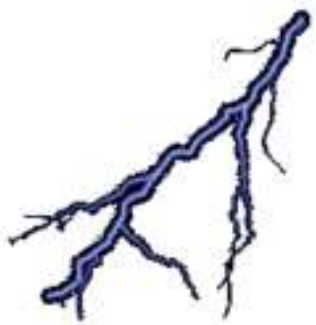
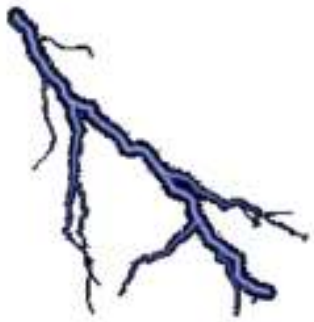
A DB2 Performance Tuning Roadmap:

A High-Level View on Managing the Performance of DB2 for z/OS





Craig S. Mullins
craig@craigsmullins.com

Author

Mullins Consulting, Inc.

This presentation was prepared by:

Craig S. Mullins
President & Principal Consultant

Mullins Consulting, Inc.
15 Coventry Ct.
Sugar Land, TX 77479
Tel: 281-494-6153
E-mail: craig@craigsmullins.com

The Tuning Progression

Problem Resolution

Mullins Consulting, Inc.

Application

- SQL
- Host Language Code

Database

- Indexing
- Database and Index Organization
- Database Design (normalization / denormalization)

DB2 Subsystem

- ZPARMs, Pools, Locking, IRLM, DDF, etc.

Environment

- Network
- TP Monitor (CICS, IMS/TM)
- Operating System

Basic Tuning Rules

Mullins Consulting, Inc.

80% of the results of tuning come from 20% of the tuning effort *-and-*

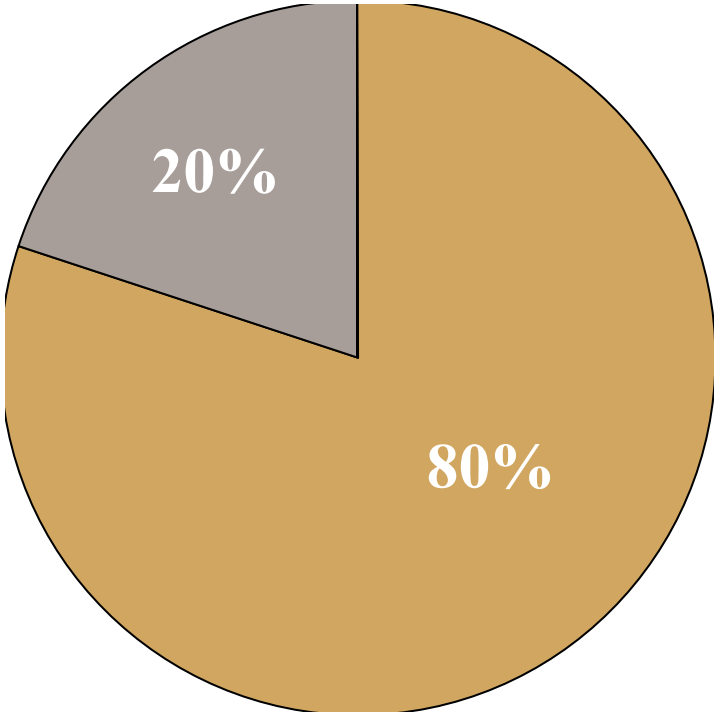
- 20% of your DB2 applications cause 80% of your problems

Tune one thing at a time

- How else do you know whether the action helped or not?

All tuning optimizes:

- **CPU, I/O** or **concurrency**



A Few General Performance Themes to Remember

Mullins Consulting, Inc.

Almost never say always or never.

- There are rarely any “rules” that **always** apply.

Don’t panic and remain humble.

- Remaining calm and open to all solutions, even ones that recognize “you” as being the culprit, is important for building efficient DB2 databases and applications.

It is better to design for performance from the start.

- The further into the development process you are, the more painful it becomes to makes changes.

