parallel Reorganize, "Ragged" Save While Active, Journal Enhancements
- **Performance**
  - DB2 SQL Query Engine enhancements
    - Star Join enhancements, Constraint Awareness, On Demand Statistics Generation
  - Faster SQL Deletes, faster Stored Procedure Call, Result Set caching
- **Usability**
  - iSeries Navigator Enhancements
  - Enhanced RPG SQL Precompiler

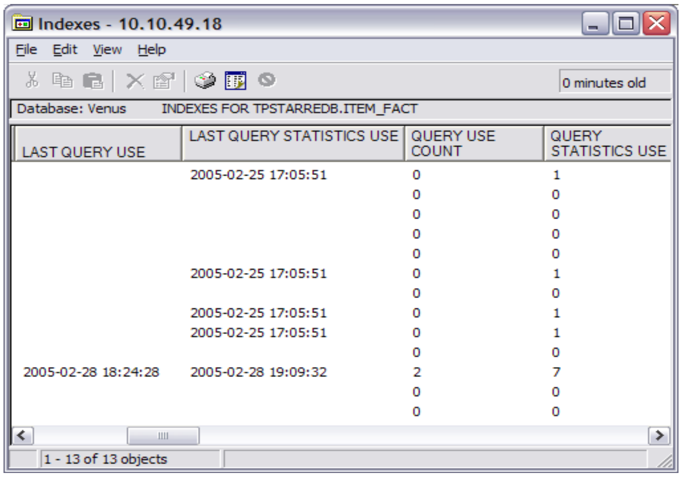
48 | 29-Mar-05 |

© Copyright IBM Corporation 2005

Get Rid of Those Damn' Indexes

- **Obtain global view of all types of indexes on a given table**
- **Assess whether or not it is being used by query functions:**
  - Last Query Use: timestamp when the index was last used to access tables in a query
  - Last Query Statistic Use: timestamp when the index was last used to gather statistical information
  - Query Use Count: number of instances the index was used in a query
  - Query Statistics Use: number of instances the index was used for statistical information
- **Counters are updated regardless of Query Engine used**

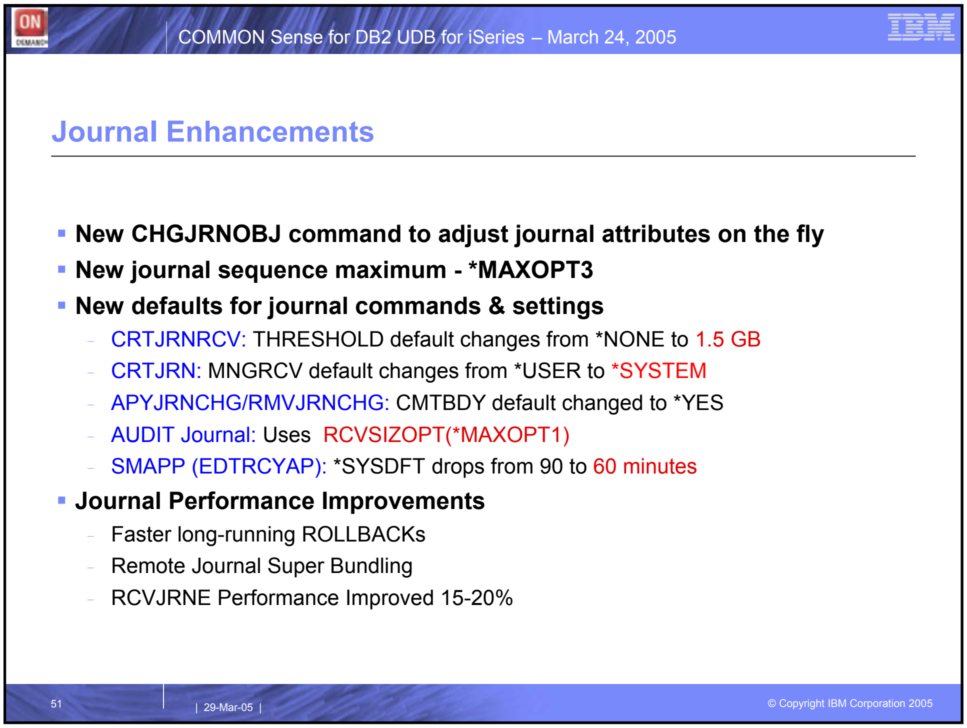
iSeries Navigator Interface



Database: Venus INDEXES FOR TPSTARREDB.ITEM_FACT

LAST QUERY USE	LAST QUERY STATISTICS USE	QUERY USE COUNT	QUERY STATISTICS USE
	2005-02-25 17:05:51	0	1
		0	0
		0	0
		0	0
		0	0
	2005-02-25 17:05:51	0	1
		0	0
	2005-02-25 17:05:51	0	1
	2005-02-25 17:05:51	0	1
		0	0
2005-02-28 18:24:28	2005-02-28 19:09:32	2	7
		0	0
		0	0

1 - 13 of 13 objects

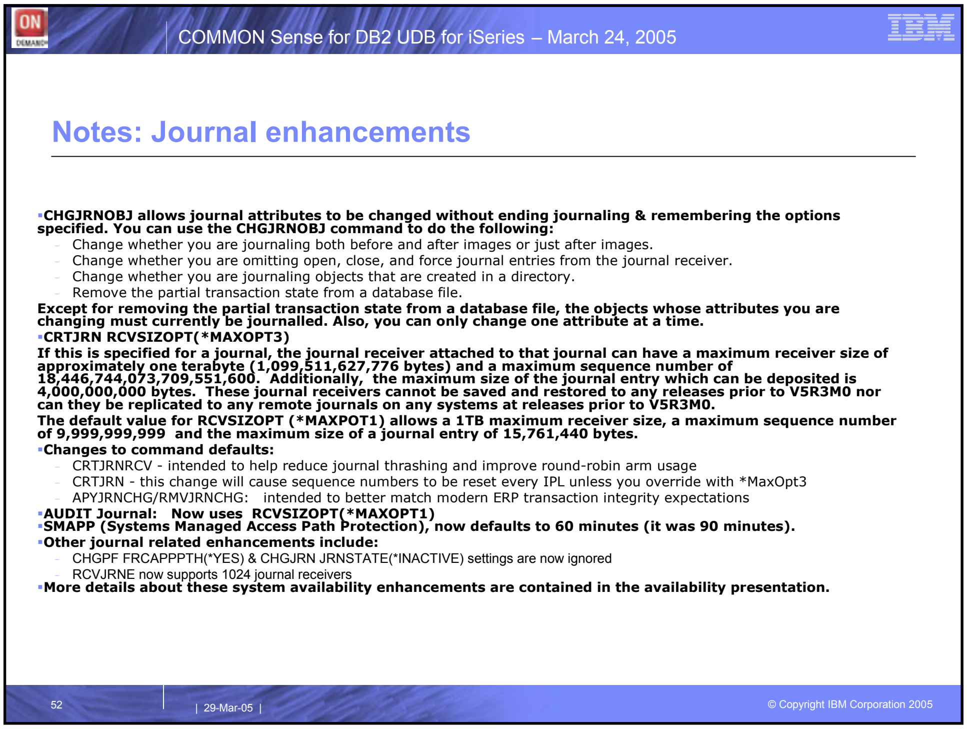


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Journal Enhancements

- **New CHGJRNOBJ command to adjust journal attributes on the fly**
- **New journal sequence maximum - *MAXOPT3**
- **New defaults for journal commands & settings**
  - CRTJRNRCV: THRESHOLD default changes from *NONE to 1.5 GB
  - CRTJRN: MNGRCV default changes from *USER to *SYSTEM
  - APYJRNCHG/RMVJRNCHG: CMTBDY default changed to *YES
  - AUDIT Journal: Uses RCVSIZOPT(*MAXOPT1)
  - SMAPP (EDTRCYAP): *SYSDFT drops from 90 to 60 minutes
- **Journal Performance Improvements**
  - Faster long-running ROLLBACKs
  - Remote Journal Super Bundling
  - RCVJRNE Performance Improved 15-20%

51 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Notes: Journal enhancements

- **CHGJRNOBJ allows journal attributes to be changed without ending journaling & remembering the options specified. You can use the CHGJRNOBJ command to do the following:**
  - Change whether you are journaling both before and after images or just after images.
  - Change whether you are omitting open, close, and force journal entries from the journal receiver.
  - Change whether you are journaling objects that are created in a directory.
  - Remove the partial transaction state from a database file.

Except for removing the partial transaction state from a database file, the objects whose attributes you are changing must currently be journalled. Also, you can only change one attribute at a time.

- **CRTJRN RCVSIZOPT(*MAXOPT3)**

If this is specified for a journal, the journal receiver attached to that journal can have a maximum receiver size of approximately one terabyte (1,099,511,627,776 bytes) and a maximum sequence number of 18,446,744,073,709,551,600. Additionally, the maximum size of the journal entry which can be deposited is 4,000,000,000 bytes. These journal receivers cannot be saved and restored to any releases prior to V5R3M0 nor can they be replicated to any remote journals on any systems at releases prior to V5R3M0.

The default value for RCVSIZOPT (*MAXPOT1) allows a 1TB maximum receiver size, a maximum sequence number of 9,999,999,999 and the maximum size of a journal entry of 15,761,440 bytes.

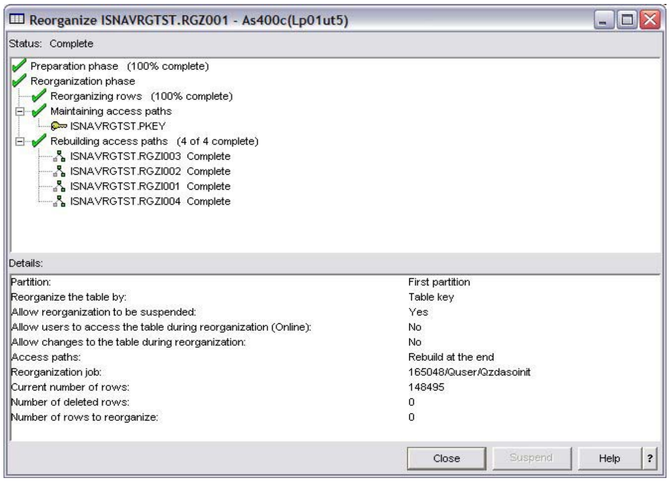
- **Changes to command defaults:**
  - CRTJRNRCV - intended to help reduce journal thrashing and improve round-robin arm usage
  - CRTJRN - this change will cause sequence numbers to be reset every IPL unless you override with *MaxOpt3
  - APYJRNCHG/RMVJRNCHG: intended to better match modern ERP transaction integrity expectations
- **AUDIT Journal: Now uses RCVSIZOPT(*MAXOPT1)**
- **SMAPP (Systems Managed Access Path Protection), now defaults to 60 minutes (it was 90 minutes).**
- **Other journal related enhancements include:**
  - CHGPF FRCAPPPTH(*YES) & CHGJRN JRNSTATE(*INACTIVE) settings are now ignored
  - RCVJRNE now supports 1024 journal receivers
- **More details about these system availability enhancements are contained in the availability presentation.**

52 | 29-Mar-05 |

Online and Parallel Reorganize

- **New Reorganize capabilities activated with new parameter ALWCANCEL(*YES)**
  - ALWCANCEL(*YES) requires file to be journaled
  - New parameter, LOCK, controls the concurrent access
  - If Exclusive lock not requested, then row order may be different & space may not be reclaimed
- **Parallel capabilities rely on DB2 SMP licensed feature being installed & activated**
- **New Index Rebuild parameter RBDACCPTH**
- **RI & Unique indexes always maintained**

Online and Parallel Reorganize



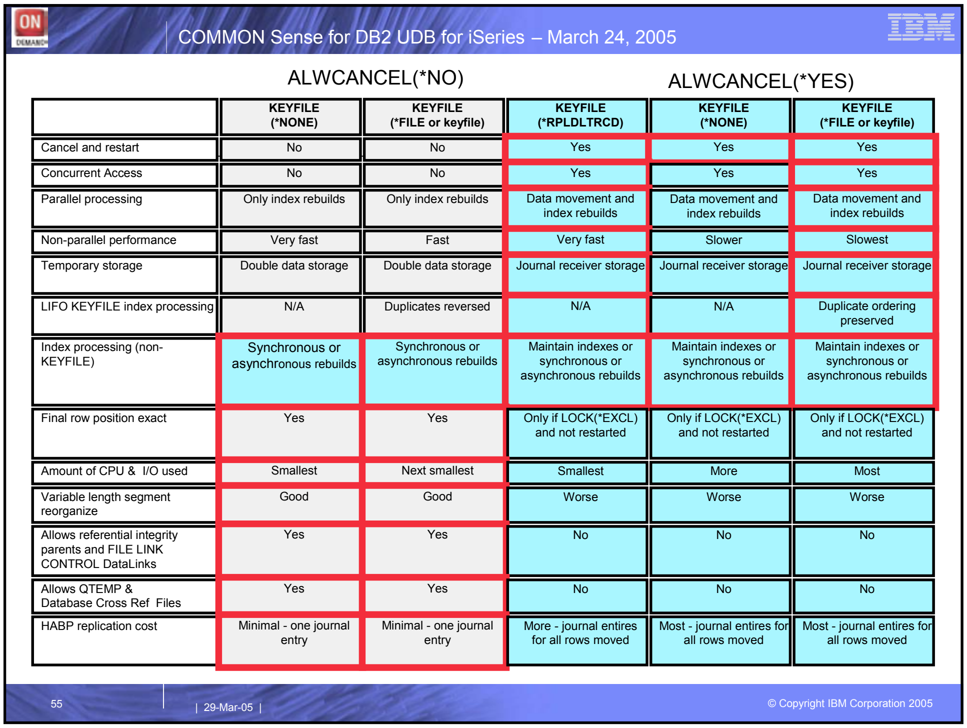
Reorganize ISNAVRGTST.RGZ001 - As400c(Lp01ut5)

Status: Complete

```
✓ Preparation phase   (100% complete)
✓ Reorganization phase
    ✓ Reorganizing rows   (100% complete)
  ⊟ ✓ Maintaining access paths
        ISNAVRGTST.PKEY
  ⊟ ✓ Rebuilding access paths   (4 of 4 complete)
        ISNAVRGTST.RGZI003  Complete
        ISNAVRGTST.RGZI002  Complete
        ISNAVRGTST.RGZI001  Complete
        ISNAVRGTST.RGZI004  Complete
```

Details:

