IBM Systems and Technology Group

DB2 for i5/OS Temporary Indexes…

The Good, The Bad, and The Ugly

Mike Cain
DB2 for i5/OS Center of Competency
Rochester, MN USA

Creating a temporary index

is Good, Bad and Ugly.

The Good…

DB2 will create an index if one needed.

The Bad…

You wait while the index is being built.

The Ugly…

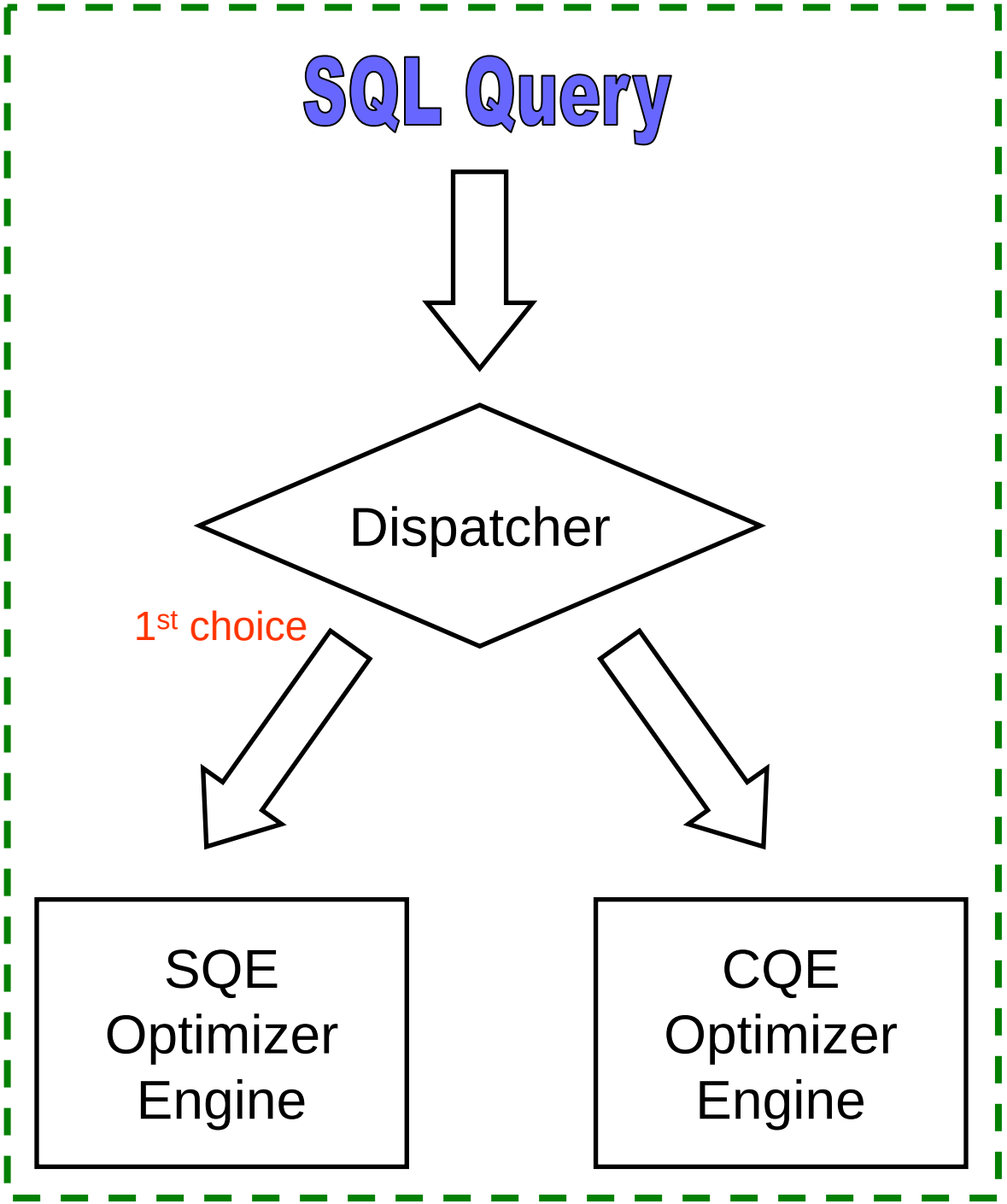
Building indexes might take a long time and use much resource.

Temporary indexes have limitations.

Optimizers and Engines within DB2 for i5/OS

a Review…

DB2 for i5/OS



New SQL Query Engine
- Phased in over time
- SQL only
- State of the Art
- Robust
- Powerful

Original Query Engine
- Phased out over time
- non SQL queries

Indexing Technology within DB2 for i5/OS

a Review…

DB2 for i5/OS

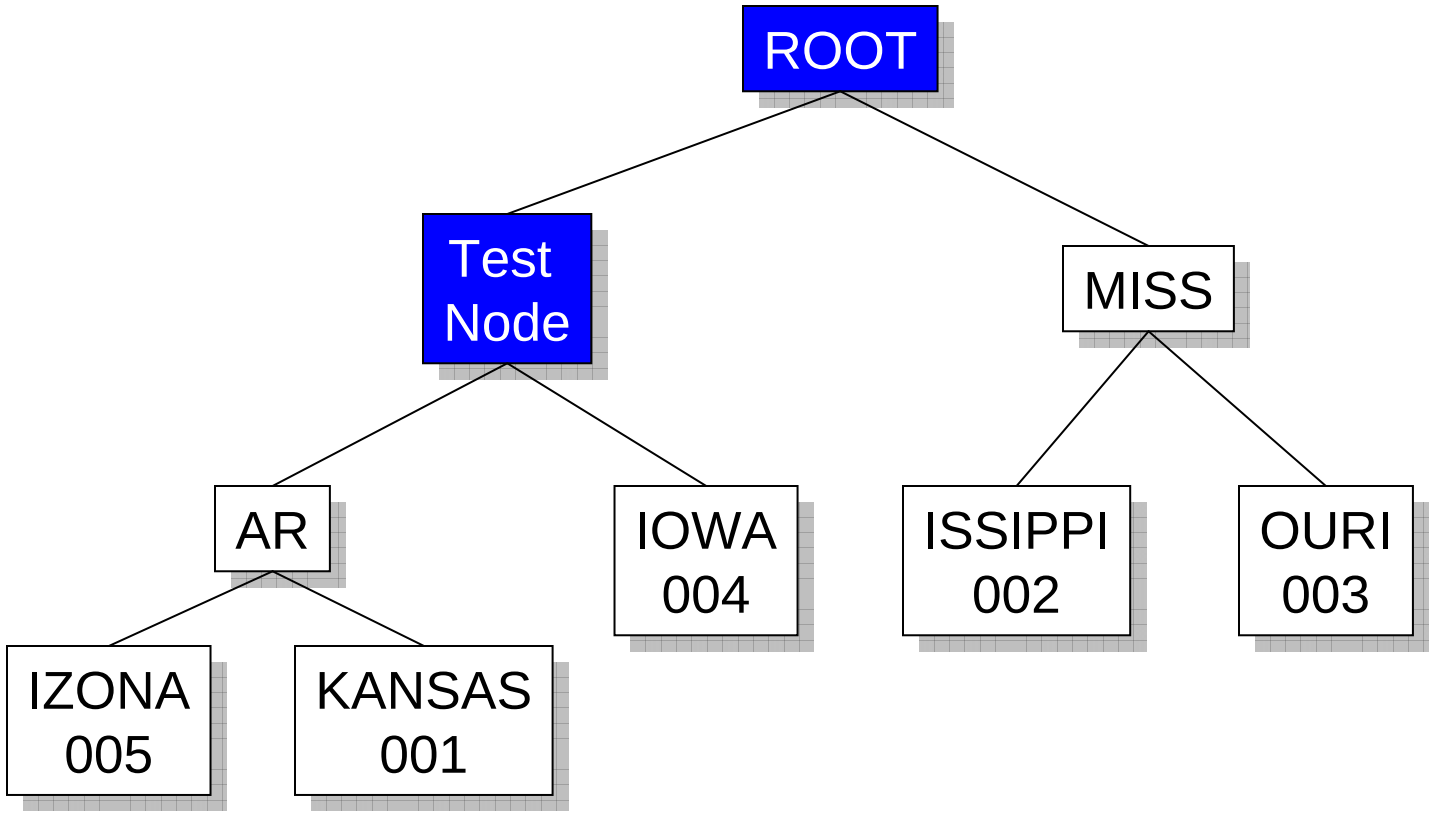
- Two types of indexing technologies are supported
  - ***Radix*** Index
  - ***Encoded Vector*** Index
- Each type of index has specific uses and advantages
- Respective indexing technologies compliment each other
- Indexes can be used for statistics and implementation
- Indexes can provide RRNs and/or 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