The Cardinal Rule…

It depends!

- Understand your circumstances and apply what makes sense.

NOTE

The cardinal rule of RDBMS development is **"It depends!"** Most DBAs and SQL experts resist giving a straight or simple answer to a general question because there is no simple and standard implementation that exists. Every situation is different, and every organization is unique in some way.

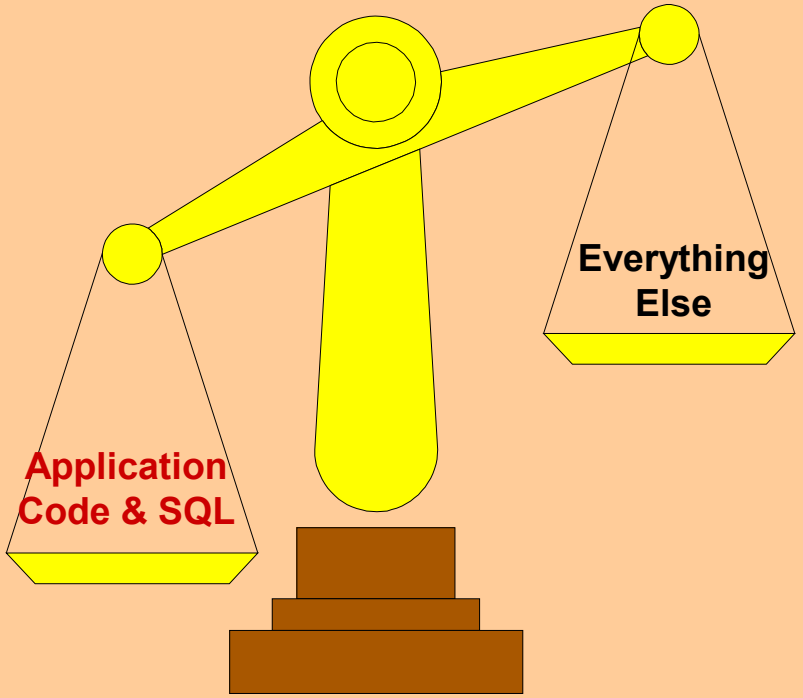
Don't be discouraged when you ask the local expert which statement will perform better, and the answer is "It depends." The expert is just doing his or her job. The secret to optimizing DB2 performance is being able to answer the follow-up question to "It depends"—and that is "What does it depend on?"

The key to effective SQL performance tuning is to document each SQL change along with the reason for the change. Follow up by monitoring the effectiveness of every change to your SQL statements before moving them into a production environment. Over time, trends will emerge that will help to clarify which types of SQL formulations perform best.

Application Code and SQL

Most relational tuning experts agree that the majority of performance problems with applications that access a relational database are caused by poorly coded programs or improperly coded SQL…

- *as high as 70% to 80%*



Application Tuning: SQL

Mullins Consulting, Inc.