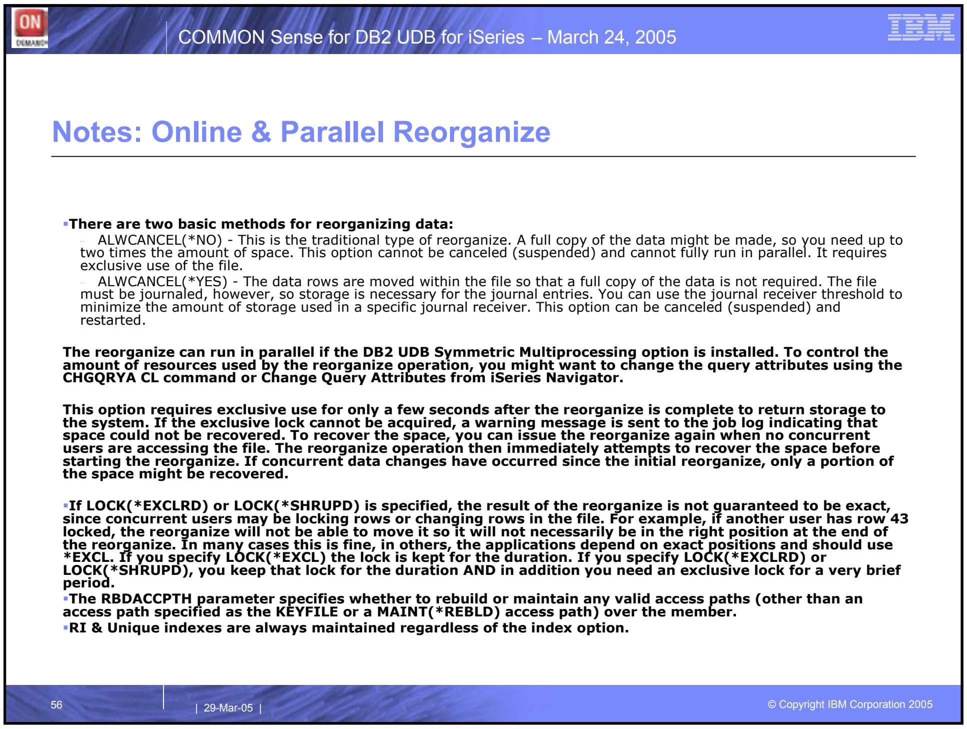
Partition:	First partition
Reorganize the table by:	Table key
Allow reorganization to be suspended:	Yes
Allow users to access the table during reorganization (Online):	No
Allow changes to the table during reorganization:	No
Access paths:	Rebuild at the end
Reorganization job:	165048/Quser/Qzdasoinit
Current number of rows:	148495
Number of deleted rows:	0
Number of rows to reorganize:	0



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

	ALWCANCEL(*NO)		ALWCANCEL(*YES)		
	KEYFILE (*NONE)	**KEYFILE (*FILE or keyfile)**	**KEYFILE (*RPLDLTRCD)**	**KEYFILE (*NONE)**	**KEYFILE (*FILE or keyfile)**
Cancel and restart	No	No	Yes	Yes	Yes
Concurrent Access	No	No	Yes	Yes	Yes
Parallel processing	Only index rebuilds	Only index rebuilds	Data movement and index rebuilds	Data movement and index rebuilds	Data movement and index rebuilds
Non-parallel performance	Very fast	Fast	Very fast	Slower	Slowest
Temporary storage	Double data storage	Double data storage	Journal receiver storage	Journal receiver storage	Journal receiver storage
LIFO KEYFILE index processing	N/A	Duplicates reversed	N/A	N/A	Duplicate ordering preserved
Index processing (non-KEYFILE)	Synchronous or asynchronous rebuilds	Synchronous or asynchronous rebuilds	Maintain indexes or synchronous or asynchronous rebuilds	Maintain indexes or synchronous or asynchronous rebuilds	Maintain indexes or synchronous or asynchronous rebuilds
Final row position exact	Yes	Yes	Only if LOCK(*EXCL) and not restarted	Only if LOCK(*EXCL) and not restarted	Only if LOCK(*EXCL) and not restarted
Amount of CPU & I/O used	Smallest	Next smallest	Smallest	More	Most
Variable length segment reorganize	Good	Good	Worse	Worse	Worse
Allows referential integrity parents and FILE LINK CONTROL DataLinks	Yes	Yes	No	No	No
Allows QTEMP & Database Cross Ref Files	Yes	Yes	No	No	No
HABP replication cost	Minimal - one journal entry	Minimal - one journal entry	More - journal entires for all rows moved	Most - journal entires for all rows moved	Most - journal entires for all rows moved

55 | 29-Mar-05 | © Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Notes: Online & Parallel Reorganize

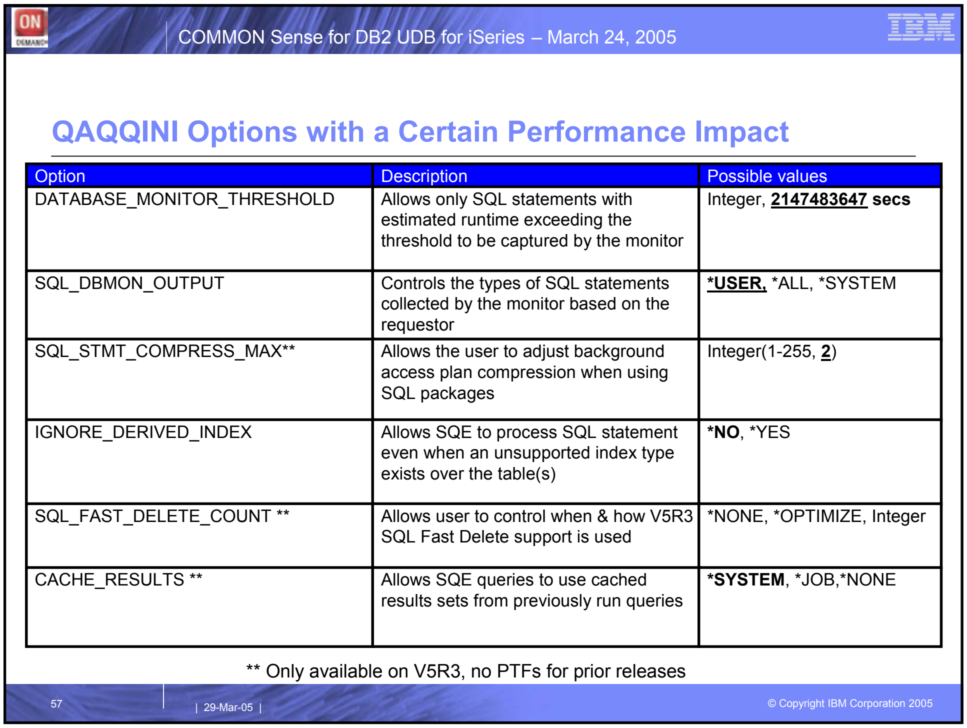
- **There are two basic methods for reorganizing data:**
  - ALWCANCEL(*NO) - This is the traditional type of reorganize. A full copy of the data might be made, so you need up to two times the amount of space. This option cannot be canceled (suspended) and cannot fully run in parallel. It requires exclusive use of the file.
  - ALWCANCEL(*YES) - The data rows are moved within the file so that a full copy of the data is not required. The file must be journaled, however, so storage is necessary for the journal entries. You can use the journal receiver threshold to minimize the amount of storage used in a specific journal receiver. This option can be canceled (suspended) and restarted.

The reorganize can run in parallel if the DB2 UDB Symmetric Multiprocessing option is installed. To control the amount of resources used by the reorganize operation, you might want to change the query attributes using the CHGQRYA CL command or Change Query Attributes from iSeries Navigator.

This option requires exclusive use for only a few seconds after the reorganize is complete to return storage to the system. If the exclusive lock cannot be acquired, a warning message is sent to the job log indicating that space could not be recovered. To recover the space, you can issue the reorganize again when no concurrent users are accessing the file. The reorganize operation then immediately attempts to recover the space before starting the reorganize. If concurrent data changes have occurred since the initial reorganize, only a portion of the space might be recovered.

- **If LOCK(*EXCLRD) or LOCK(*SHRUPD) is specified, the result of the reorganize is not guaranteed to be exact, since concurrent users may be locking rows or changing rows in the file. For example, if another user has row 43 locked, the reorganize will not be able to move it so it will not necessarily be in the right position at the end of the reorganize. In many cases this is fine, in others, the applications depend on exact positions and should use *EXCL. If you specify LOCK(*EXCL) the lock is kept for the duration. If you specify LOCK(*EXCLRD) or LOCK(*SHRUPD), you keep that lock for the duration AND in addition you need an exclusive lock for a very brief period.**
- **The RBDACCPTH parameter specifies whether to rebuild or maintain any valid access paths (other than an access path specified as the KEYFILE or a MAINT(*REBLD) access path) over the member.**
- **RI & Unique indexes are always maintained regardless of the index option.**

56 | 29-Mar-05 | © Copyright IBM Corporation 2005



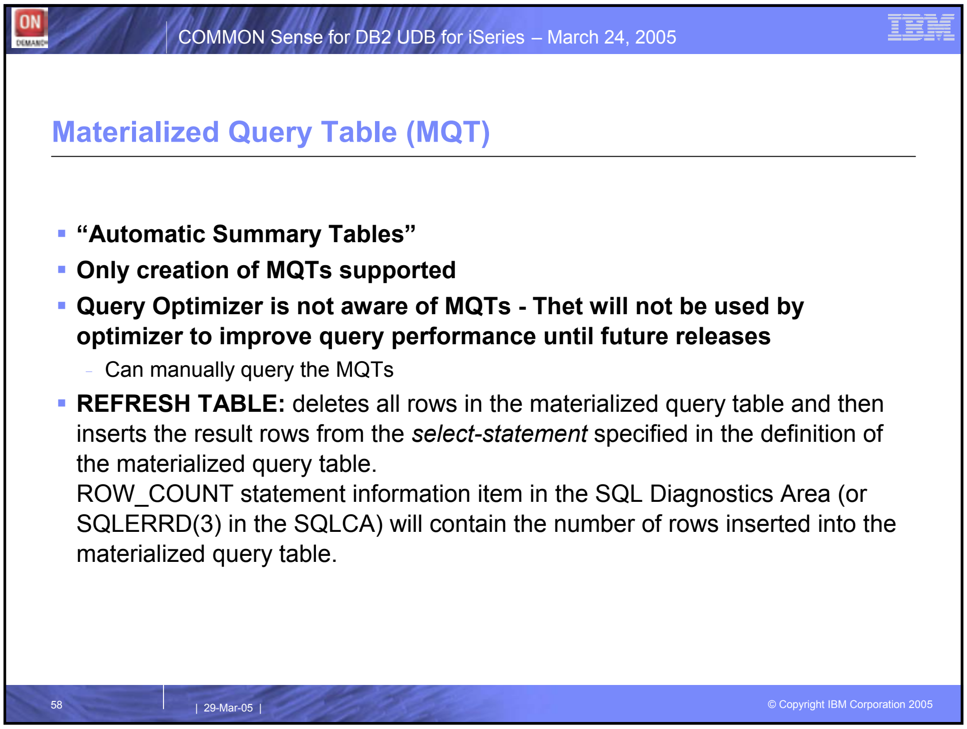
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

QAQQINI Options with a Certain Performance Impact

Option	Description	Possible values
DATABASE_MONITOR_THRESHOLD	Allows only SQL statements with estimated runtime exceeding the threshold to be captured by the monitor	Integer, **2147483647 secs**
SQL_DBMON_OUTPUT	Controls the types of SQL statements collected by the monitor based on the requestor	***USER,** *ALL, *SYSTEM
SQL_STMT_COMPRESS_MAX**	Allows the user to adjust background access plan compression when using SQL packages	Integer(1-255, **2**)
IGNORE_DERIVED_INDEX	Allows SQE to process SQL statement even when an unsupported index type exists over the table(s)	***NO**, *YES
SQL_FAST_DELETE_COUNT **	Allows user to control when & how V5R3 SQL Fast Delete support is used	*NONE, *OPTIMIZE, Integer
CACHE_RESULTS **	Allows SQE queries to use cached results sets from previously run queries	***SYSTEM**, *JOB,*NONE

** Only available on V5R3, no PTFs for prior releases

57 | 29-Mar-05 | © Copyright IBM Corporation 2005

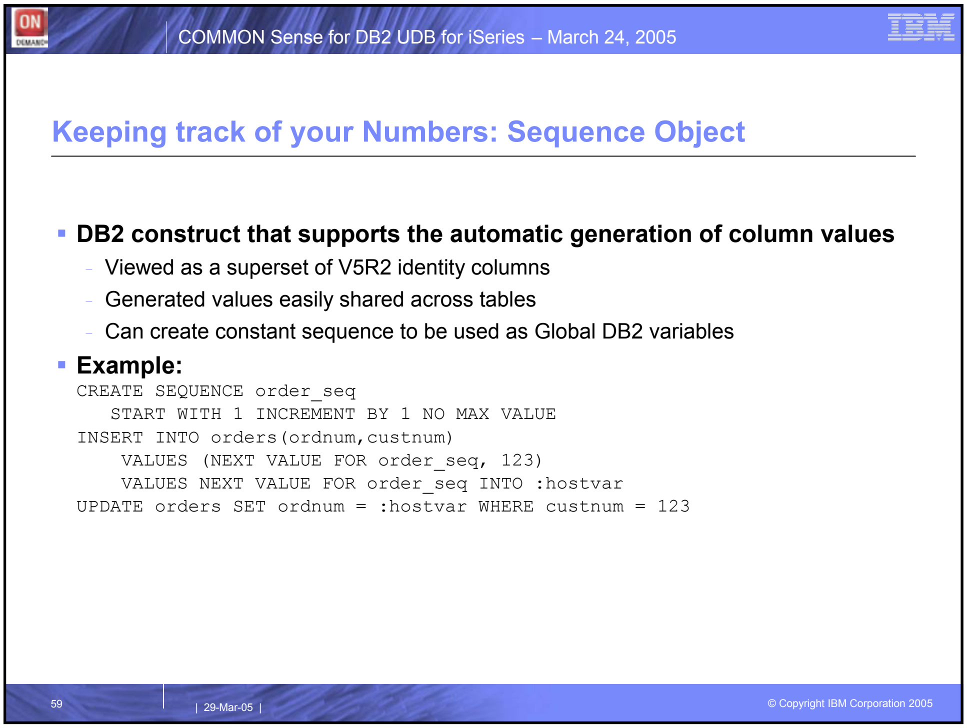


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Materialized Query Table (MQT)

- **“Automatic Summary Tables”**
- **Only creation of MQTs supported**
- **Query Optimizer is not aware of MQTs - Thet will not be used by optimizer to improve query performance until future releases**
  - Can manually query the MQTs
- **REFRESH TABLE:** deletes all rows in the materialized query table and then inserts the result rows from the *select-statement* specified in the definition of the materialized query table.
  ROW_COUNT statement information item in the SQL Diagnostics Area (or SQLERRD(3) in the SQLCA) will contain the number of rows inserted into the materialized query table.

