IBM eServer iSeries
Initiative for Tools Innovation

Centerfield TECHNOLOGY
strategic tools for strategic teams applications

insure/SQL

IBM eServer iSeries

Indexing Strategies for DB2 UDB for iSeries

Mike Cain
DB2 UDB for iSeries Center of Competency
Rochester, MN USA

why “i”? it’s simple.

Scenario

Find the first occurrence of "**IBM**" in a very large book…

What do you do first?

Turn to the index!

in·dex Something that serves to guide, point out, or otherwise facilitate reference.

Creating a useful index

is both a Science and an Art.

Indexing Technology within DB2 UDB for iSeries

DB2 UDB for iSeries

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 or data

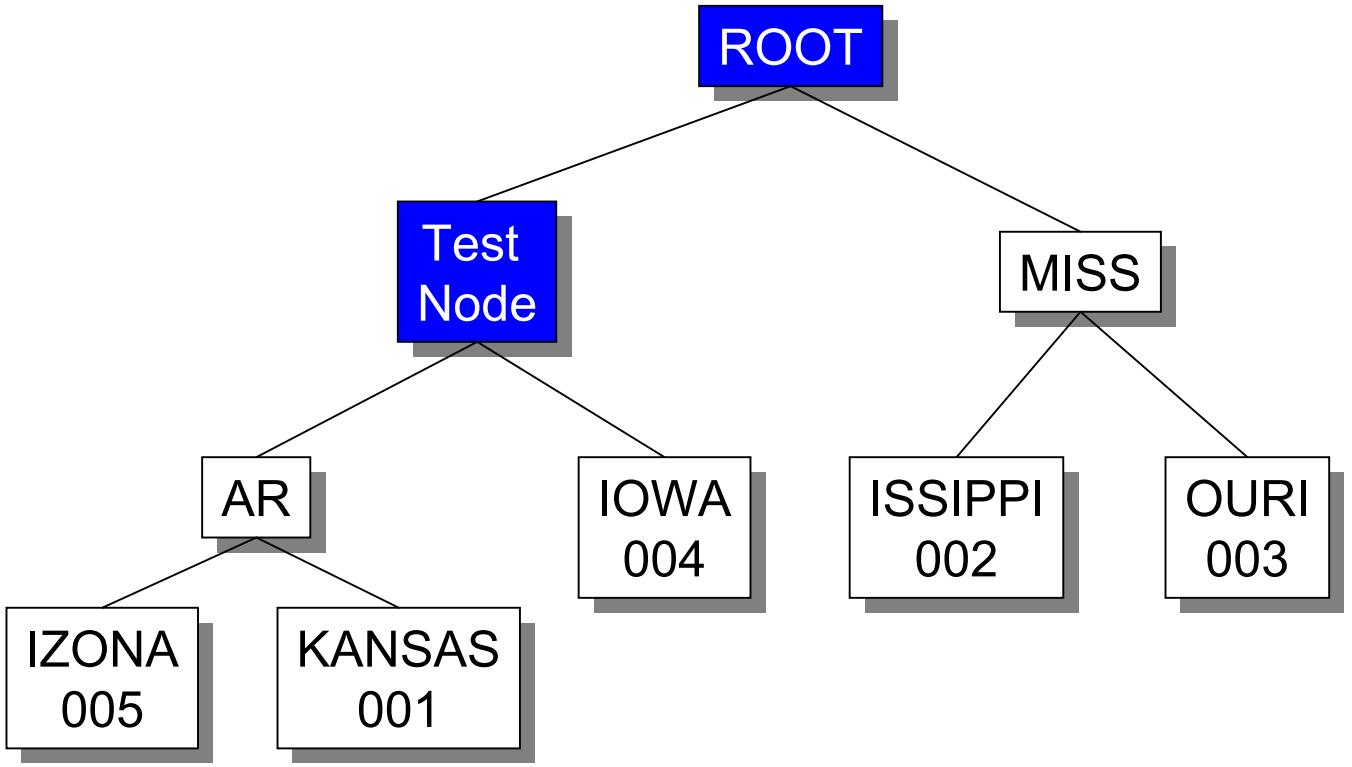
Indexes are scanned or probed

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

Radix Index

Database Table	
001	ARKANSAS
002	MISSIPPI
003	MISSOURI
004	IOWA
005	ARIZONA
…	…



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

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

Encoded Vector Index (EVI)