Simpler is better, but complex SQL can be efficient

In general, let SQL do the work, not the program

Retrieve the absolute minimum # of rows required

Retrieve only those columns required - never more

Always provide join predicates *(i.e. no Cartesian products)*

Favor Stage 1 and Indexable predicates

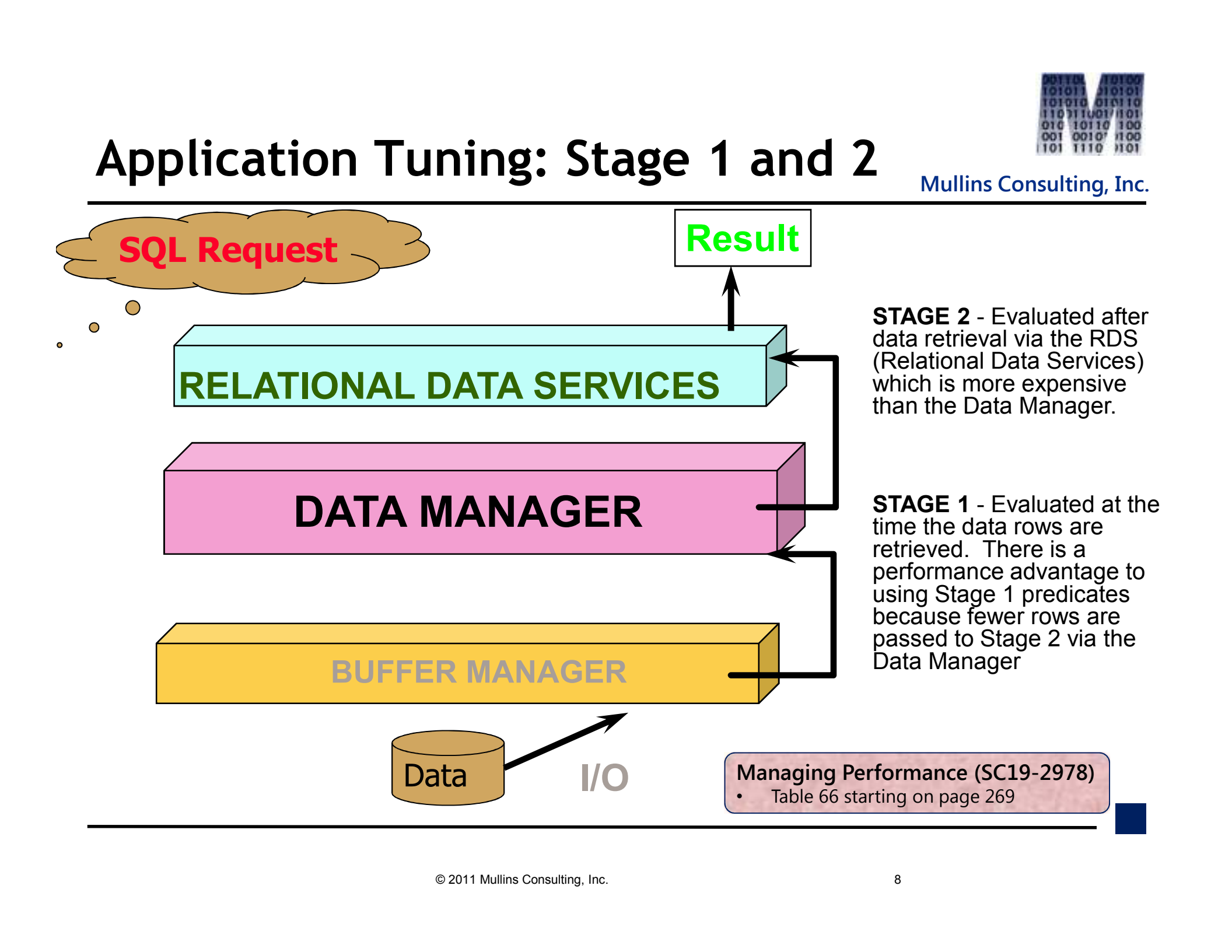
- Host variable data type/length should match column

Avoid tablespace scans for large tables *(usually)*

Avoid sorting when possible:

- indexes for ORDER BY and GROUP BY
- judicious use of DISTINCT
- UNION ALL versus UNION *(if possible)*

Application Tuning: Stage 1 and 2



STAGE 2 - Evaluated after data retrieval via the RDS (Relational Data Services) which is more expensive than the Data Manager.

STAGE 1 - Evaluated at the time the data rows are retrieved. There is a performance advantage to using Stage 1 predicates because fewer rows are passed to Stage 2 via the Data Manager

Managing Performance (SC19-2978)
- Table 66 starting on page 269

Stage 3?

Mullins Consulting, Inc.

Not an actual Stage but

- It can be helpful to think of moving predicates from SQL into your programs as Stage 3
- Stage 1 better than Stage 2
- Stage 2 better than Stage 3

Ask Only for What You Absolutely Need

Mullins Consulting, Inc.