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

Radix Index

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



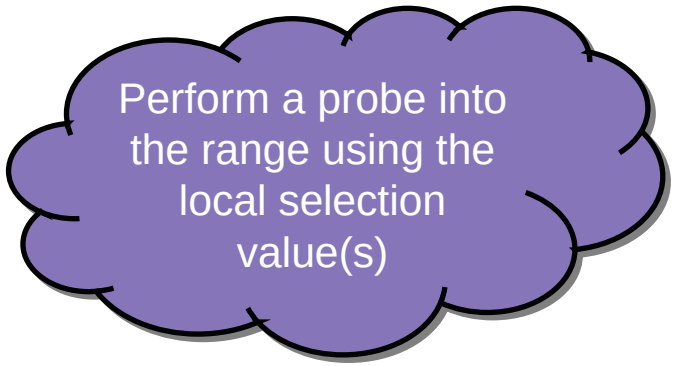
ADVANTAGES:
- Very fast access to a single key value
- Also fast for small, selected range of key values
- Provides order

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

Index Probe Example

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

```
SELECT *
   FROM CUSTOMER
   WHERE STATE = 'IOWA'
```



CUSTOMER Index

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

RRN

CUSTOMER 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

Encoded Vector Index (EVI)

- 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 System i Navigator GUI

```
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
…				
Wisconsin	49	7	30111	340
Wyoming	50	252	83000	2760

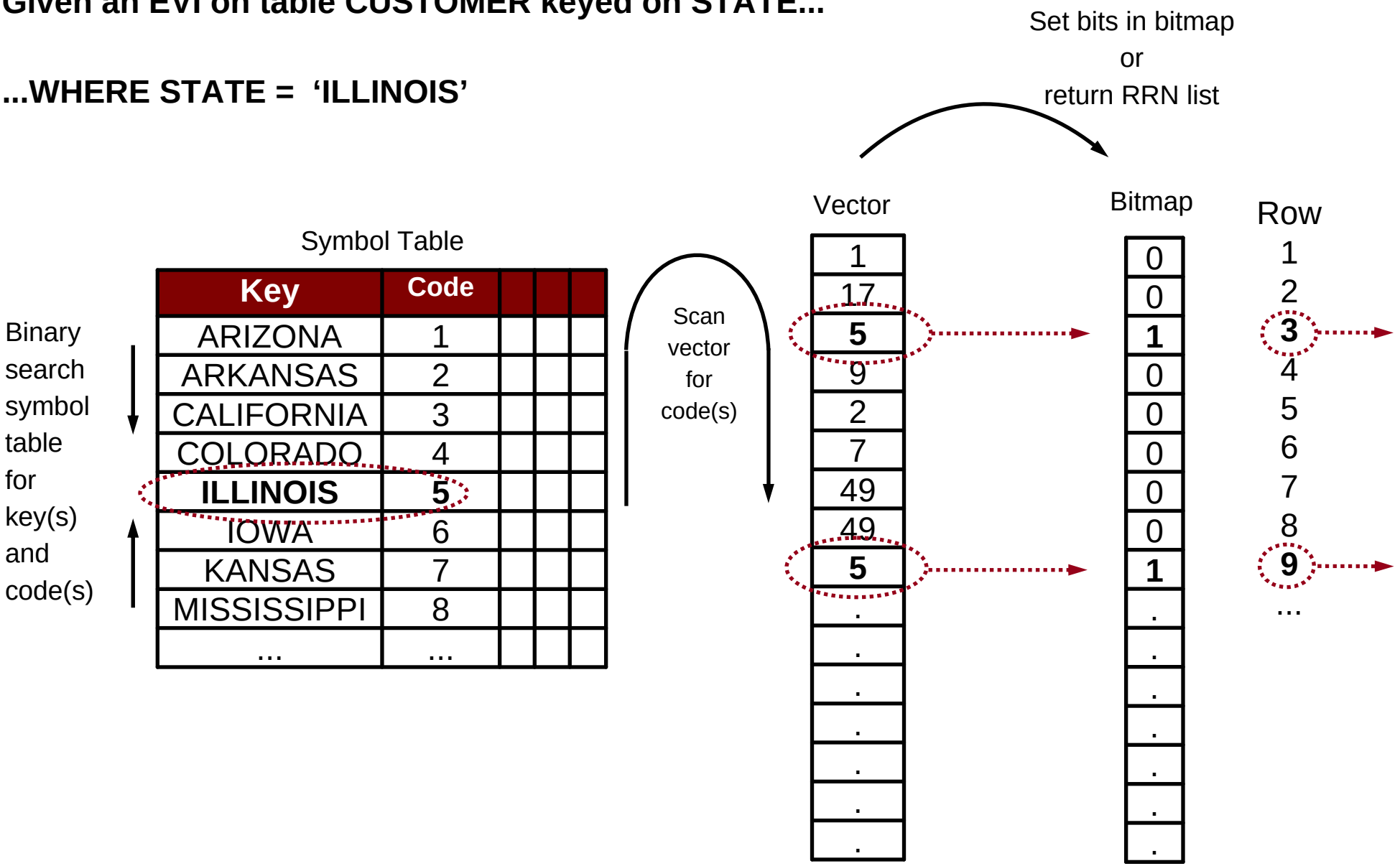
Vector	RRN
1	1
17	2
5	3
9	4
2	5
7	6
49	7
49	8
5	9
…	…

- 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
- Vector is an array of codes
- Elements in the vector are in the same order as the RRNs in the table

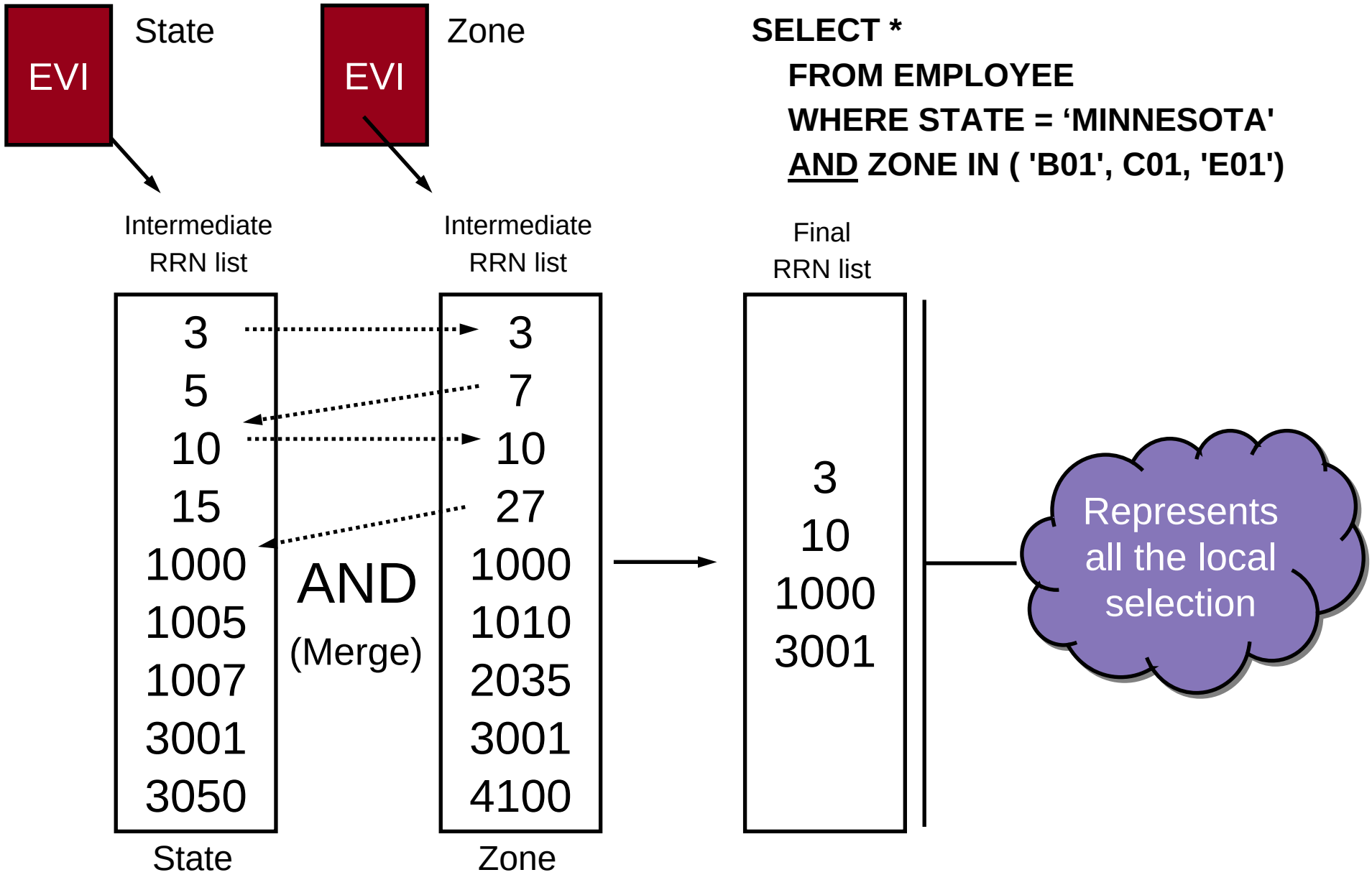