- Index for delivering fast data access in decision support and query reporting environments
  - Advanced technology from IBM Research
  - Variation on bitmap indexing
  - Fast access to statistics improve query optimizer decision making
- Not a “tree” structure
- Can only be created through an SQL interface or iSeries Navigator

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

Encoded Vector Index (EVI)

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

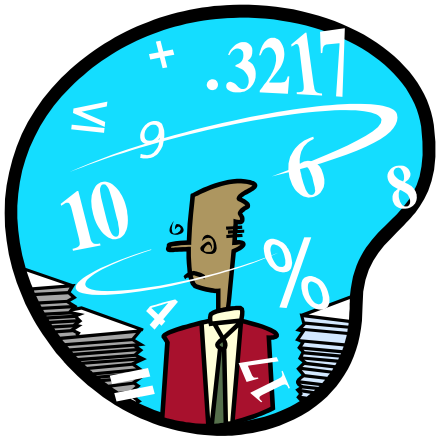
Vector	RRN
1	1
17	2
18	3
9	4
2	5
7	6
38	7
38	8
1	9
…	…

- Symbol table contains information for each distinct key value
  - Each key value is assigned a unique code (key compression)
  - Code is 1, 2, or 4 bytes depending on number of distinct key values
- Rather then a bit array for each distinct key value, the use one array of codes

DB2 UDB for iSeries

cardinality The number of elements in a set.

- High cardinality = large distinct number of values
- Low cardinality = small distinct number of values



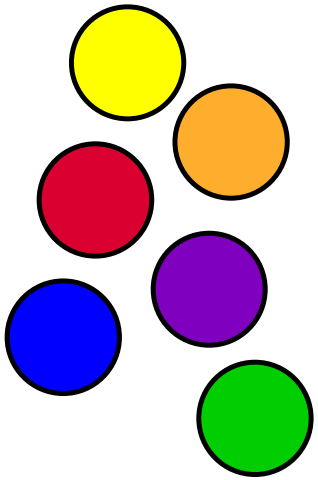
In general…

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

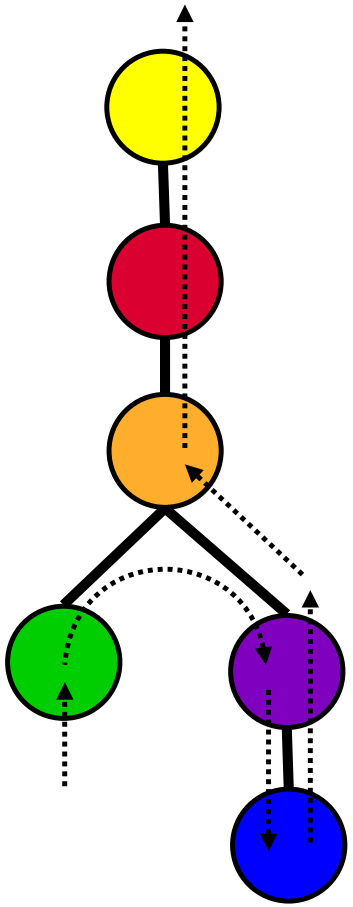
Query Optimization
(using indexes)