Retrieve the absolute minimum # of rows required

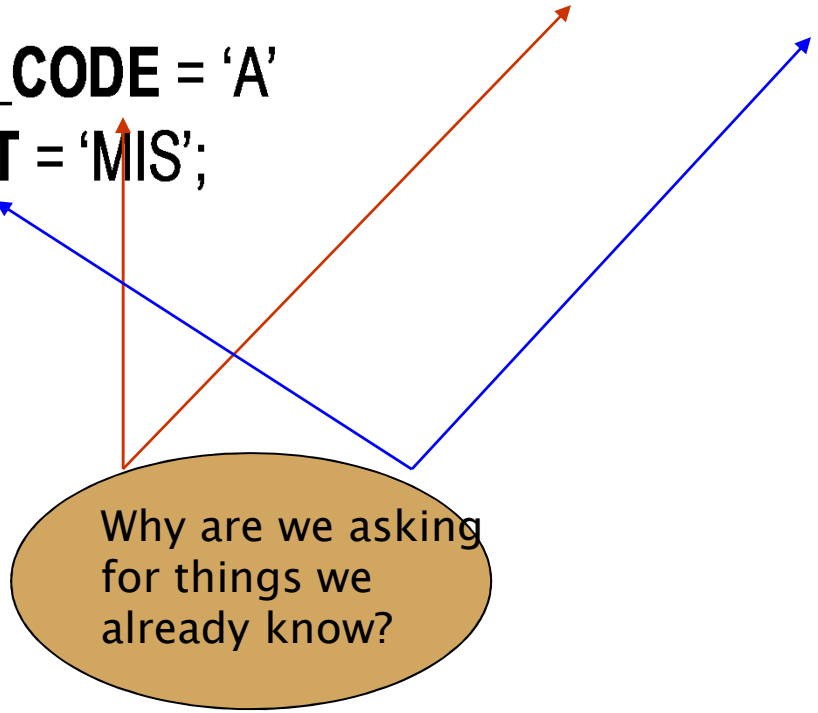
- Code appropriate WHERE clauses
- The only rows that should be returned to your program should be those that you need to process

Retrieve only those columns required: never more

- Don't ask for what you don't need
- Sometimes shortened to → Avoid SELECT *
  - This is a good idea for several reasons:
    1. Insulation of programs from change
    2. Performance
  - But it is not enough…

What is Wrong with this SQL?


```
SELECT  LAST_NAME, FIRST_NAME, JOB_CODE, DEPT
FROM    EMP
WHERE   JOB_CODE = 'A'
AND     DEPT = 'MIS';
```