Bitmap / RRN List Example

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

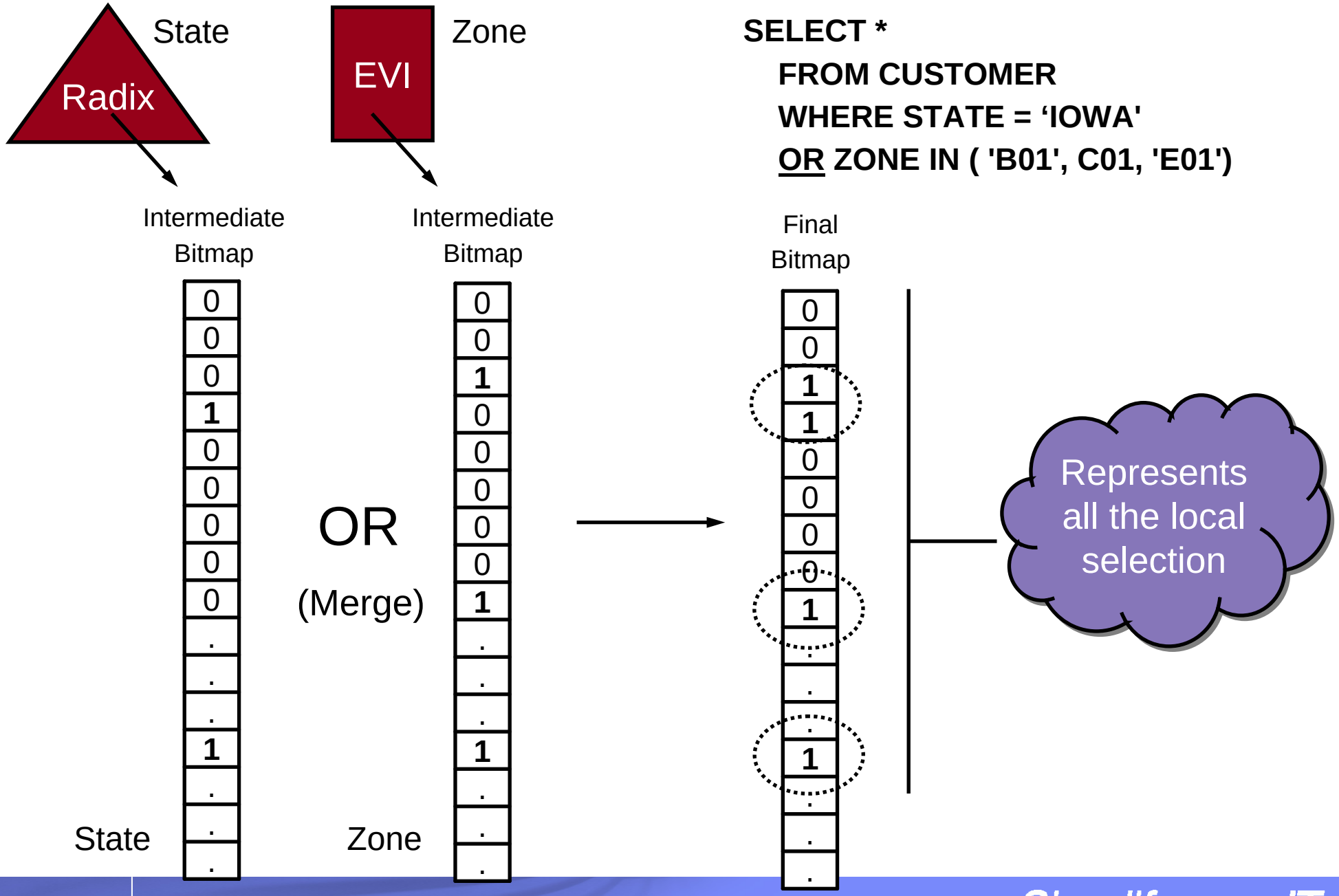
...WHERE STATE = 'ILLINOIS'



Index ANDing / ORing Example



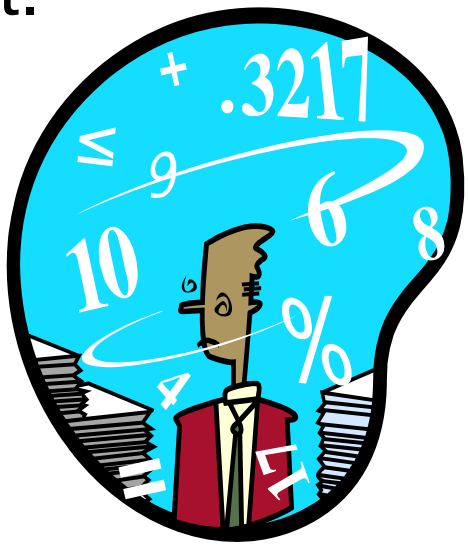
Index ANDing / ORing Example 1



DB2 for i5/OS

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

Indexes, how are they used in DB2 for i5/OS?

- SELECTING
  - Table scan
  - ✓ **Table scan or probe via bitmap or RRN list**
  - ✓ **Index scan or probe**
- JOINING
  - ✓ **Index scan or probe**
  - Hash table probe
  - Sorted list probe
- GROUPING
  - ✓ **Index scan or probe**
  - Hash table scan
- ORDERING
  - ✓ **Index scan or probe**
  - Sorted list scan

Many, many different combinations of techniques can be used, especially with SQE.

Visual Explain is the best tool to see them all in context.

Temporary Indexes…

It’s Autonomic!

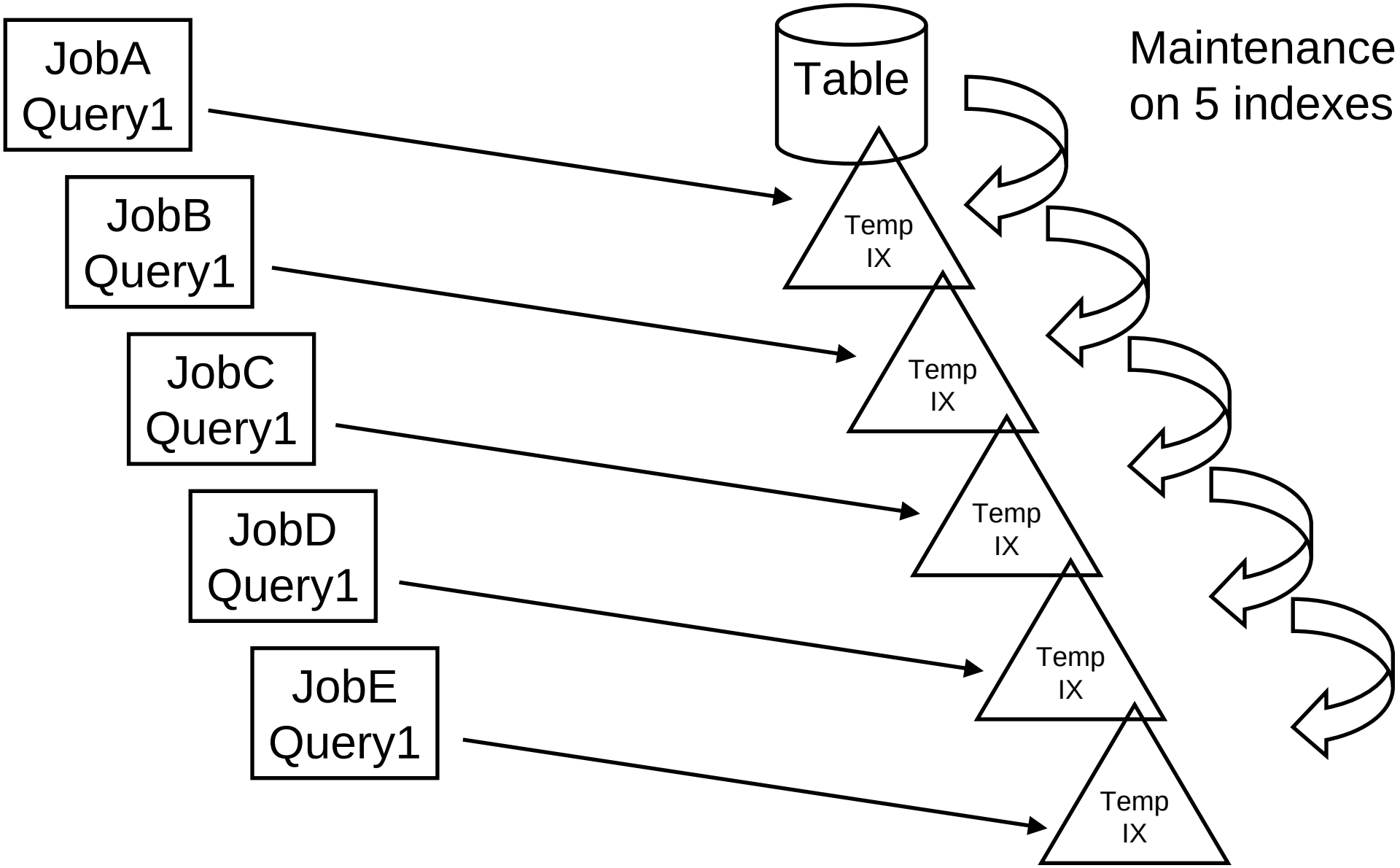
Autonomic Index Creation – Why?

- When a proper permanent index is not available, choices are limited
- CQE optimizer behavior leans toward using an index
- SQE considers all temporary data structures equally