Query Graphs and Flows

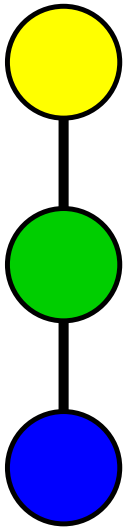
Set of methods



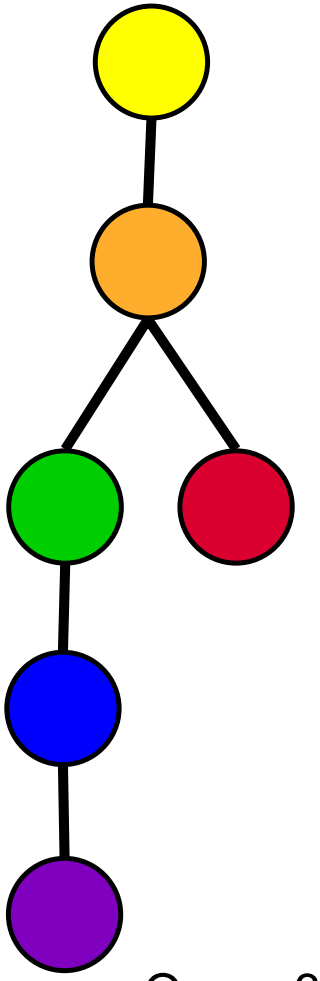
Assembled into query "graphs"