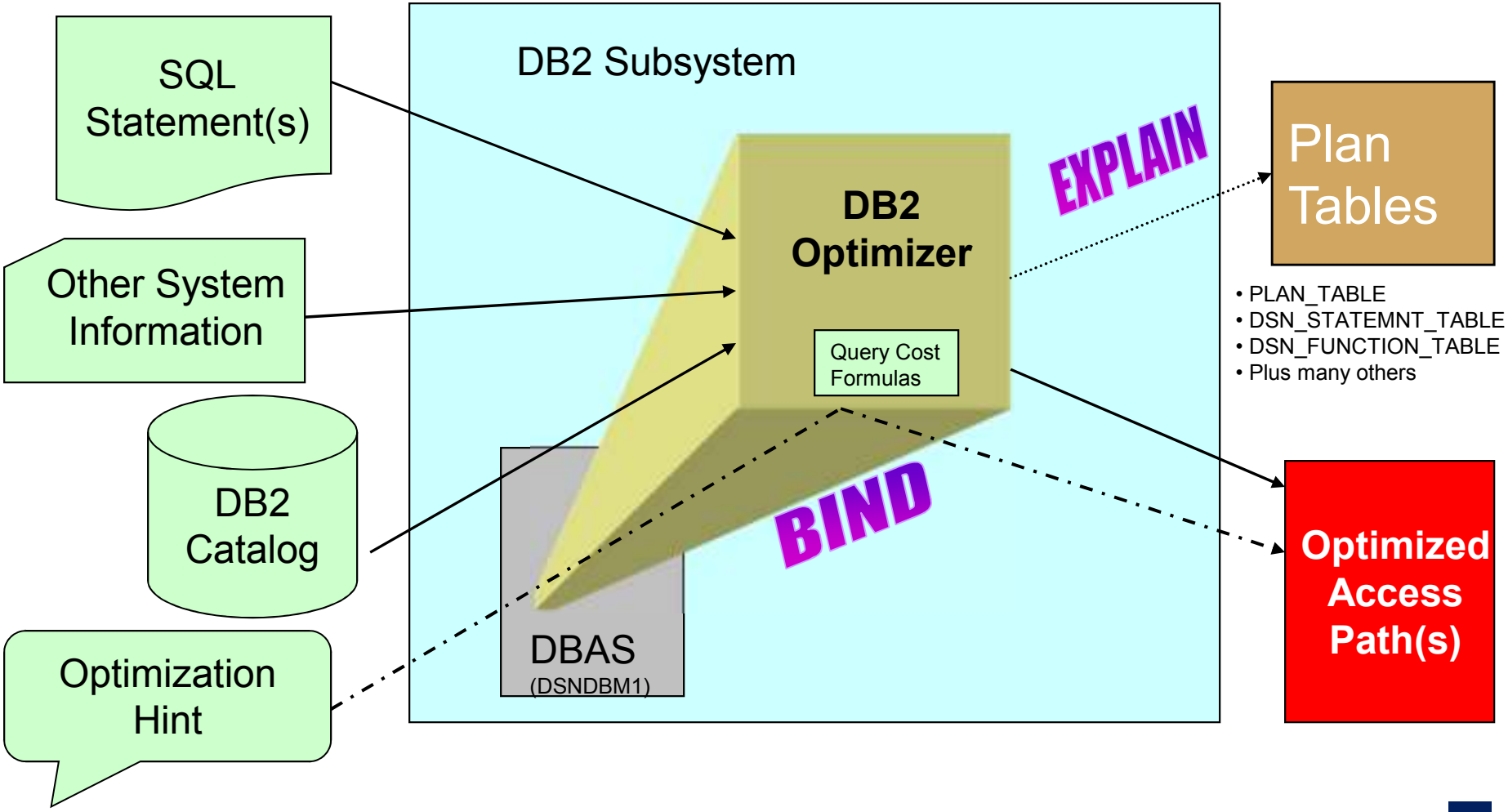


Avoid Black Boxes

Stage 4?

http://www.craigsmullins.com/dbu_0703.htm

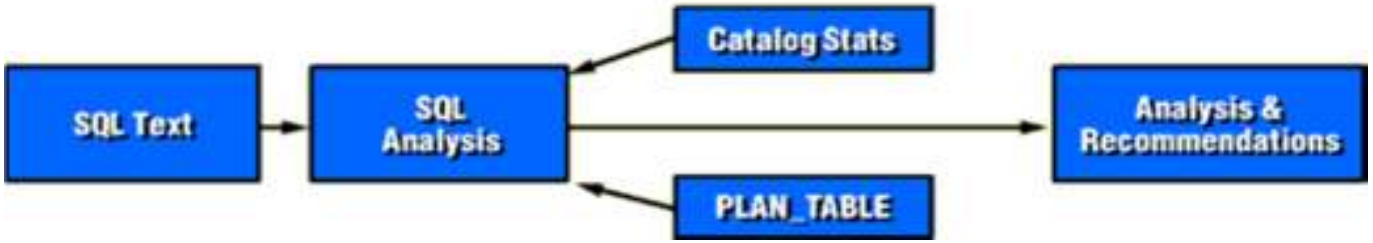
Application Tuning: Optimization



OPTHINT in PLAN_TABLE

Application Tuning: *EXPLAIN Analysis*

Mullins Consulting, Inc.



Hint used?

Index used?

- Single, Multiple

Matching column(s)?

Index only?

TS scan ***(page range)***

Type of Join?

- Nested Loop
- Merge Scan
- Hybrid

SQL Text

Table & Index Information

- DDL
- Stats

Cardinality

Other Stuff

- Triggers
- RI
- Constraints

Prefetch?