- When a permanent index is not available, one can be, or will be created
- The cost (i.e. time) to create the index is taken into consideration
- Index access is required for certain environments
  - Sensitive cursor
  - Memory constrained
  - Inequality join condition
  - Predicates with derived values
- Self tune the DB access - SQE

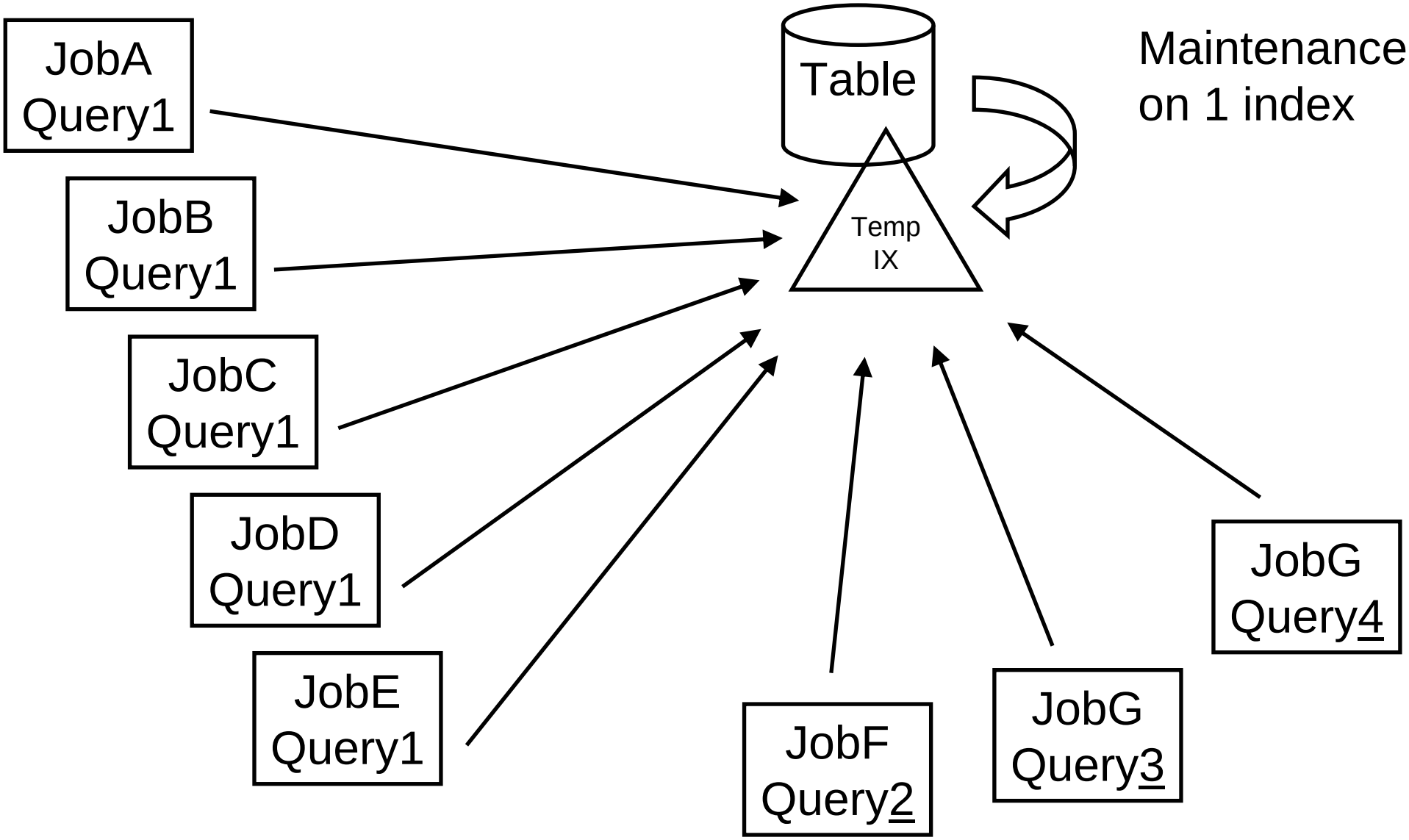
Autonomic Index Creation

- Optimizer can have the DB Engine create a temporary radix 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 significantly 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

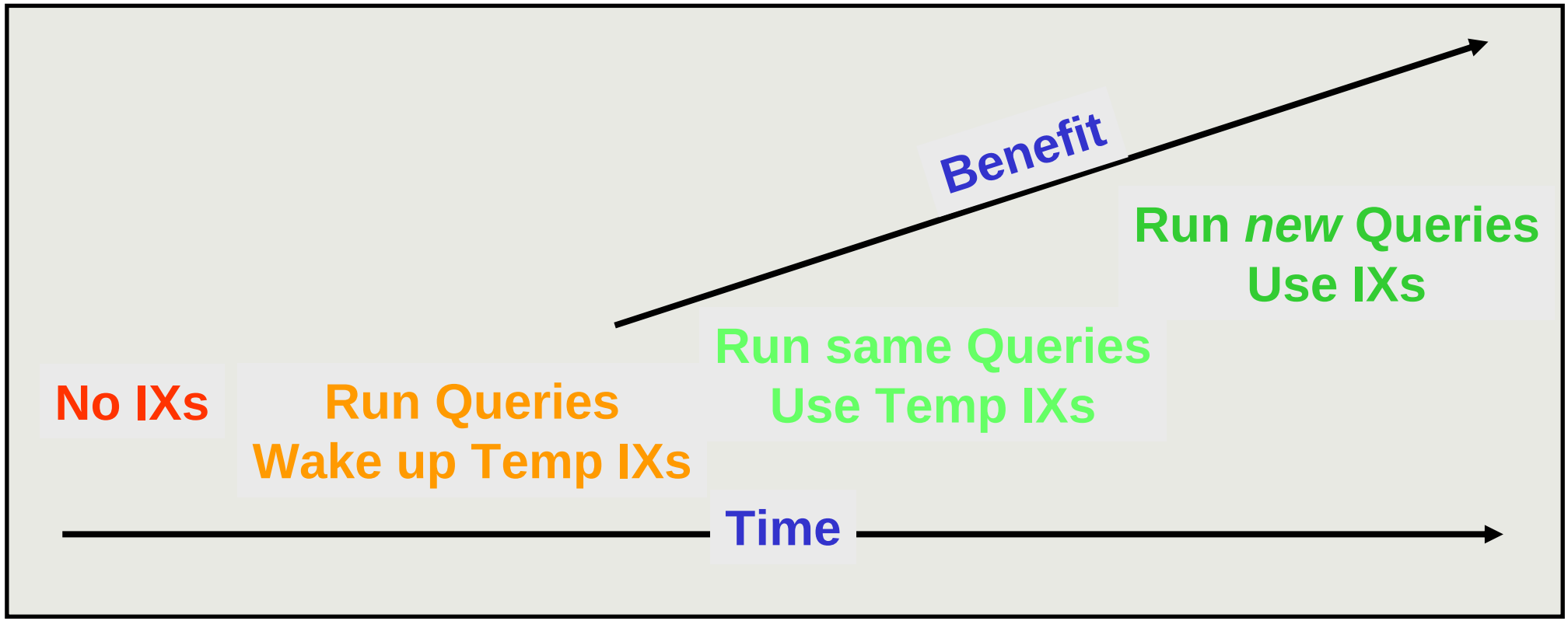
Autonomic Index Creation – CQE behavior



Autonomic Index Creation – V5R4 SQE behavior



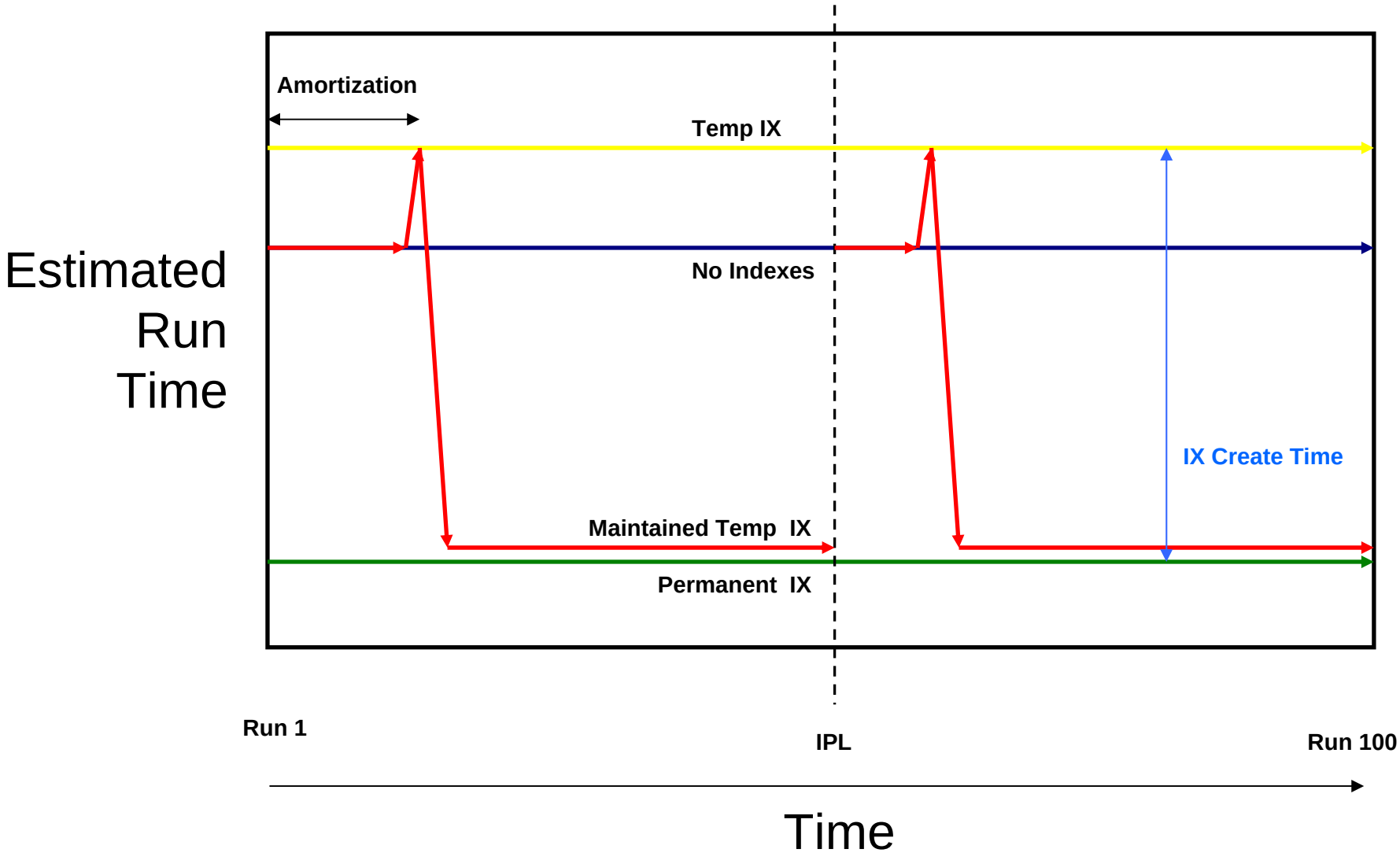
Autonomic Index Creation – SQE benefits



Watching, learning, and tuning table access over time

Autonomic Index Creation – SQE in *Action*

Same query run 100 times…



Autonomic Index Creation Assessment

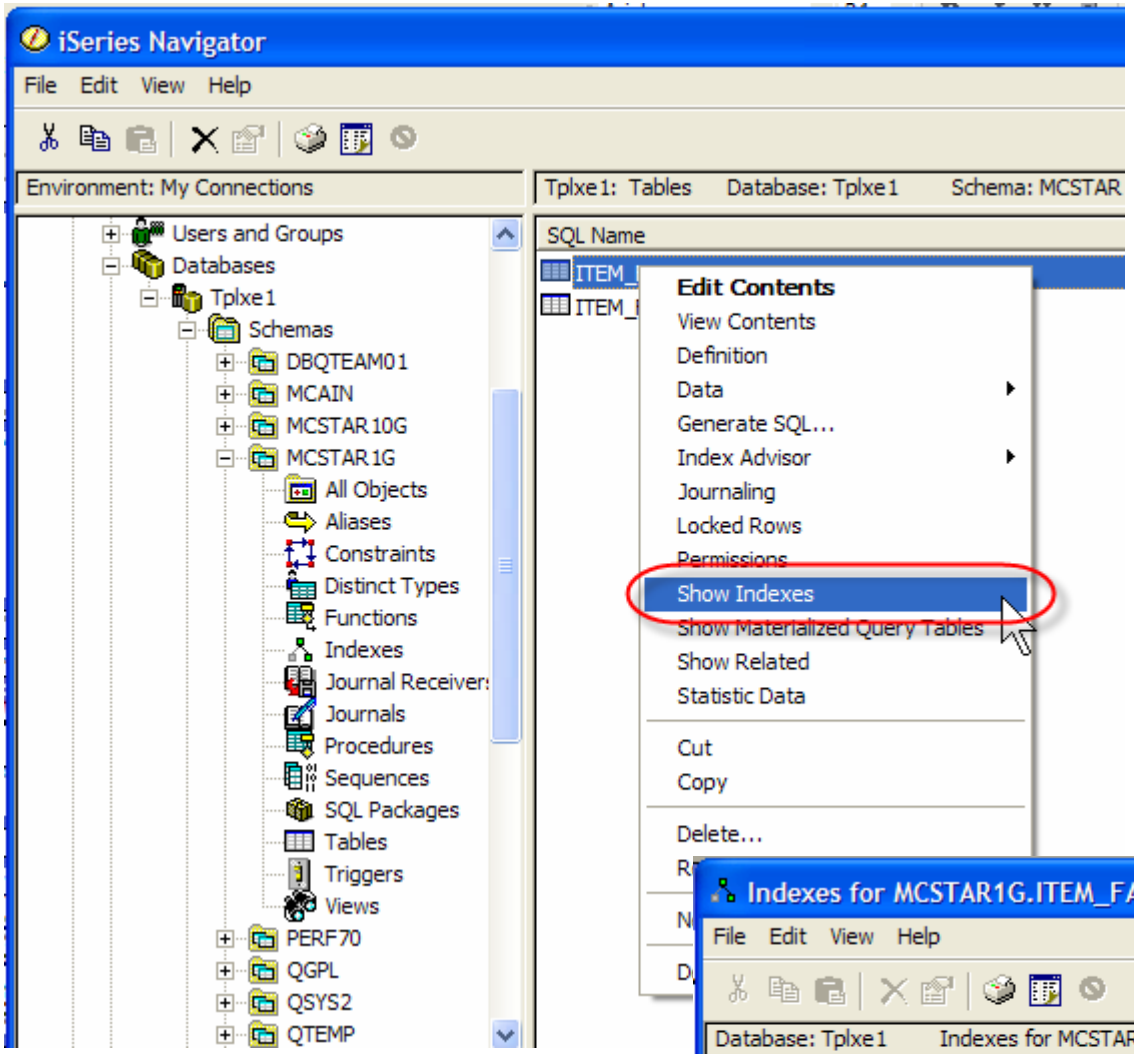
- CQE temporary indexes represent a great opportunity for tuning
  - Temporary indexes are not created for local selection