Query 1



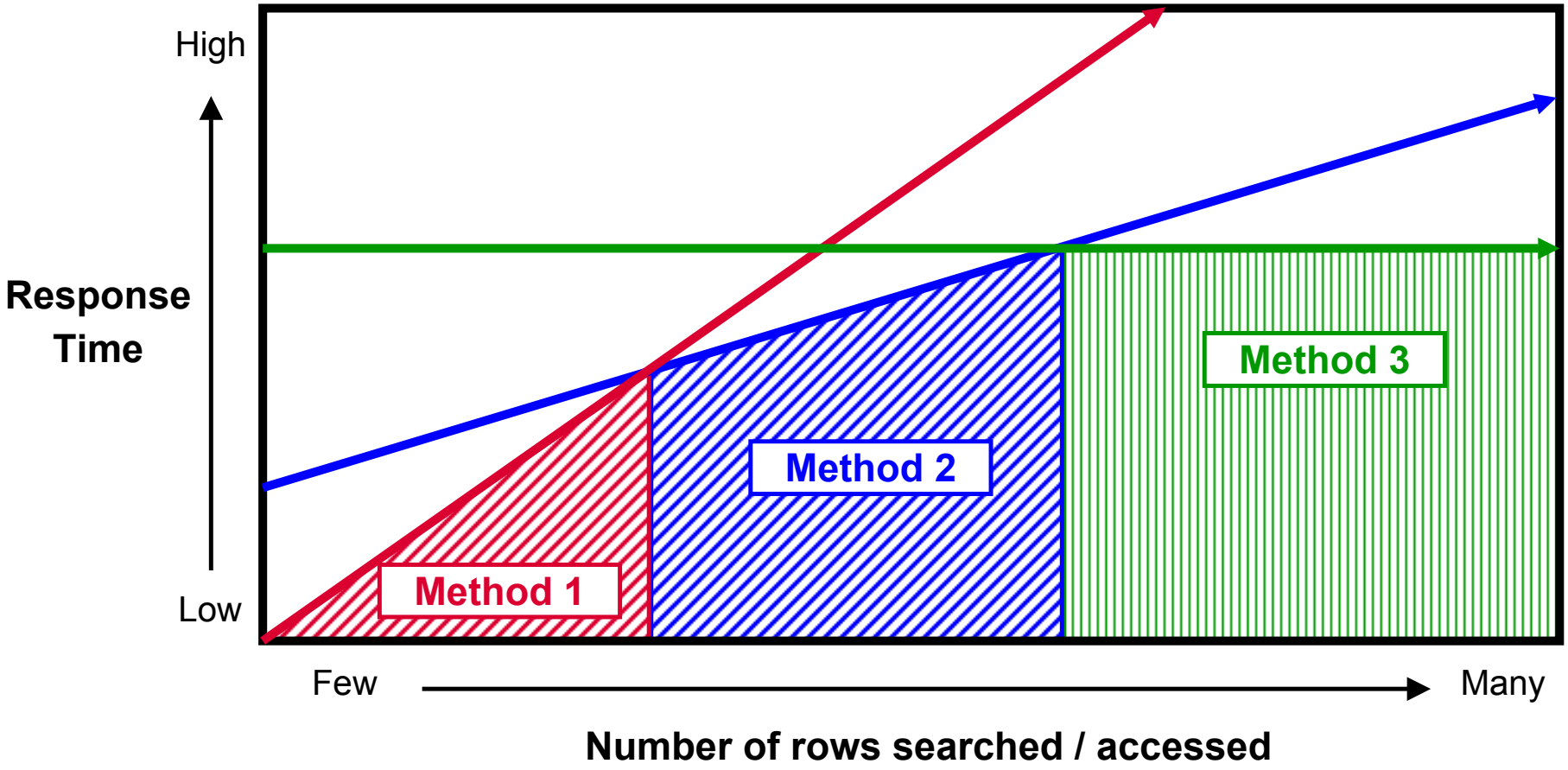
Query 2



Query 3

Data Access Methods

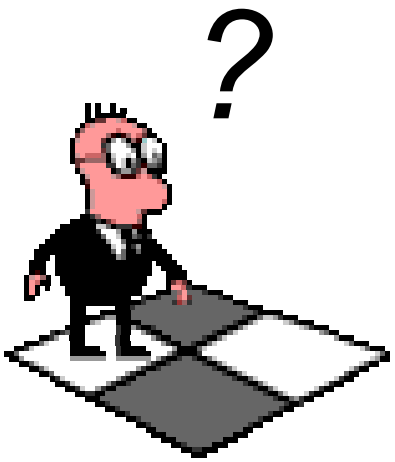
Cost based optimization dictates that the fastest access method for a given table will vary based upon selectivity of the query



Strategy for Query Optimization

Query optimization will generally follow this simplified strategy:

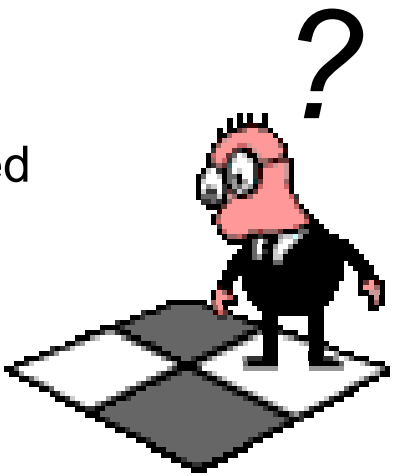
- Gather meta-data and statistics for costing
  - Selectivity statistics
  - Indexes available to be costed
    - Sort the indexes based upon their usefulness
  - Environmental attributes that may affect the costs
- Generate default cost
  - Build an access plan associated with the default plan
- For each index:
  - Gather information needed specific to this index
  - Build an access plan based on this index
  - Cost the use of the index with this access plan
  - Compare the resulting cost against the cost from the current best plan