58 | 29-Mar-05 | © Copyright IBM Corporation 2005



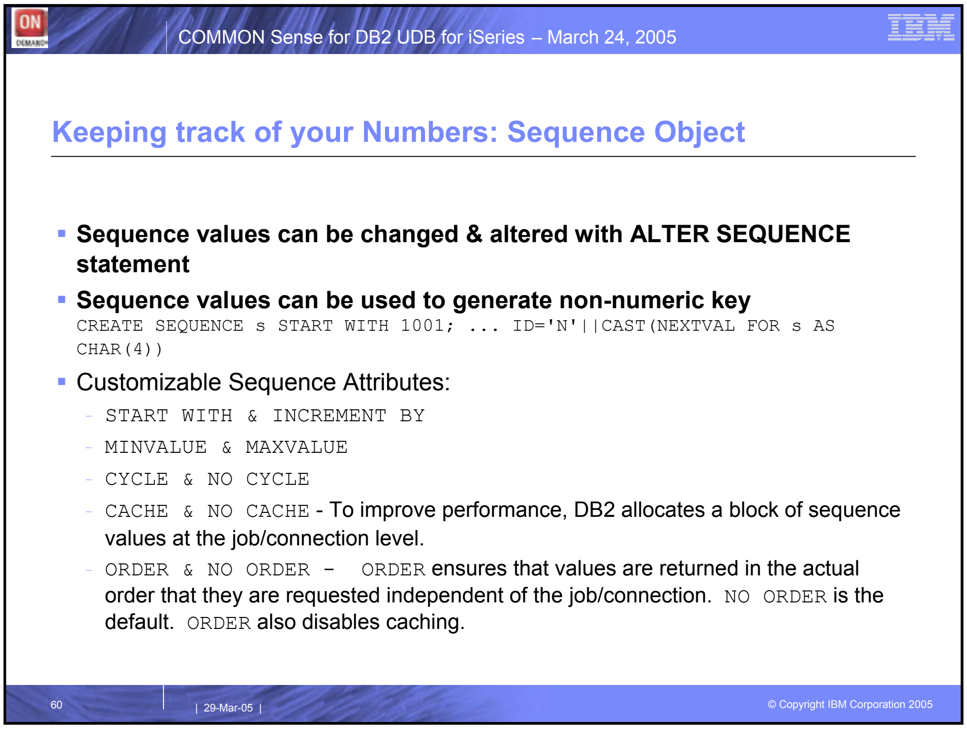
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Keeping track of your Numbers: Sequence Object

- **DB2 construct that supports the automatic generation of column values**
  - Viewed as a superset of V5R2 identity columns
  - Generated values easily shared across tables
  - Can create constant sequence to be used as Global DB2 variables
- **Example:**

```
CREATE SEQUENCE order_seq
   START WITH 1 INCREMENT BY 1 NO MAX VALUE
INSERT INTO orders(ordnum,custnum)
    VALUES (NEXT VALUE FOR order_seq, 123)
    VALUES NEXT VALUE FOR order_seq INTO :hostvar
UPDATE orders SET ordnum = :hostvar WHERE custnum = 123
```

59 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

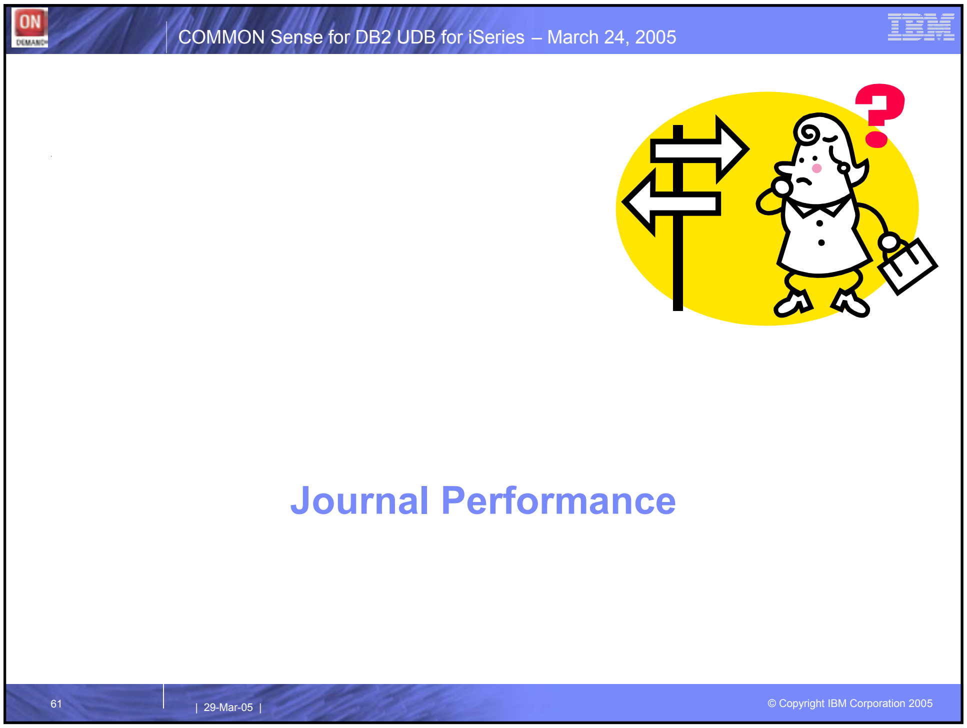
Keeping track of your Numbers: Sequence Object

- **Sequence values can be changed & altered with ALTER SEQUENCE statement**
- **Sequence values can be used to generate non-numeric key**

```
CREATE SEQUENCE s START WITH 1001; ... ID='N'||CAST(NEXTVAL FOR s AS
CHAR(4))
```

- Customizable Sequence Attributes:
  - `START WITH & INCREMENT BY`
  - `MINVALUE & MAXVALUE`
  - `CYCLE & NO CYCLE`
  - `CACHE & NO CACHE` - To improve performance, DB2 allocates a block of sequence values at the job/connection level.
  - `ORDER & NO ORDER –` `ORDER` ensures that values are returned in the actual order that they are requested independent of the job/connection. `NO ORDER` is the default. `ORDER` also disables caching.

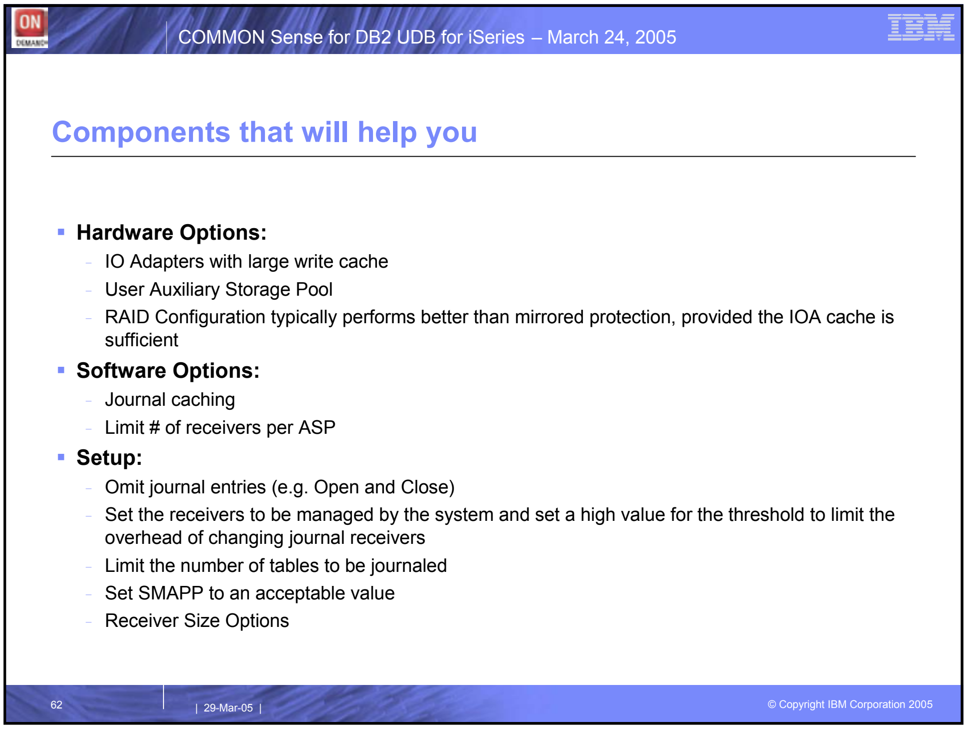
60 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Journal Performance

61 | 29-Mar-05 |

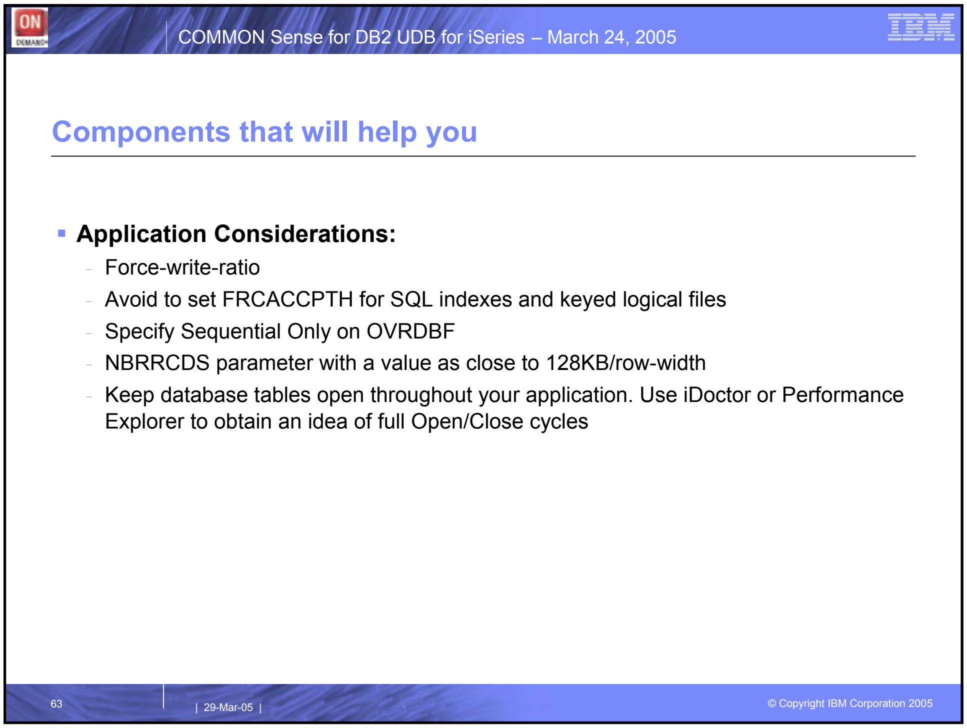


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Components that will help you

- **Hardware Options:**
  - IO Adapters with large write cache
  - User Auxiliary Storage Pool
  - RAID Configuration typically performs better than mirrored protection, provided the IOA cache is sufficient
- **Software Options:**
  - Journal caching
  - Limit # of receivers per ASP
- **Setup:**
  - Omit journal entries (e.g. Open and Close)
  - Set the receivers to be managed by the system and set a high value for the threshold to limit the overhead of changing journal receivers
  - Limit the number of tables to be journaled
  - Set SMAPP to an acceptable value
  - Receiver Size Options

62 | 29-Mar-05 |

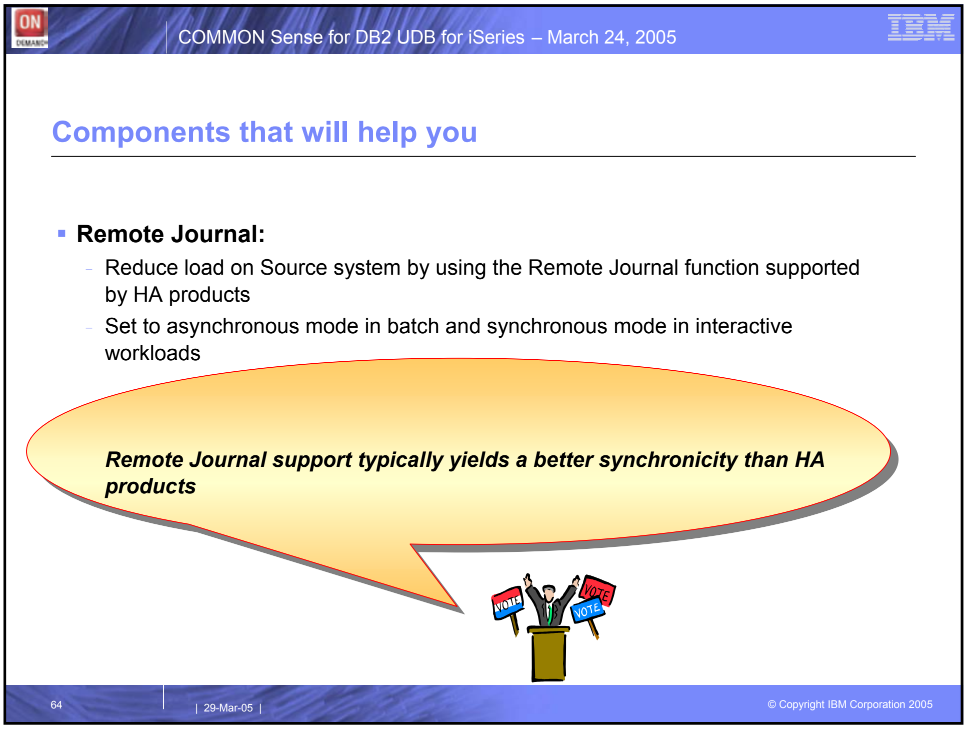


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Components that will help you

- **Application Considerations:**
  - Force-write-ratio
  - Avoid to set FRCACCPTH for SQL indexes and keyed logical files
  - Specify Sequential Only on OVRDBF
  - NBRRCDS parameter with a value as close to 128KB/row-width
  - Keep database tables open throughout your application. Use iDoctor or Performance Explorer to obtain an idea of full Open/Close cycles

63 | 29-Mar-05 | © Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

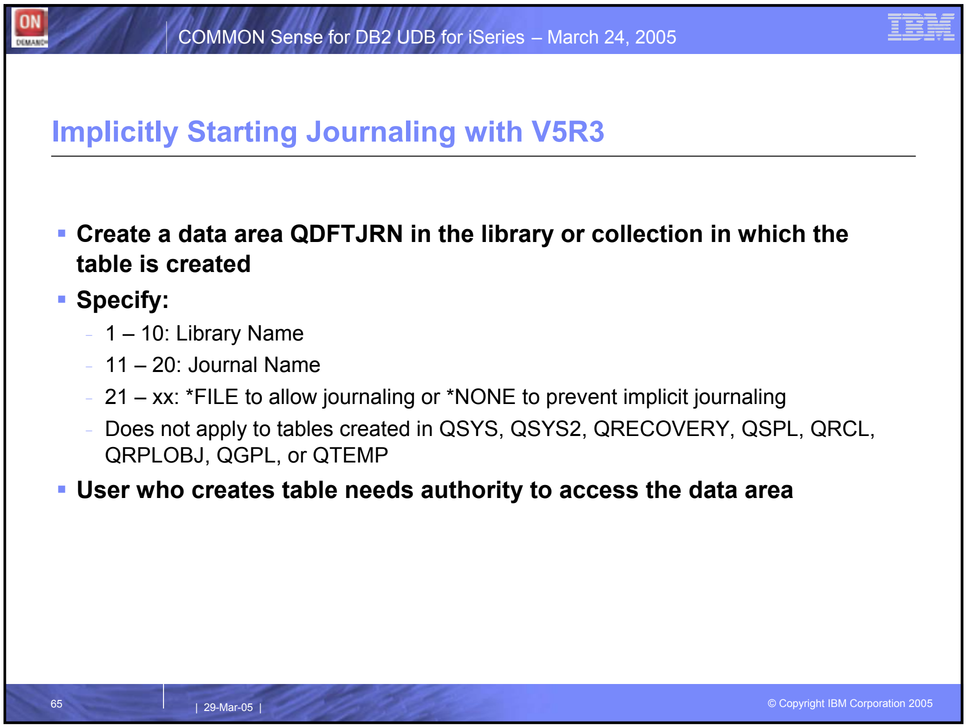
Components that will help you

- **Remote Journal:**
  - Reduce load on Source system by using the Remote Journal function supported by HA products
  - Set to asynchronous mode in batch and synchronous mode in interactive workloads