  - Temporary indexes are not shared or reused
  - Number of temporary index builds can be very large, and costly
- SQE temporary indexes represent DB2 self tuning
  - Temporary indexes are created as needed, based on advice
  - Temporary indexes are shared and reused, providing more benefit, over time
- Temporary indexes are: *temporary!*

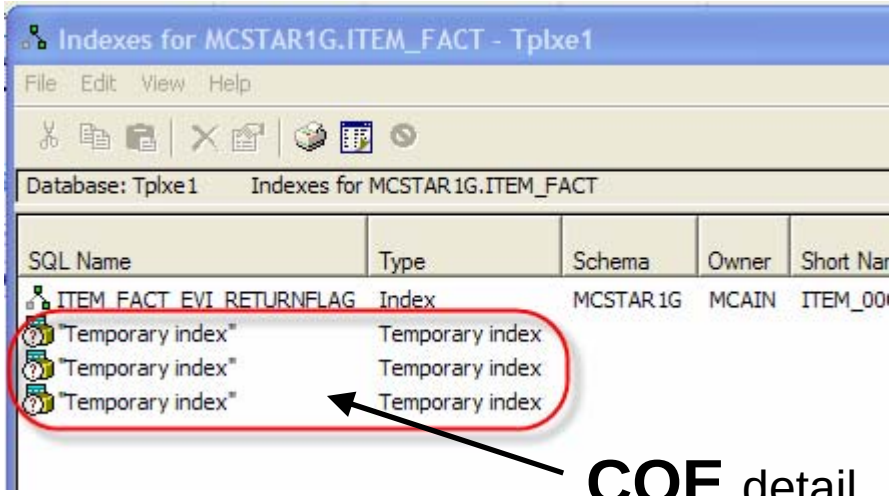
In many situations, lack of indexes and the creation of temporary indexes are the number one cause of poor query performance.

Seek help…

Index Evaluator (Show Indexes)

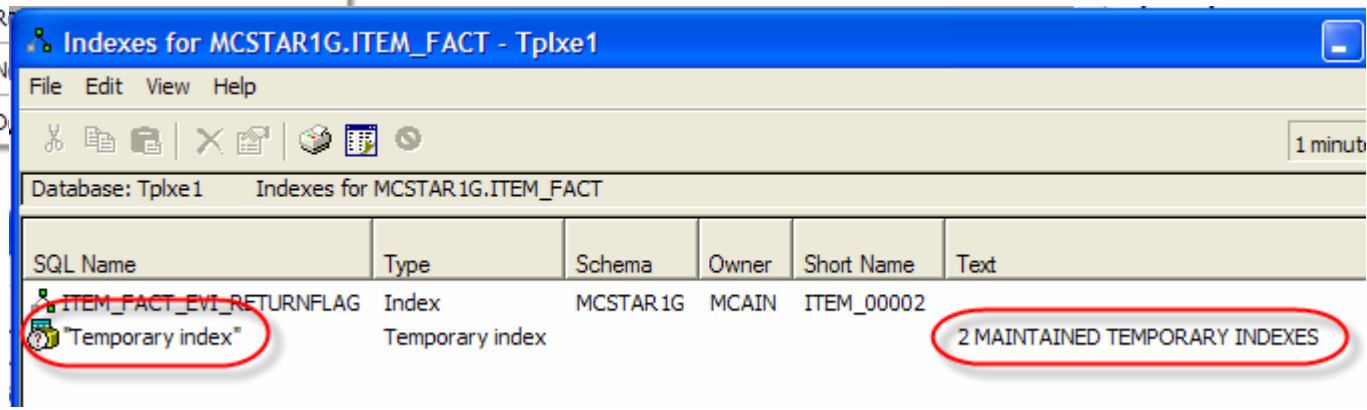
Are there temporary indexes on the table?





CQE detail

SQE summary →



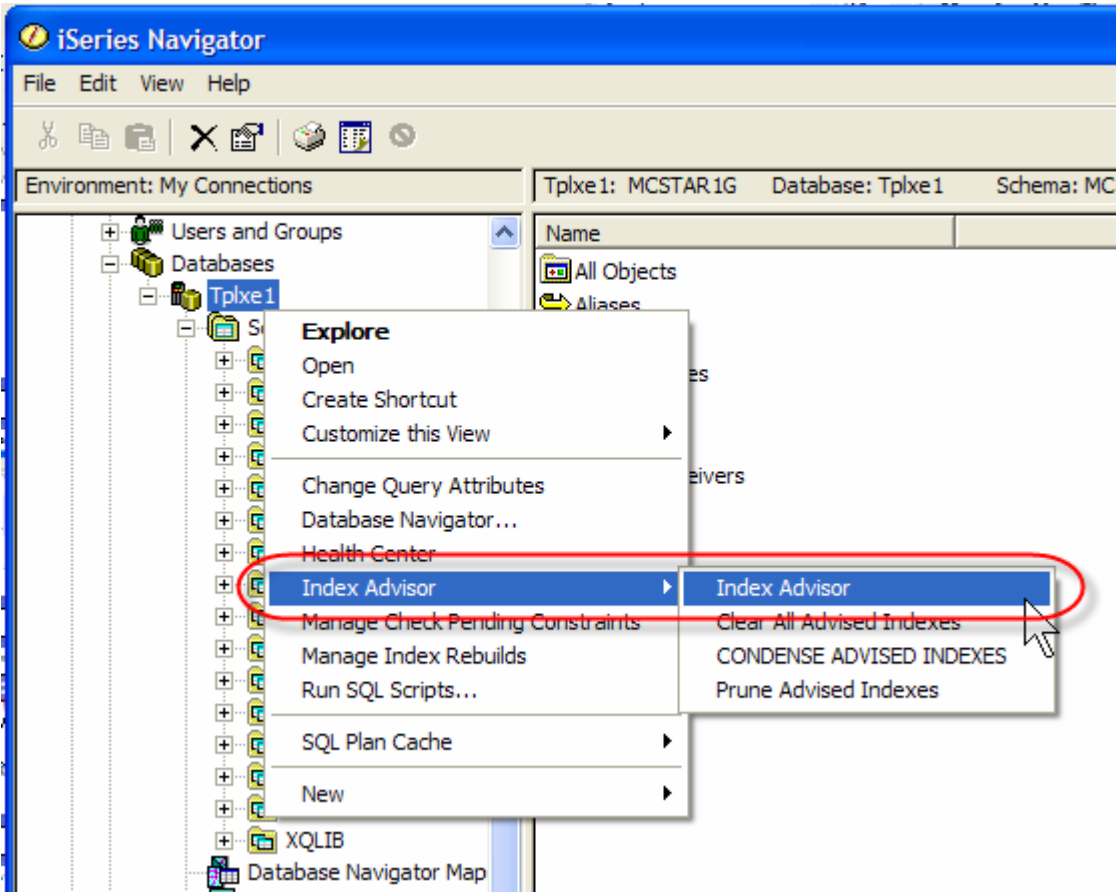
Indexing Advice from the Optimizer

- 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

Index Advised – System wide

- New V5R4 feature
- System wide index advice
  - 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
  - Consolidated, “condensed” advice can help determine best index

Index Advised – System wide



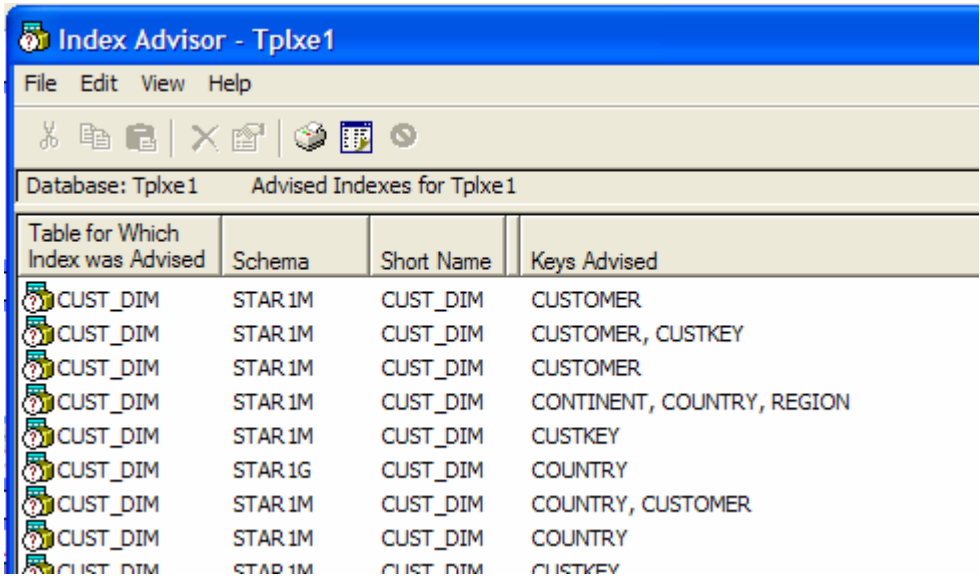