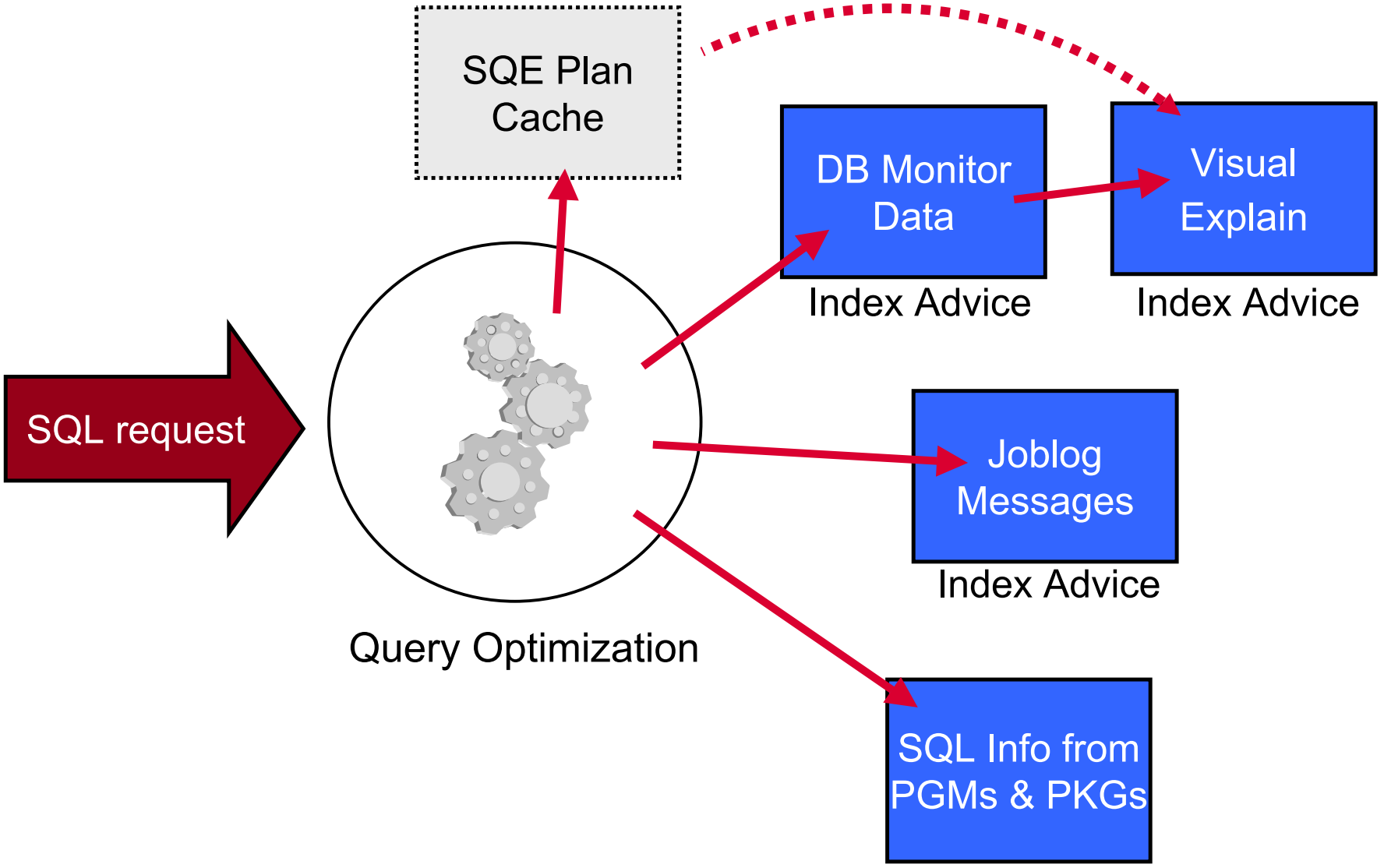
Strategy for Query Optimization

Optimizing indexes will generally follow this simplified strategy:

- Gather list of indexes for statistics and costing
- Sort the list of indexes considering how the index can be used
  - Local selection
  - Joining
  - Grouping
  - Ordering
  - Index only access
- One index may be useful for statistics, and another useful for implementation