Remote Journal support typically yields a better synchronicity than HA products

VOTE
VOTE
VOTE

64 | 29-Mar-05 | © Copyright IBM Corporation 2005

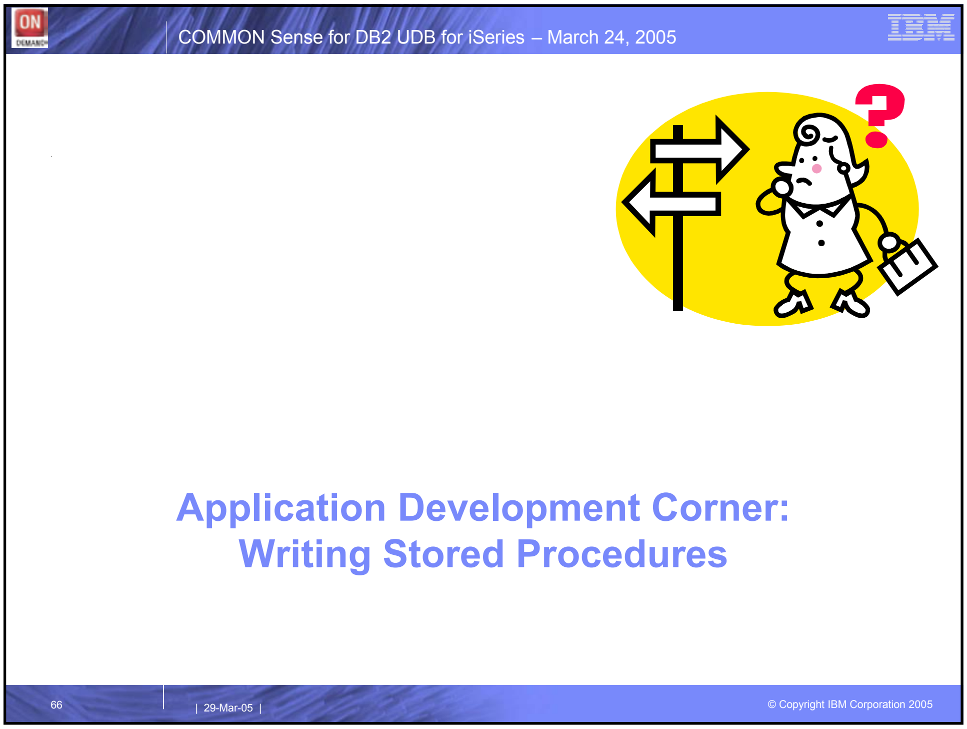


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Implicitly Starting Journaling with V5R3

- **Create a data area QDFTJRN in the library or collection in which the table is created**
- **Specify:**
  - 1 – 10: Library Name
  - 11 – 20: Journal Name
  - 21 – xx: *FILE to allow journaling or *NONE to prevent implicit journaling
  - Does not apply to tables created in QSYS, QSYS2, QRECOVERY, QSPL, QRCL, QRPLOBJ, QGPL, or QTEMP
- **User who creates table needs authority to access the data area**

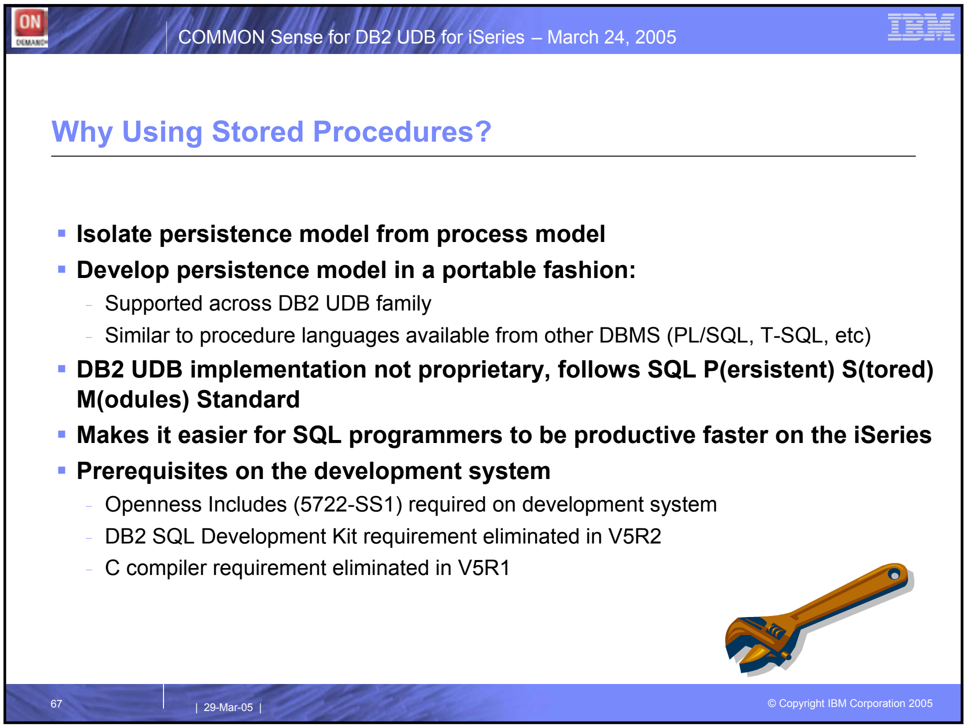
65 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Application Development Corner: Writing Stored Procedures

66 | 29-Mar-05 |

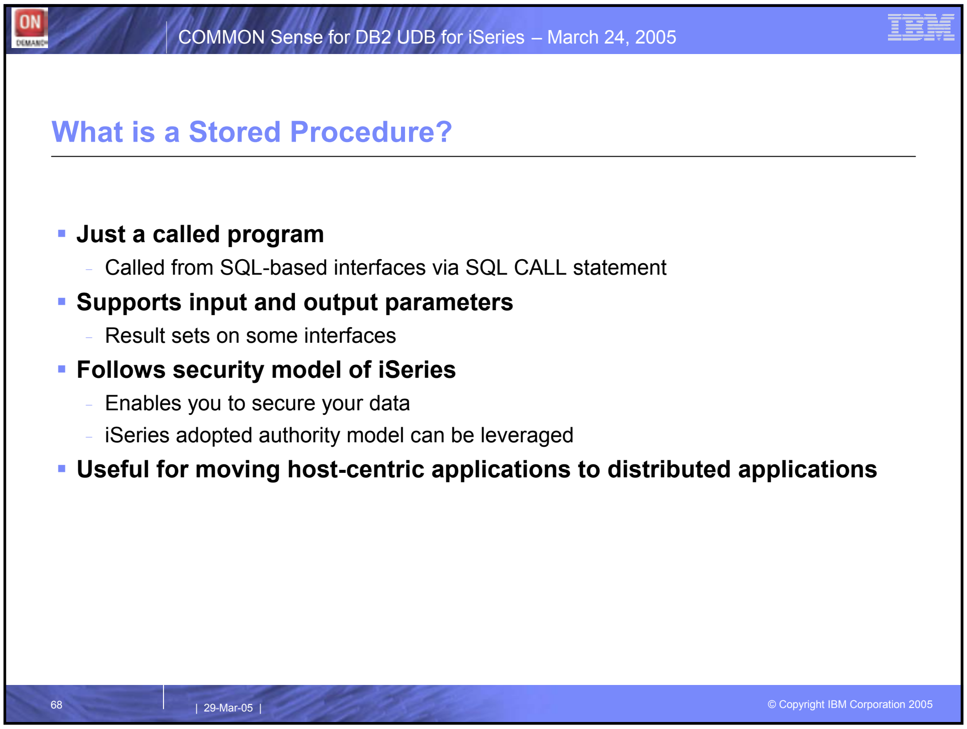


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Why Using Stored Procedures?

- **Isolate persistence model from process model**
- **Develop persistence model in a portable fashion:**
  - Supported across DB2 UDB family
  - Similar to procedure languages available from other DBMS (PL/SQL, T-SQL, etc)
- **DB2 UDB implementation not proprietary, follows SQL P(ersistent) S(tored) M(odules) Standard**
- **Makes it easier for SQL programmers to be productive faster on the iSeries**
- **Prerequisites on the development system**
  - Openness Includes (5722-SS1) required on development system
  - DB2 SQL Development Kit requirement eliminated in V5R2
  - C compiler requirement eliminated in V5R1

67 | 29-Mar-05 |

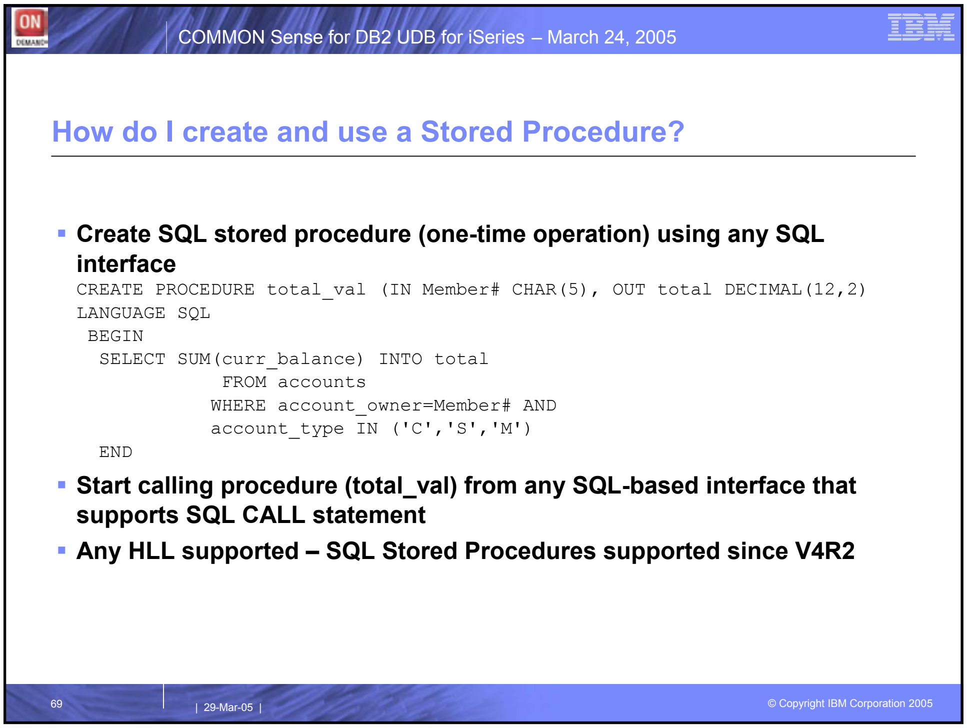


COMMON Sense for DB2 UDB for iSeries – March 24, 2005

What is a Stored Procedure?

- **Just a called program**
  - Called from SQL-based interfaces via SQL CALL statement
- **Supports input and output parameters**
  - Result sets on some interfaces
- **Follows security model of iSeries**
  - Enables you to secure your data
  - iSeries adopted authority model can be leveraged
- **Useful for moving host-centric applications to distributed applications**

68 | 29-Mar-05 |

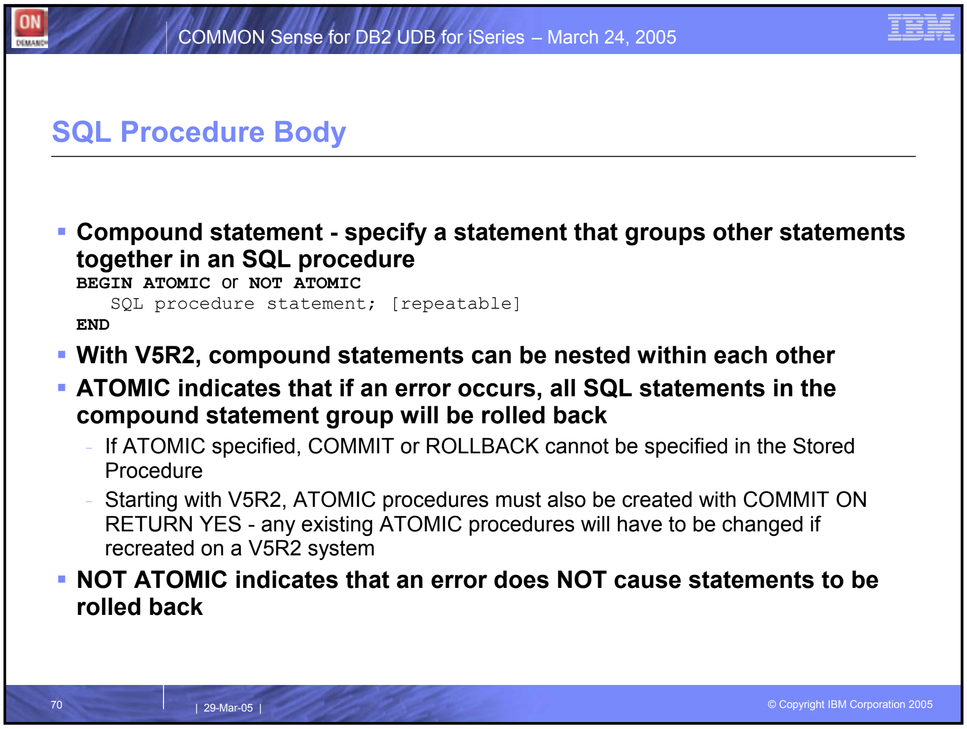


How do I create and use a Stored Procedure?