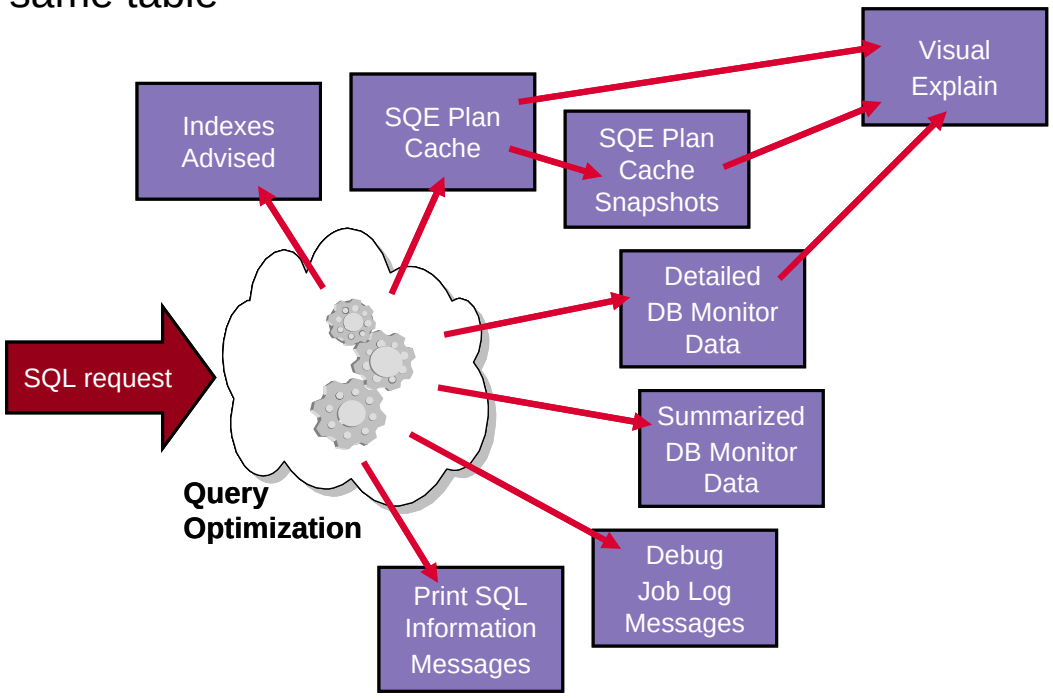


Table for Which Index was Advised	Schema	Short Name	Keys Advised
CUST_DIM	STAR1M	CUST_DIM	CUSTOMER
CUST_DIM	STAR1M	CUST_DIM	CUSTOMER, CUSTKEY
CUST_DIM	STAR1M	CUST_DIM	CUSTOMER
CUST_DIM	STAR1M	CUST_DIM	CONTINENT, COUNTRY, REGION
CUST_DIM	STAR1M	CUST_DIM	CUSTKEY
CUST_DIM	STAR1G	CUST_DIM	COUNTRY
CUST_DIM	STAR1M	CUST_DIM	COUNTRY, CUSTOMER
CUST_DIM	STAR1M	CUST_DIM	COUNTRY

Indexed Advised from other Mechanisms

- SQE Plan Cache (V5R4)
  - Filter queries with 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



Indexing Strategies

DB2 for i5/OS

The goals of creating permanent indexes are:

1. Provide the optimizer the statistics needed to understand the data, based on the query
2. 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

Creating Indexes

- Creating permanent or temporary indexes is resource intensive!
- On multi-CPU servers consider creating indexes with SMP
  - DB2 SMP feature installed (option 26 of i5/OS)
  - CHGQRYA DEGREE(*MAX or *OPTIMIZE)
  - As much memory in the job's pool as possible
  - Single thread index creation jobs through batch (one at a time)