Query Optimization Feedback



Indexing Strategies

DB2 UDB for iSeries

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

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

Understand the data being queried

- Column selectivity
- Column cardinality

Separating complex queries into individual parts by table

- Selecting
- Joining
- Grouping
- Ordering
- Subquery
- View

Indexing Strategy - Basic Approach

Radix Indexes

- Local selection columns
- Join columns

} Minimum

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

Encoded Vector Indexes

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

Indexing Strategy - Examples

```
-- Query 1
SELECT      A.CUSTOMER_NO, A.ORDER_DATE, A.QUANTITY
FROM        ORDERS A
WHERE       A.CUSTOMER_NO = 0112358;

CREATE INDEX ORDERS_IX1 ON ORDERS (CUSTOMER_NO);


-- Query 2
SELECT      A.CUSTOMER_NO, A.ORDER_DATE, A.QUANTITY
FROM        ORDERS A
WHERE       A.CUSTOMER_NO = 0112358
AND         A.ITEM_ID = 'ABC123YXZ';

CREATE INDEX ORDERS_IX2 ON ORDERS (CUSTOMER_NO, ITEM_ID);
```

Indexing Strategy - Examples

```
-- Query 3
SELECT      A.CUSTOMER_NO, A.CUSTOMER, A.ORDER_DATE
FROM        ORDERS A
WHERE       A.CUSTOMER_NO IN (0112358, 1321345, 5891442)
AND         A.ORDER_DATE > '2005/06/30'
ORDER BY    A.ORDER_DATE;

CREATE INDEX ORDERS_IX3a ON ORDERS (CUSTOMER_NO, ORDER_DATE);
CREATE INDEX ORDERS_IX3b ON ORDERS (ORDER_DATE, CUSTOMER_NO);

-- Query 4
SELECT      A.CUSTOMER_NO, A.CUSTOMER, A.ORDER_DATE
FROM        ORDERS A
WHERE       A.CUSTOMER_NO = 0112358
OR          A.ORDER_DATE = '2005/06/30';

CREATE INDEX ORDERS_IX4 ON ORDERS (CUSTOMER_NO);
CREATE ENCODED VECTOR INDEX ORDERS_EVI4
        ON ORDERS (ORDER_DATE);
```

Indexing Strategy - Examples

```
-- Query 5
SELECT      A.CUSTOMER_NO, B.CUSTOMER, A.ORDER_DATE, A.QUANTITY
FROM        ORDERS A,
            CUSTOMERS B,
            ITEMS C
WHERE       A.CUSTKEY = B.CUSTKEY
AND         A.ITEMKEY = C.ITEMKEY
AND         A.CUSTOMER_NO = 0112358;

CREATE INDEX ORDERS_IX5a ON ORDERS (CUSTOMER_NO, CUSTKEY);
CREATE INDEX ORDERS_IX5b ON ORDERS (CUSTOMER_NO, ITEMKEY);
CREATE INDEX CUSTOMERS_IX5 ON CUSTOMERS (CUSTKEY);
CREATE INDEX ITEMS_IX5 ON ITEMS (ITEMKEY);
```

Indexing Strategy - Examples

If the optimizer feedback indicates:

Full table scan → Create an index on local selection columns

Temporary index → Create an index on join columns
→ Create an index on grouping columns
→ Create an index on ordering columns

Hash table → Create an index on join columns
→ Create an index on grouping columns

“Perfect”, multiple key column radix indexes are usually best

More information and examples at:
ibm.com/servers/enable/site/education/abstracts/indxng_abs.html

Looking into the Future…

Centerfield TECHNOLOGY

Thank You

WANT MORE INFORMATION?

Centerfield Technology – Rochester, Minnesota

http:www.centerfieldtechnology.com

IBM eServer iSeries Initiative for Tools Innovation

http://www.developer.ibm.com/vic/hardware/portal/iii_pages/iii_tools_innov_index

Indexing Strategies for DB2 UDB on iSeries

http://www-03.ibm.com/servers/enable/site/education/abstracts/indxng_abs.html

IBM

IBM eServer iSeries Initiative for Tools Innovation

insure/SQL

Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AS/400	e-business on demand	OS/400
AS/400e	IBM	i5/OS
eServer	IBM (logo)	
	iSeries	

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.
Other company, product or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.