- **Create SQL stored procedure (one-time operation) using any SQL interface**

```
CREATE PROCEDURE total_val (IN Member# CHAR(5), OUT total DECIMAL(12,2)
LANGUAGE SQL
 BEGIN
  SELECT SUM(curr_balance) INTO total
             FROM accounts
            WHERE account_owner=Member# AND
            account_type IN ('C','S','M')
  END
```

- **Start calling procedure (total_val) from any SQL-based interface that supports SQL CALL statement**
- **Any HLL supported – SQL Stored Procedures supported since V4R2**

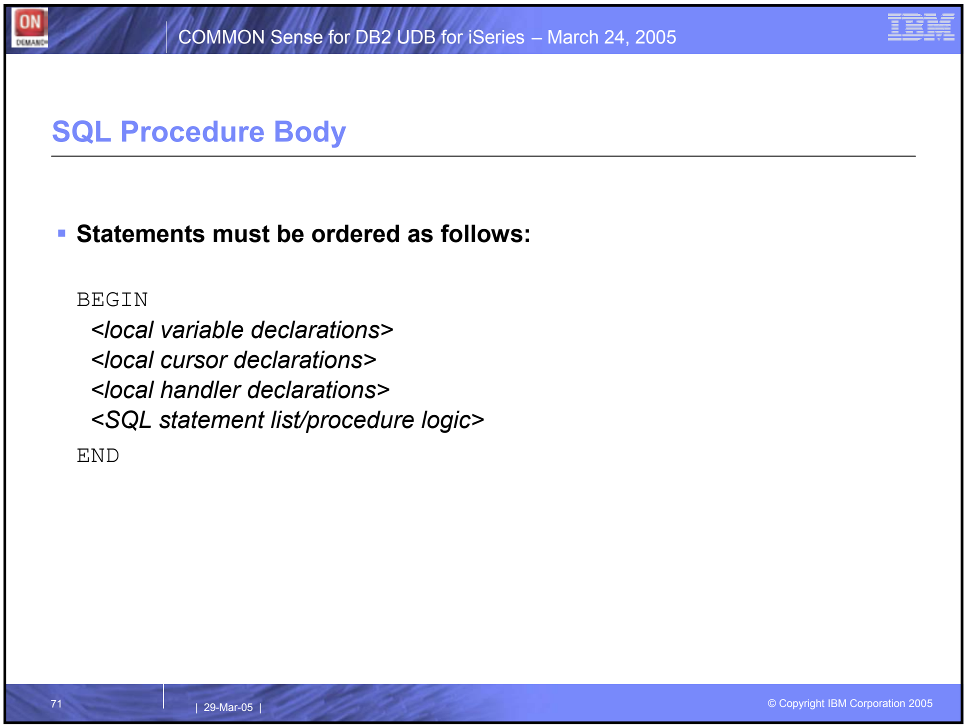


SQL Procedure Body

- **Compound statement - specify a statement that groups other statements together in an SQL procedure**

```
BEGIN ATOMIC or NOT ATOMIC
   SQL procedure statement; [repeatable]
END
```

- **With V5R2, compound statements can be nested within each other**
- **ATOMIC indicates that if an error occurs, all SQL statements in the compound statement group will be rolled back**
  - If ATOMIC specified, COMMIT or ROLLBACK cannot be specified in the Stored Procedure
  - Starting with V5R2, ATOMIC procedures must also be created with COMMIT ON RETURN YES - any existing ATOMIC procedures will have to be changed if recreated on a V5R2 system
- **NOT ATOMIC indicates that an error does NOT cause statements to be rolled back**



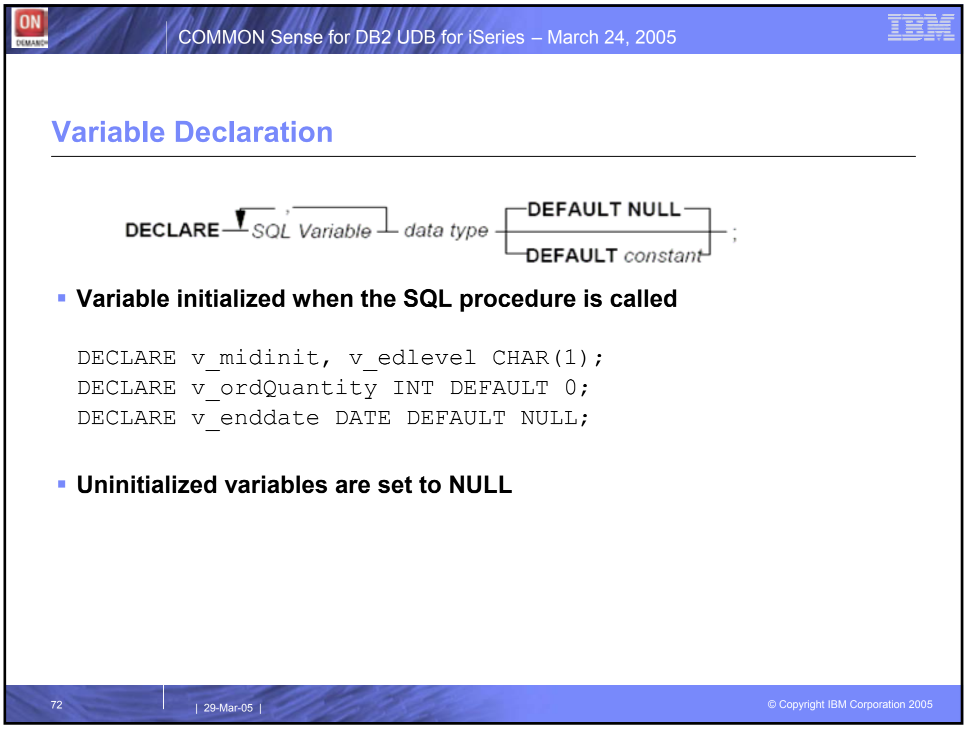
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

SQL Procedure Body

- **Statements must be ordered as follows:**

```
BEGIN
  <local variable declarations>
  <local cursor declarations>
  <local handler declarations>
  <SQL statement list/procedure logic>
END
```

71 | 29-Mar-05 |



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Variable Declaration

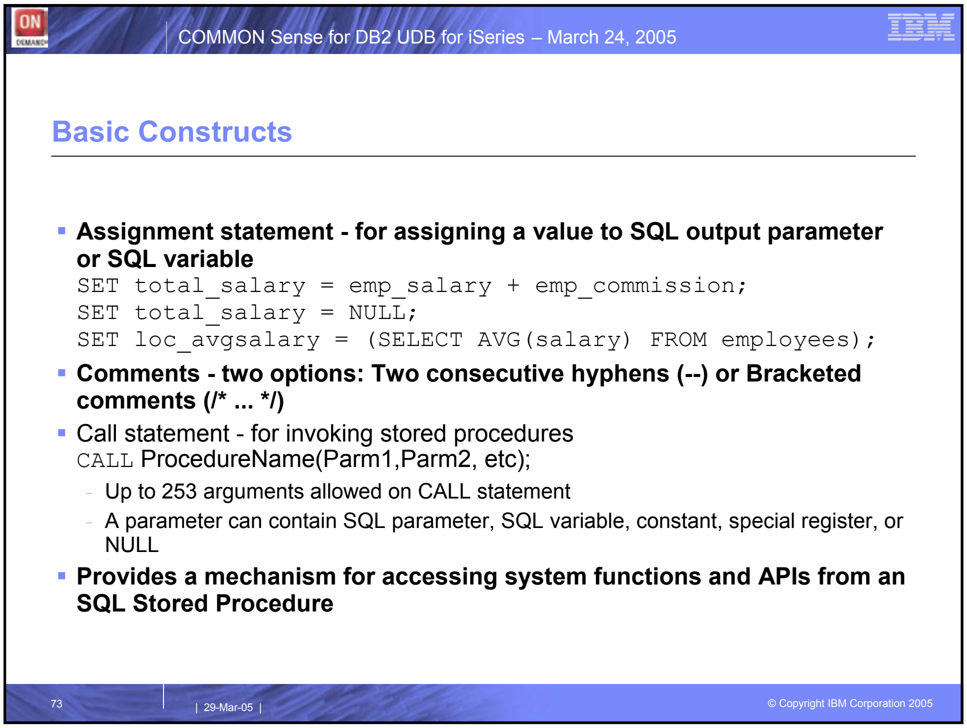
DECLARE ─ SQL Variable (,) ─ data type ─ DEFAULT NULL | DEFAULT *constant* ─ ;

- **Variable initialized when the SQL procedure is called**

```
DECLARE v_midinit, v_edlevel CHAR(1);
DECLARE v_ordQuantity INT DEFAULT 0;
DECLARE v_enddate DATE DEFAULT NULL;
```

- **Uninitialized variables are set to NULL**

72 | 29-Mar-05 |

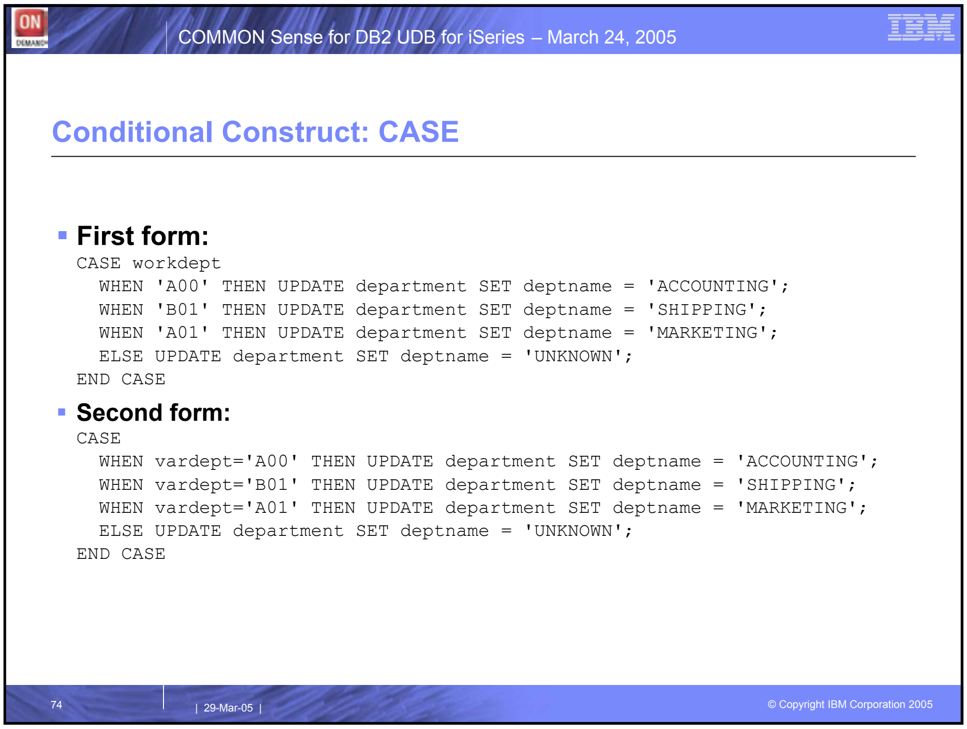


Basic Constructs

- **Assignment statement - for assigning a value to SQL output parameter or SQL variable**

```
SET total_salary = emp_salary + emp_commission;
SET total_salary = NULL;
SET loc_avgsalary = (SELECT AVG(salary) FROM employees);
```

- **Comments - two options: Two consecutive hyphens (--) or Bracketed comments (/* ... */)**
- Call statement - for invoking stored procedures
  CALL ProcedureName(Parm1,Parm2, etc);
  - Up to 253 arguments allowed on CALL statement
  - A parameter can contain SQL parameter, SQL variable, constant, special register, or NULL
- **Provides a mechanism for accessing system functions and APIs from an SQL Stored Procedure**



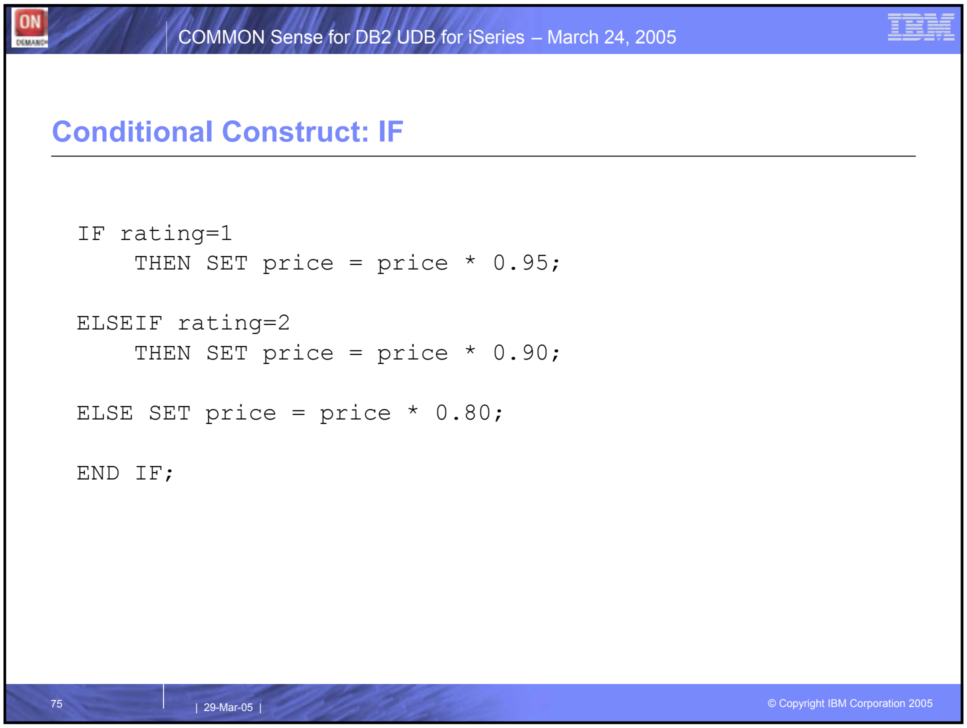
Conditional Construct: CASE

- **First form:**

```
CASE workdept
  WHEN 'A00' THEN UPDATE department SET deptname = 'ACCOUNTING';
  WHEN 'B01' THEN UPDATE department SET deptname = 'SHIPPING';
  WHEN 'A01' THEN UPDATE department SET deptname = 'MARKETING';
  ELSE UPDATE department SET deptname = 'UNKNOWN';
END CASE
```

- **Second form:**

```
CASE
  WHEN vardept='A00' THEN UPDATE department SET deptname = 'ACCOUNTING';
  WHEN vardept='B01' THEN UPDATE department SET deptname = 'SHIPPING';
  WHEN vardept='A01' THEN UPDATE department SET deptname = 'MARKETING';
  ELSE UPDATE department SET deptname = 'UNKNOWN';
END CASE
```



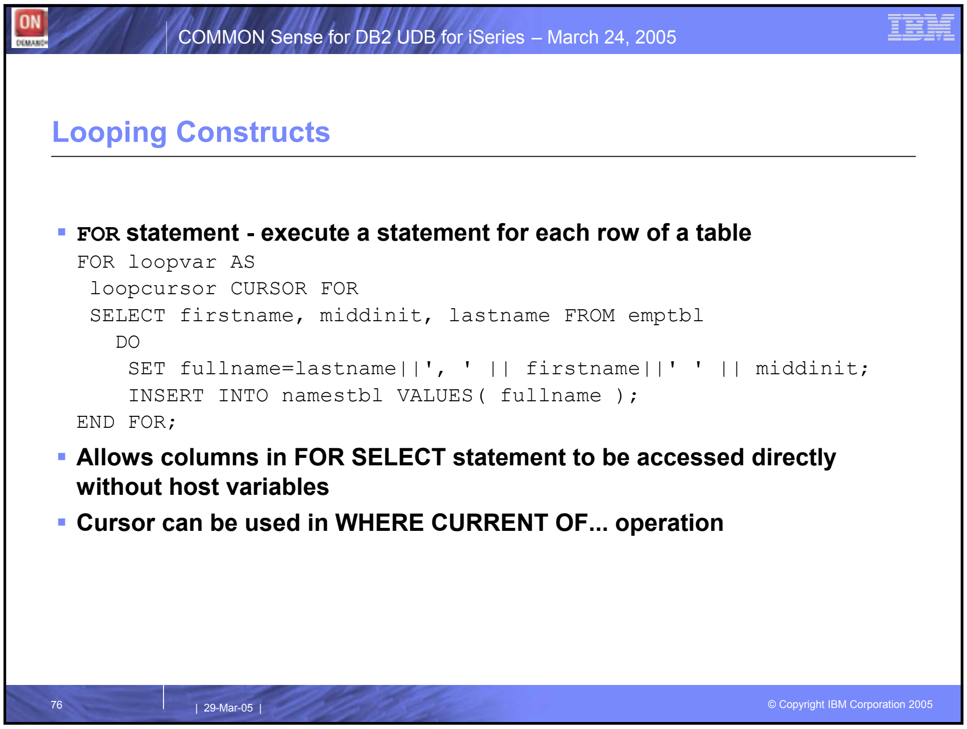
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Conditional Construct: IF

```
IF rating=1
    THEN SET price = price * 0.95;

ELSEIF rating=2
    THEN SET price = price * 0.90;

ELSE SET price = price * 0.80;

END IF;
```