  - Linear scalability: 4 CPUs = ~4x faster creation
- On multi-CPU servers consider creating indexes concurrently
  - CHGQRYA DEGREE(*NONE)
  - As much memory in the job's pool as possible
  - Multi-thread index creation jobs through batch (one job per processor)
- Each additional index created for a table will add overhead when:
  - Inserts, updates, deletes to the table
  - Optimization occurs for that table (index evaluation)
  - Overhead is less than you think

DB2 for i5/OS

To be successful…

You must create some indexes!

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

Indexing Strategy - Basic Approach

Radix Indexes

- Local selection columns
- Join columns

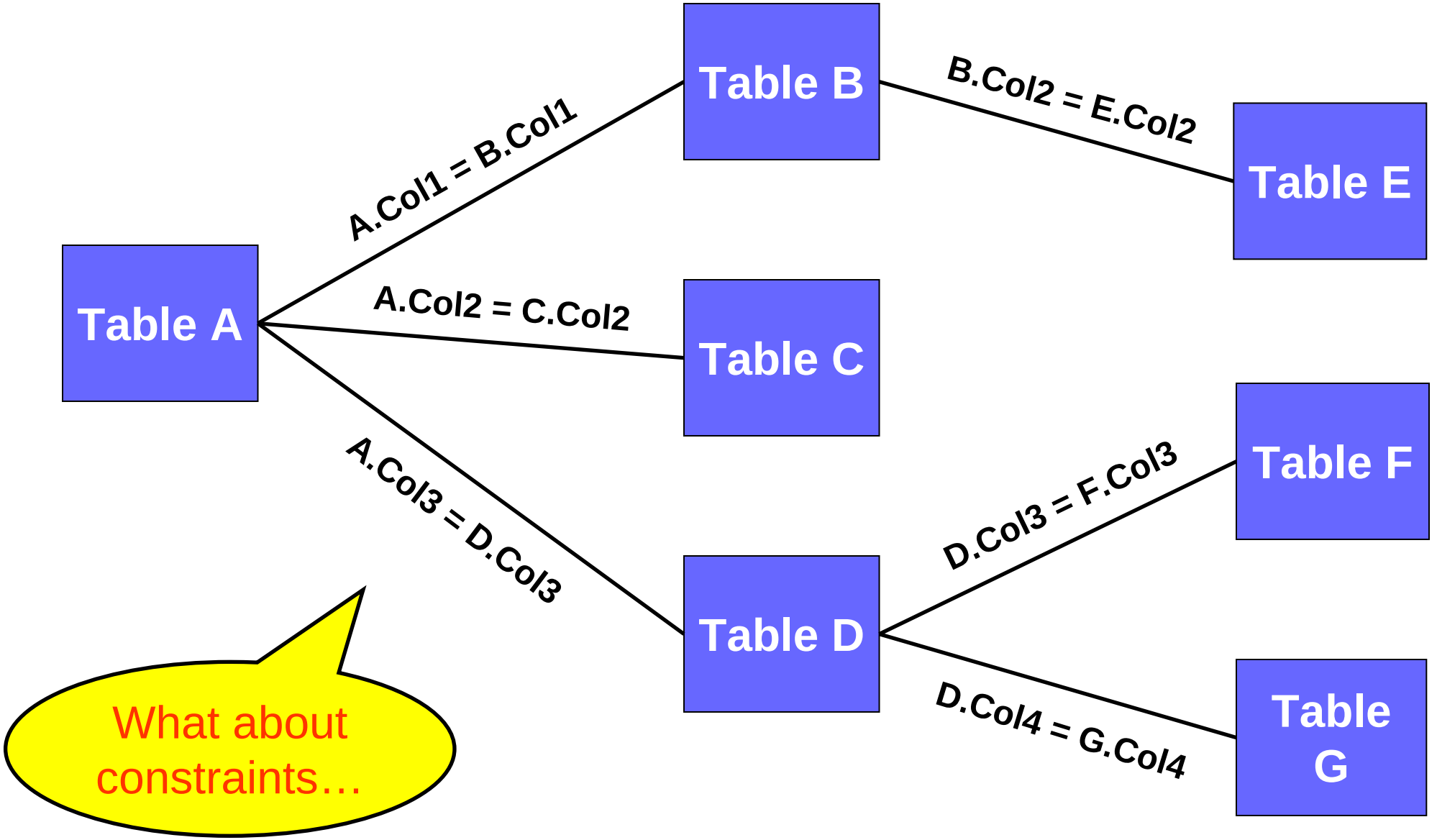
} Minimum

- 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)
- Simple aggregates – COUNT(*) or DISTINCT

Indexing Strategy - Examples



Indexing Strategy - Examples

If the optimizer information 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 use → Create an index on join columns
→ Create an index on grouping columns

Indexing Strategy – Maintenance v Query Access

- 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
- Drop unnecessary 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)
  - Set protection threshold appropriately

Questions & Answers…

Thank You

Trademarks

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml: AS/400, DBE, e-business logo, ESCO, eServer, FICON, IBM, IBM Logo, iSeries, MVS, OS/390, pSeries, RS/6000, S/30, VM/ESA, VSE/ESA, Websphere, xSeries, z/OS, zSeries, z/VM

The following are trademarks or registered trademarks of other companies

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
LINUX is a registered trademark of Linux Torvalds
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered trademark of Intel Corporation
* All other products may be trademarks or registered trademarks of their respective companies.

NOTES:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput 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 improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

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

Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.

The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.