75 | 29-Mar-05 |

© Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Looping Constructs

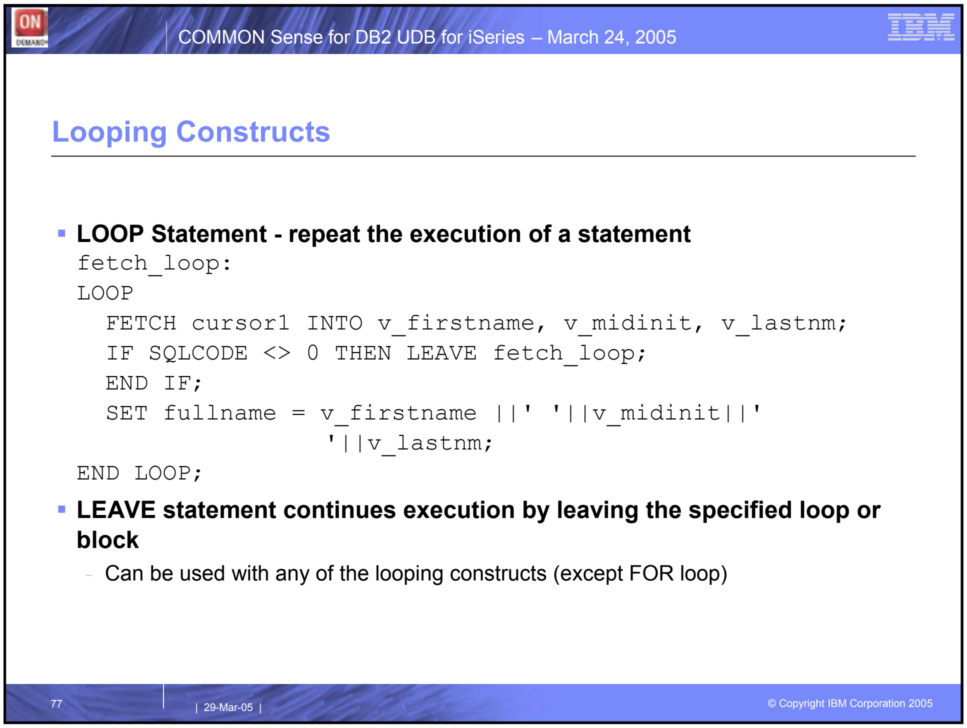
- **FOR statement - execute a statement for each row of a table**

```
FOR loopvar AS
 loopcursor CURSOR FOR
 SELECT firstname, middinit, lastname FROM emptbl
   DO
    SET fullname=lastname||', ' || firstname||' ' || middinit;
    INSERT INTO namestbl VALUES( fullname );
END FOR;
```

- **Allows columns in FOR SELECT statement to be accessed directly without host variables**
- **Cursor can be used in WHERE CURRENT OF... operation**

76 | 29-Mar-05 |

© Copyright IBM Corporation 2005



COMMON Sense for DB2 UDB for iSeries – March 24, 2005

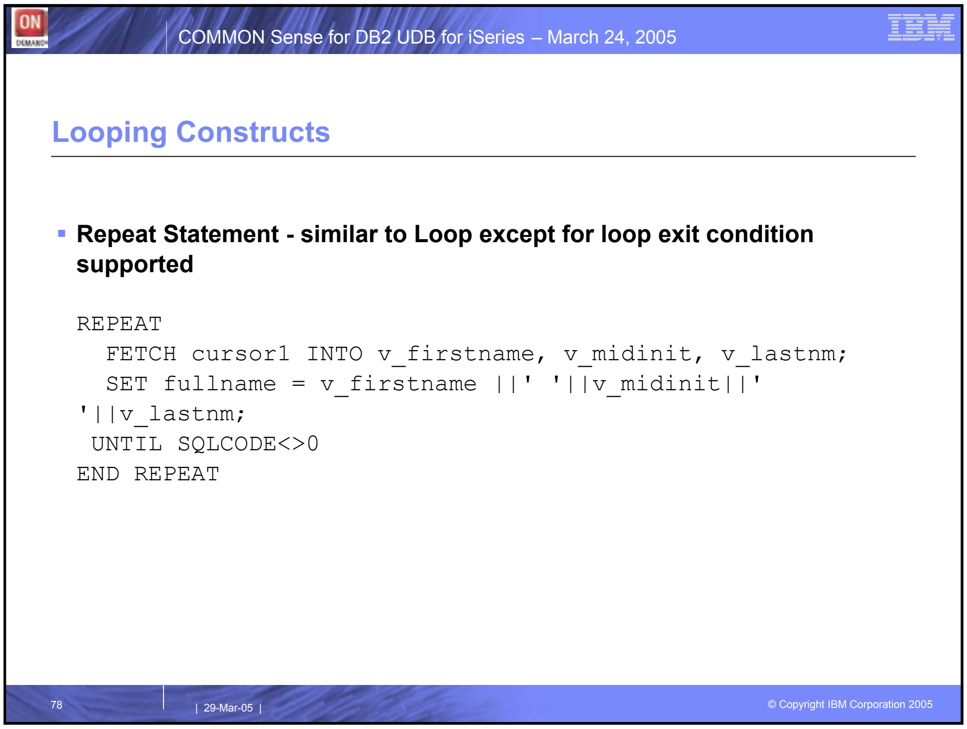
Looping Constructs

- **LOOP Statement - repeat the execution of a statement**

```
fetch_loop:
LOOP
  FETCH cursor1 INTO v_firstname, v_midinit, v_lastnm;
  IF SQLCODE <> 0 THEN LEAVE fetch_loop;
  END IF;
  SET fullname = v_firstname ||' '||v_midinit||'
                '||v_lastnm;
END LOOP;
```

- **LEAVE statement continues execution by leaving the specified loop or block**
  - Can be used with any of the looping constructs (except FOR loop)

77 | 29-Mar-05 | © Copyright IBM Corporation 2005



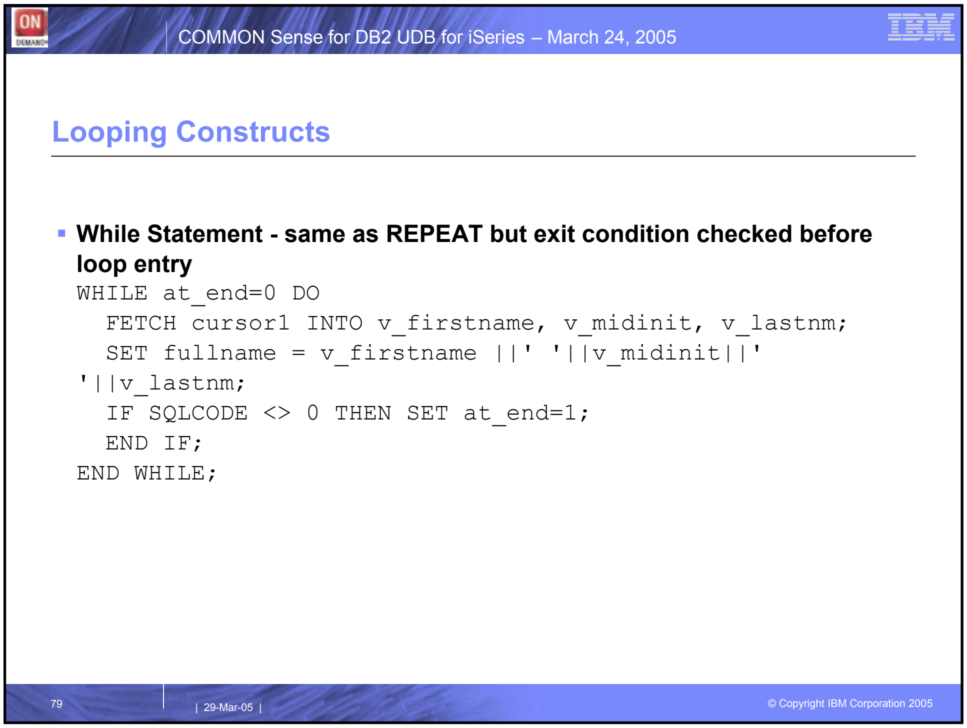
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Looping Constructs

- **Repeat Statement - similar to Loop except for loop exit condition supported**

```
REPEAT
  FETCH cursor1 INTO v_firstname, v_midinit, v_lastnm;
  SET fullname = v_firstname ||' '||v_midinit||'
'||v_lastnm;
 UNTIL SQLCODE<>0
END REPEAT
```

78 | 29-Mar-05 | © Copyright IBM Corporation 2005



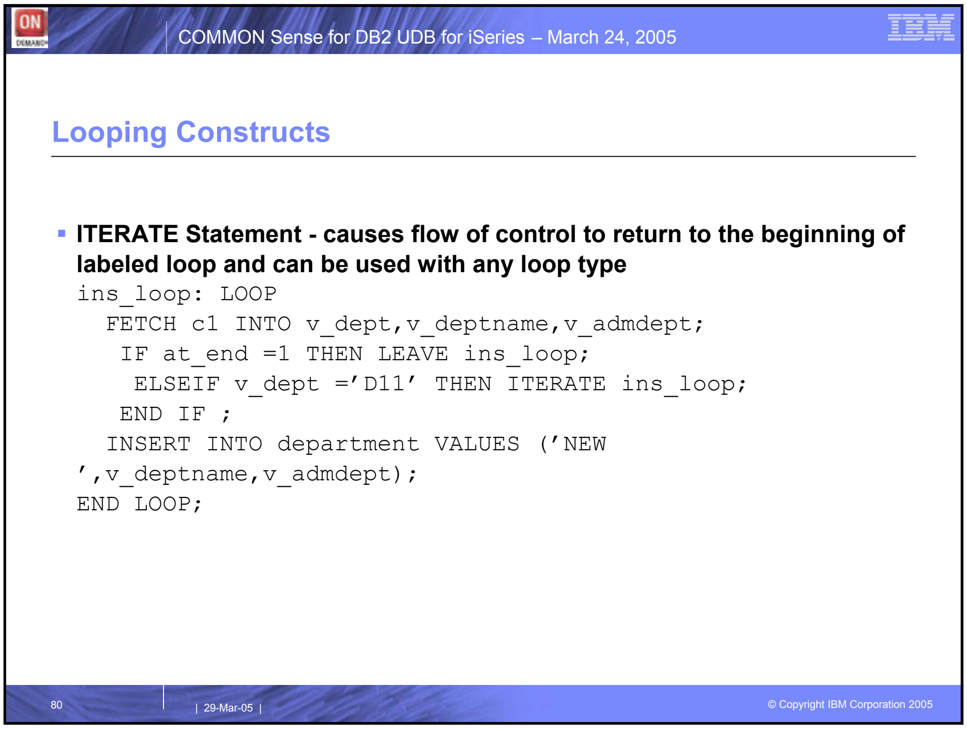
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Looping Constructs

- **While Statement - same as REPEAT but exit condition checked before loop entry**

```
WHILE at_end=0 DO
  FETCH cursor1 INTO v_firstname, v_midinit, v_lastnm;
  SET fullname = v_firstname ||' '||v_midinit||'
'||v_lastnm;
  IF SQLCODE <> 0 THEN SET at_end=1;
  END IF;
END WHILE;
```

79 | 29-Mar-05 |



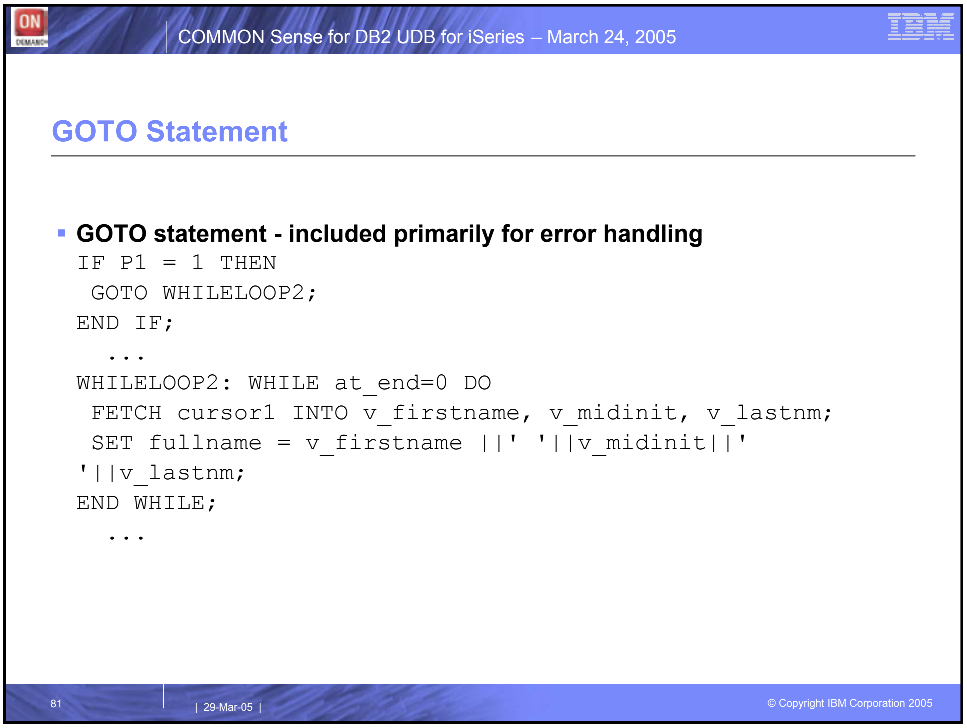
COMMON Sense for DB2 UDB for iSeries – March 24, 2005

Looping Constructs

- **ITERATE Statement - causes flow of control to return to the beginning of labeled loop and can be used with any loop type**

```
ins_loop: LOOP
  FETCH c1 INTO v_dept,v_deptname,v_admdept;
   IF at_end =1 THEN LEAVE ins_loop;
    ELSEIF v_dept =’D11’ THEN ITERATE ins_loop;
   END IF ;
  INSERT INTO department VALUES (’NEW
’,v_deptname,v_admdept);
END LOOP;
```

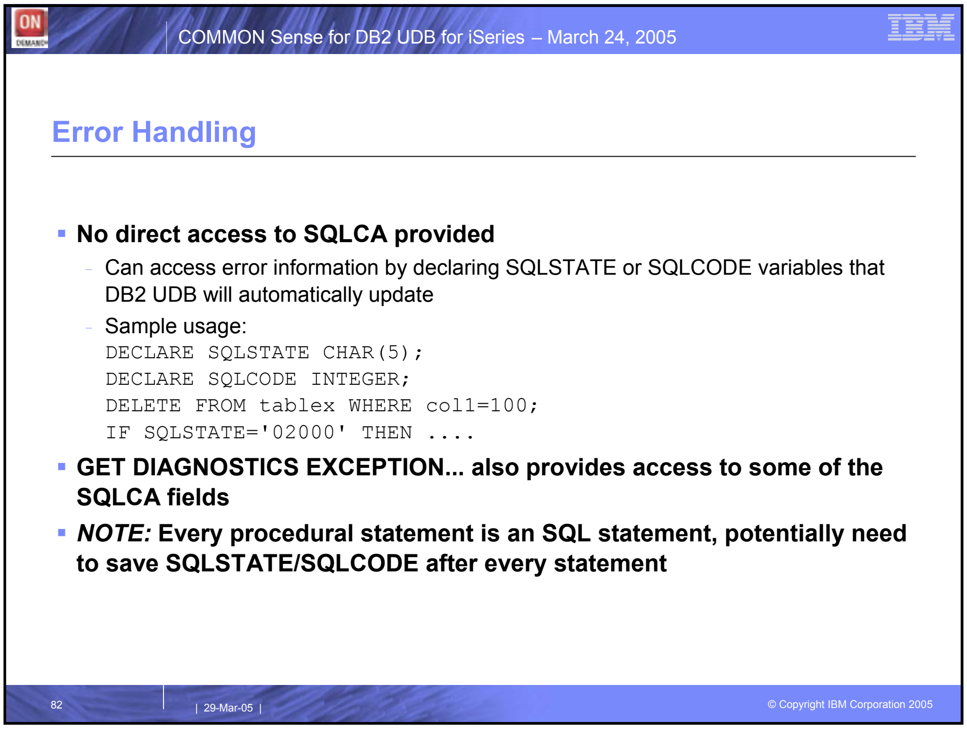
80 | 29-Mar-05 |



GOTO Statement

- **GOTO statement - included primarily for error handling**

```
IF P1 = 1 THEN
 GOTO WHILELOOP2;
END IF;
  ...
WHILELOOP2: WHILE at_end=0 DO
 FETCH cursor1 INTO v_firstname, v_midinit, v_lastnm;
 SET fullname = v_firstname ||' '||v_midinit||'
'||v_lastnm;
END WHILE;
  ...
```

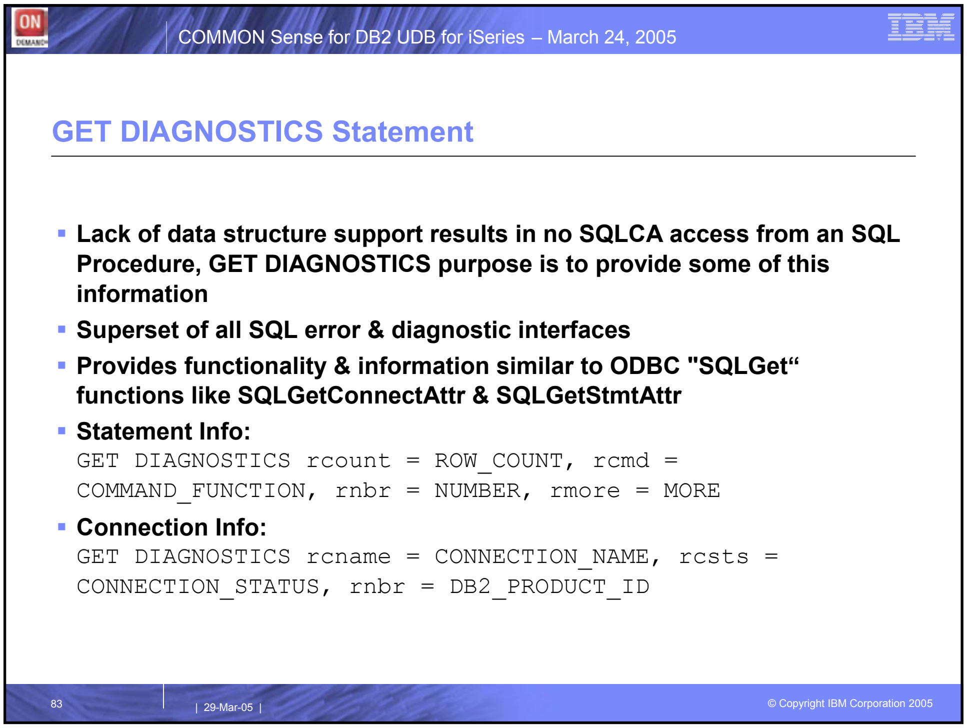


Error Handling

- **No direct access to SQLCA provided**
  - Can access error information by declaring SQLSTATE or SQLCODE variables that DB2 UDB will automatically update
  - Sample usage:

```
DECLARE SQLSTATE CHAR(5);
DECLARE SQLCODE INTEGER;
DELETE FROM tablex WHERE col1=100;
IF SQLSTATE='02000' THEN ....
```

- **GET DIAGNOSTICS EXCEPTION... also provides access to some of the SQLCA fields**
- ***NOTE:* Every procedural statement is an SQL statement, potentially need to save SQLSTATE/SQLCODE after every statement**



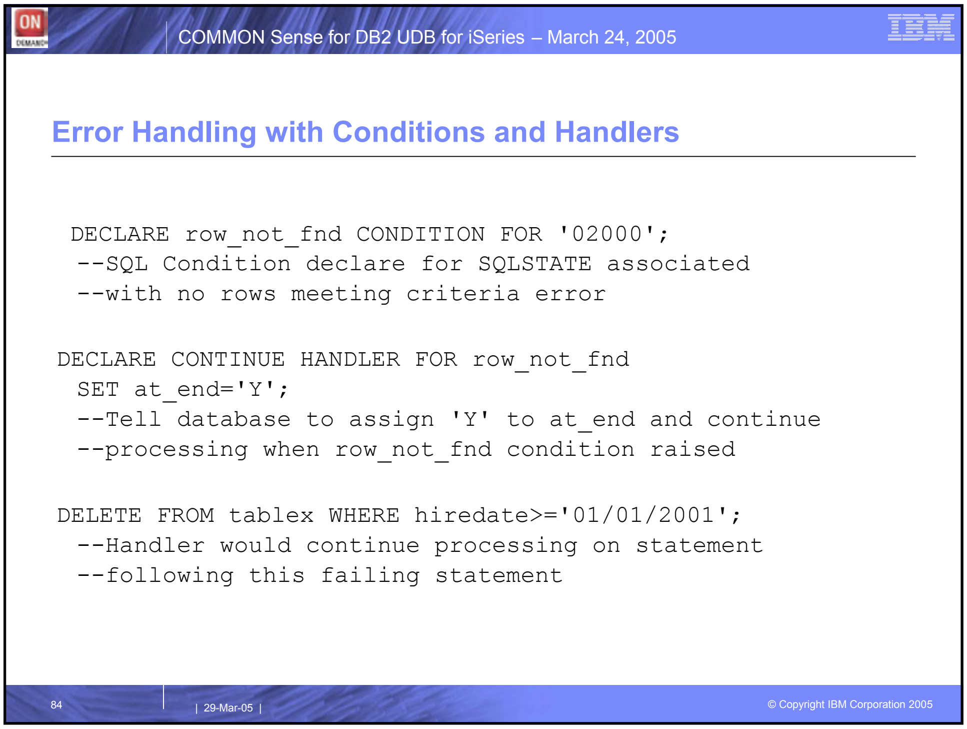
GET DIAGNOSTICS Statement

- **Lack of data structure support results in no SQLCA access from an SQL Procedure, GET DIAGNOSTICS purpose is to provide some of this information**
- **Superset of all SQL error & diagnostic interfaces**
- **Provides functionality & information similar to ODBC "SQLGet“ functions like SQLGetConnectAttr & SQLGetStmtAttr**
- **Statement Info:**

```
GET DIAGNOSTICS rcount = ROW_COUNT, rcmd =
COMMAND_FUNCTION, rnbr = NUMBER, rmore = MORE
```

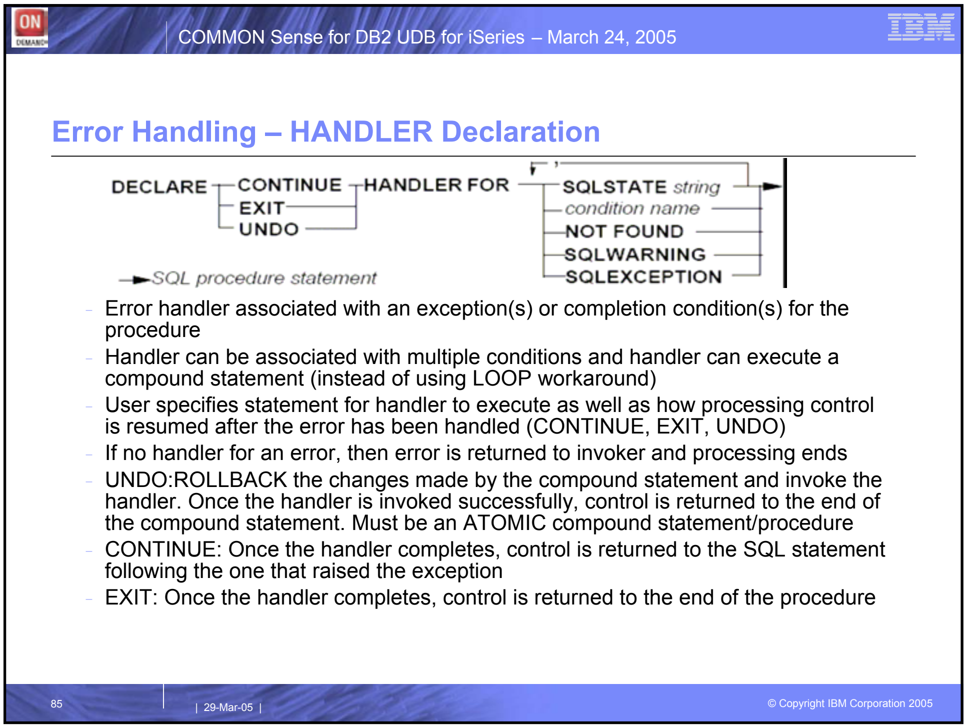
- **Connection Info:**

```
GET DIAGNOSTICS rcname = CONNECTION_NAME, rcsts =
CONNECTION_STATUS, rnbr = DB2_PRODUCT_ID
```



Error Handling with Conditions and Handlers

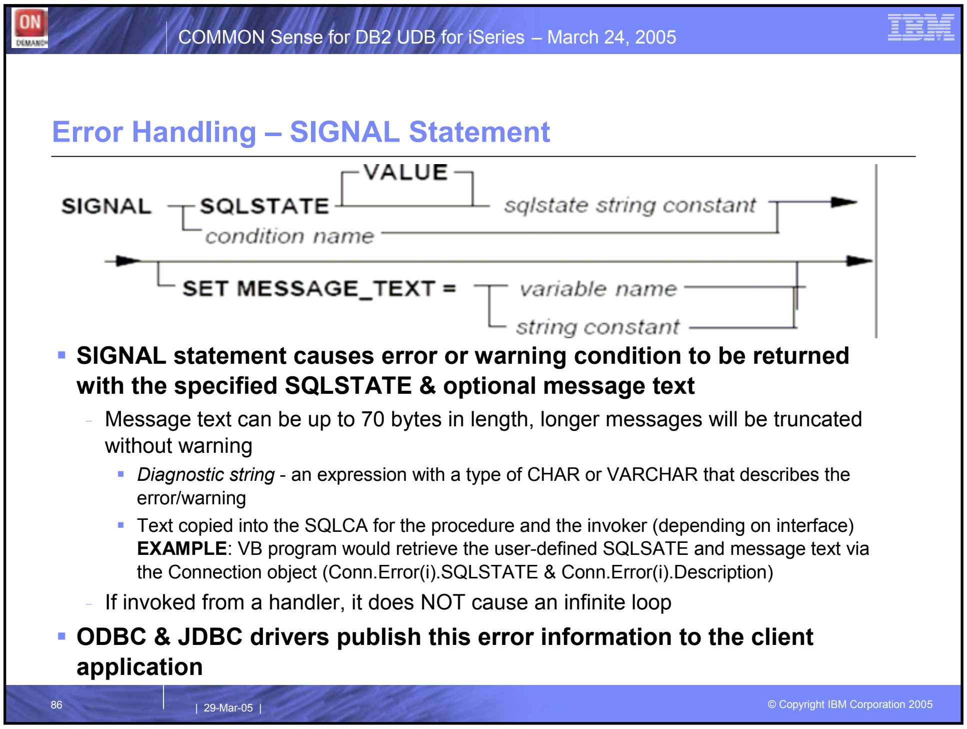
```
 DECLARE row_not_fnd CONDITION FOR '02000';
  --SQL Condition declare for SQLSTATE associated
  --with no rows meeting criteria error

DECLARE CONTINUE HANDLER FOR row_not_fnd
  SET at_end='Y';
  --Tell database to assign 'Y' to at_end and continue
  --processing when row_not_fnd condition raised

DELETE FROM tablex WHERE hiredate>='01/01/2001';
  --Handler would continue processing on statement
  --following this failing statement
```



Error Handling – HANDLER Declaration

```
DECLARE ┬ CONTINUE ┬ HANDLER FOR ──┬ SQLSTATE string ┬─►
        ├ EXIT ────┤               ├ condition name ─┤
        └ UNDO ────┘               ├ NOT FOUND ──────┤
                                   ├ SQLWARNING ─────┤
                                   └ SQLEXCEPTION ───┘
──►SQL procedure statement
```

- Error handler associated with an exception(s) or completion condition(s) for the procedure
- Handler can be associated with multiple conditions and handler can execute a compound statement (instead of using LOOP workaround)
- User specifies statement for handler to execute as well as how processing control is resumed after the error has been handled (CONTINUE, EXIT, UNDO)
- If no handler for an error, then error is returned to invoker and processing ends
- UNDO:ROLLBACK the changes made by the compound statement and invoke the handler. Once the handler is invoked successfully, control is returned to the end of the compound statement. Must be an ATOMIC compound statement/procedure
- CONTINUE: Once the handler completes, control is returned to the SQL statement following the one that raised the exception
- EXIT: Once the handler completes, control is returned to the end of the procedure



Error Handling – SIGNAL Statement

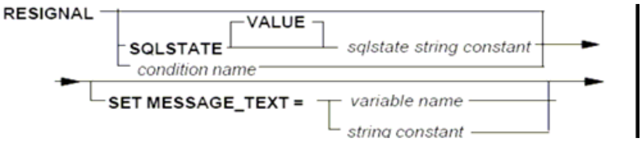
```
                    ┌ VALUE ┐
SIGNAL ┬ SQLSTATE ──┴───────┴── sqlstate string constant ┬──►
       └ condition name ─────────────────────────────────┘
►──┬─────────────────────────────────────────────────┬──►
   └ SET MESSAGE_TEXT = ┬ variable name ──┬──────────┘
                        └ string constant ┘
```

- **SIGNAL statement causes error or warning condition to be returned with the specified SQLSTATE & optional message text**
  - Message text can be up to 70 bytes in length, longer messages will be truncated without warning
    - *Diagnostic string* - an expression with a type of CHAR or VARCHAR that describes the error/warning
    - Text copied into the SQLCA for the procedure and the invoker (depending on interface) **EXAMPLE**: VB program would retrieve the user-defined SQLSATE and message text via the Connection object (Conn.Error(i).SQLSTATE & Conn.Error(i).Description)
  - If invoked from a handler, it does NOT cause an infinite loop
- **ODBC & JDBC drivers publish this error information to the client application**

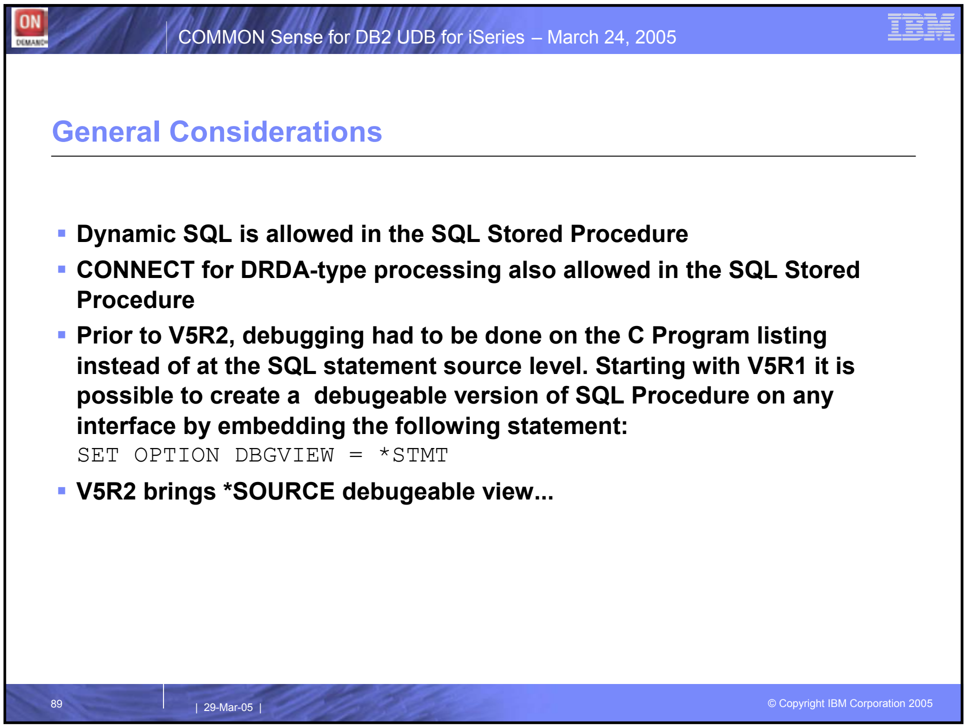
Error Handling – SIGNAL Statement

- **If SQLSTATE class is '01' or '02', warning is signaled and the SQLCODE is set to +438**
  - Otherwise SQLCODE set to -438
  - Negative SQLCODE cause output variables to NOT be returned
  - SystemMessage for SQLCODE 438 does NOT exist, just a code for program to program communications
- **If a handler for the signalled exception exists, exception is handled and control transferred to handler**
  - Otherwise, control returned immediately to the invoker for exceptions OR to the next statement for warning (01 & 02 class)
- **Recommendation: Define your own SQLSTATEs based on the ranges reserved for applications**

Error Handling – RESIGNAL Statement



- **Similar to SIGNAL, except that RESIGNAL can only be used in a handler to resignal an error or warning**
  - Don’t need to specify a State or Message just: RESIGNAL - In this case, the invoker is returned the original condition that caused the handler to be invoked
  - Have the option of supplying your own SQLSTATE or diagnostic message text
  - Example:
    Override system message for SQLSTATE 02000 "Row not found“ with "Specified part not found“
    Invoker would be able to access overwritten message in SQLCA

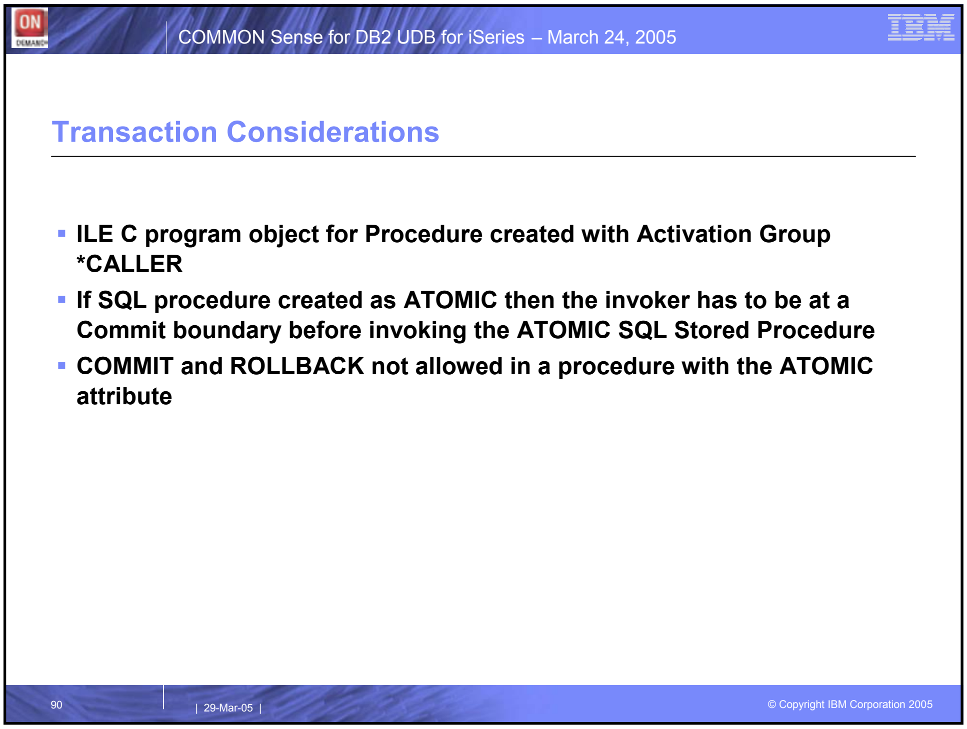


General Considerations

- Dynamic SQL is allowed in the SQL Stored Procedure
- CONNECT for DRDA-type processing also allowed in the SQL Stored Procedure
- Prior to V5R2, debugging had to be done on the C Program listing instead of at the SQL statement source level. Starting with V5R1 it is possible to create a debugeable version of SQL Procedure on any interface by embedding the following statement:

```
SET OPTION DBGVIEW = *STMT
```

- V5R2 brings *SOURCE debugeable view...



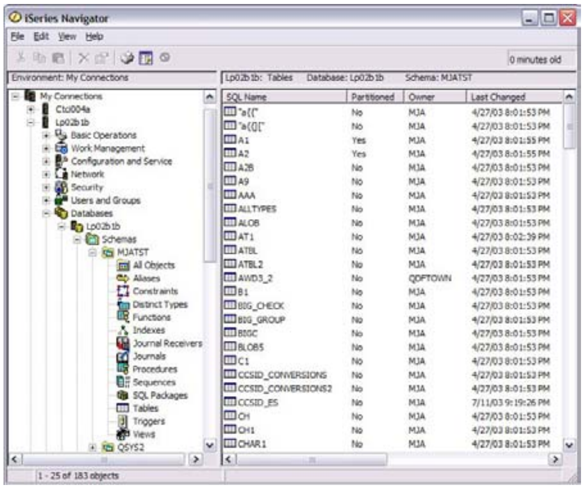
Transaction Considerations

- ILE C program object for Procedure created with Activation Group *CALLER
- If SQL procedure created as ATOMIC then the invoker has to be at a Commit boundary before invoking the ATOMIC SQL Stored Procedure
- COMMIT and ROLLBACK not allowed in a procedure with the ATOMIC attribute

iSeries DB2 Tools

iSeries Navigator

- DB2 Object Type Folders
- Show Related (graphical DSPDBR)
- Constraint Management Interface
- Reorganize Table Manager
- Index Analyzer
- SQE Aware Visual Explain
- Run SQL
- Multi-tasking Support

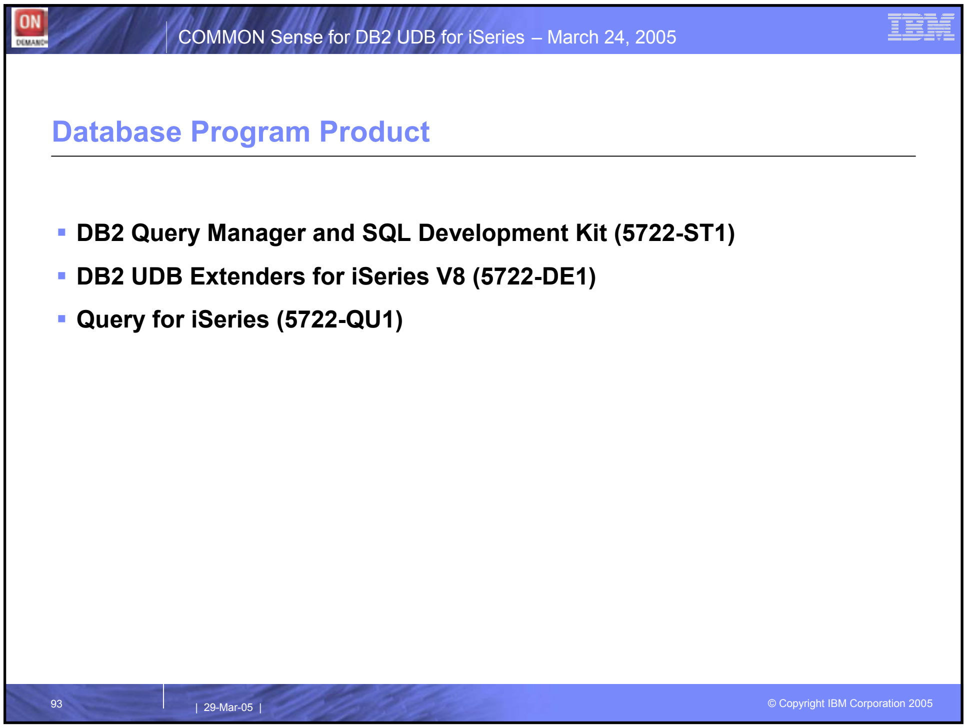


0 minutes old

Lp02b1b: Tables Database: Lp02b1b Schema: MJATST

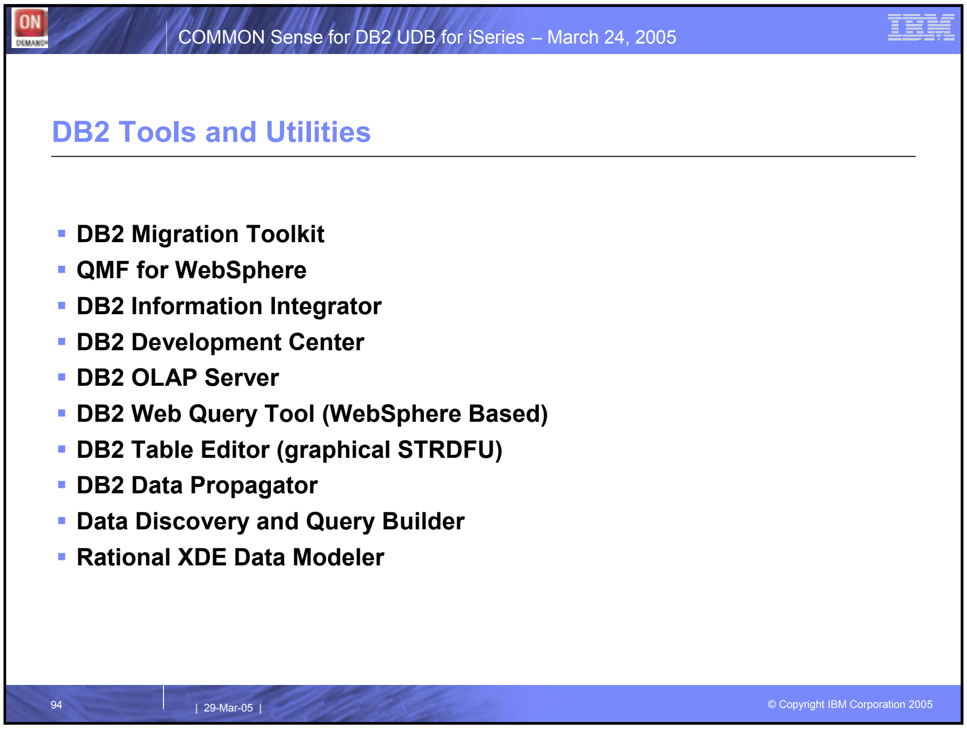
SQL Name	Partitioned	Owner	Last Changed
"a{{"	No	MJA	4/27/03 8:01:53 PM
"a{{|["	No	MJA	4/27/03 8:01:53 PM
A1	Yes	MJA	4/27/03 8:01:55 PM
A2	Yes	MJA	4/27/03 8:01:55 PM
A2B	No	MJA	4/27/03 8:01:53 PM
A9	No	MJA	4/27/03 8:01:53 PM
AAA	No	MJA	4/27/03 8:01:53 PM
ALLTYPES	No	MJA	4/27/03 8:01:53 PM
ALOB	No	MJA	4/27/03 8:01:53 PM
AT1	No	MJA	4/27/03 8:02:39 PM
ATBL	No	MJA	4/27/03 8:01:53 PM
ATBL2	No	MJA	4/27/03 8:01:53 PM
AWD3_2	No	QDFTOWN	4/27/03 8:01:53 PM
B1	No	MJA	4/27/03 8:01:53 PM
BIG_CHECK	No	MJA	4/27/03 8:01:53 PM
BIG_GROUP	No	MJA	4/27/03 8:01:53 PM
BIGC	No	MJA	4/27/03 8:01:53 PM
BLOBS	No	MJA	4/27/03 8:01:53 PM
C1	No	MJA	4/27/03 8:01:53 PM
CCSID_CONVERSIONS	No	MJA	4/27/03 8:01:53 PM
CCSID_CONVERSIONS2	No	MJA	4/27/03 8:01:53 PM
CCSID_ES	No	MJA	7/11/03 9:19:26 PM
CH	No	MJA	4/27/03 8:01:53 PM
CH1	No	MJA	4/27/03 8:01:53 PM
CHAR1	No	MJA	4/27/03 8:01:53 PM

1 - 25 of 183 objects



Database Program Product

- DB2 Query Manager and SQL Development Kit (5722-ST1)
- DB2 UDB Extenders for iSeries V8 (5722-DE1)
- Query for iSeries (5722-QU1)



DB2 Tools and Utilities

- DB2 Migration Toolkit
- QMF for WebSphere
- DB2 Information Integrator
- DB2 Development Center
- DB2 OLAP Server
- DB2 Web Query Tool (WebSphere Based)
- DB2 Table Editor (graphical STRDFU)
- DB2 Data Propagator
- Data Discovery and Query Builder
- Rational XDE Data Modeler