- Sequential
- List

Parallelism used?

- I/O, CPU, Sysplex
- Degree

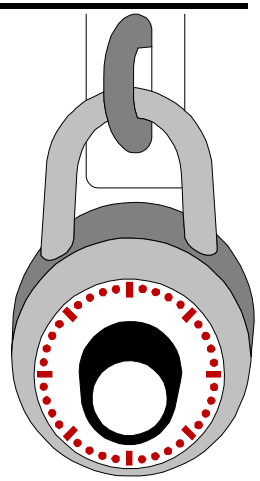
Sort required?

- Join, Unique, Group By, Order By

Locking

Application Tuning: Locking

Mullins Consulting, Inc.



Minimize deadlocks by coding updates in the same sequence regardless of program

Issue data modification SQL statements as close to the end of the UOW as possible

- the later in the UOW the update occurs, the shorter the duration of the lock

Encourage Lock Avoidance

- ISOLATION(CS) / CURRENTDATA(NO)
- Can be used only by read only cursors

Use LOCK TABLE judiciously

Consider ISOLATION(UR) to avoid locking

Application Tuning: Commit

Mullins Consulting, Inc.

Avoid Bachelor Programming Syndrome

Fear of COMMITing

Plan and implement a COMMIT strategy

- or experience TIMEOUTs and DEADLOCKs

Application Tuning: Program

Mullins Consulting, Inc.

Do not embed efficient SQL in inefficient program logic

- Classic example: finely tuned, efficient SQL inside of a program loop that executes 3,510,627 times!

Let SQL do the work when possible

- e.g.) do not code "program" joins

Sign of trouble: SQL followed by lots of IF...ELSE or CASE statements

If you are only going to retrieve one row, consider coding a singleton SELECT (usually)

Consider adopting multi-row FETCH

- Multiple tests have shown that moving to multi-row FETCH can yield between a 50% to 60% improvement for 100 and 1000 FETCHes

Application Tuning:
Online vs. Batch

Mullins Consulting, Inc.

When designing online transactions, limit the amount of data to be retrieved to a reasonable amount

- No one reads hundreds of pages/screens online!

Limit online sorting and joining *(but be reasonable)*

Consider OPTIMIZE FOR 1 ROW to disable list prefetch

- With list prefetch, DB2 acquires a list of RIDs from a matching index, sorts the RIDs, & accesses data by the RID list
- Can be very inefficient for a multiple page transaction

Dynamic vs. Static SQL

Mullins Consulting, Inc.

Dynamic SQL

- Dynamic SQL is coded and embedded into an application program differently than static SQL. Instead of hard-coding, the SQL is built within the program “on the fly” as it executes.
- Once built, the dynamic SQL statement must be compiled using the PREPARE statement; or, alternately, an implicit PREPARE is issued behind the scenes when implementing the EXECUTE IMMEDIATE flavor of dynamic SQL.

Static SQL

- Static SQL is hard-coded and embedded into an application program. The SQL is bound into a package, which determines the access path that DB2 will use when the program is run. Although dynamic SQL is more flexible than static, static SQL offers some flexibility by using host variables.

Static vs. Dynamic SQL Considerations

Mullins Consulting, Inc.

The following criteria should be used when determining whether to favor dynamic SQL over static SQL:

- Performance sensitivity of the SQL statement
  - Dynamic SQL will incur a higher initial cost per SQL statement due to the need to prepare the SQL before use. But once prepared, the difference in execution time for dynamic SQL compared to static SQL diminishes.
- Data uniformity
  - Dynamic SQL can result in more efficient access paths than static SQL is whenever data is:
    1. Non-uniformly distributed. (e.g. cigar smokers skews male)
    2. Correlated (e.g. CITY, STATE, and ZIP_CODE data will be correlated)
- Use of range predicates
  - The more frequently you need to use range predicates (<, >, <=, >=, BETWEEN, LIKE) the more you should favor dynamic SQL.
  - The optimizer can take advantage of distribution statistics & histogram statistics to formulate better access paths because the actual range will be known.

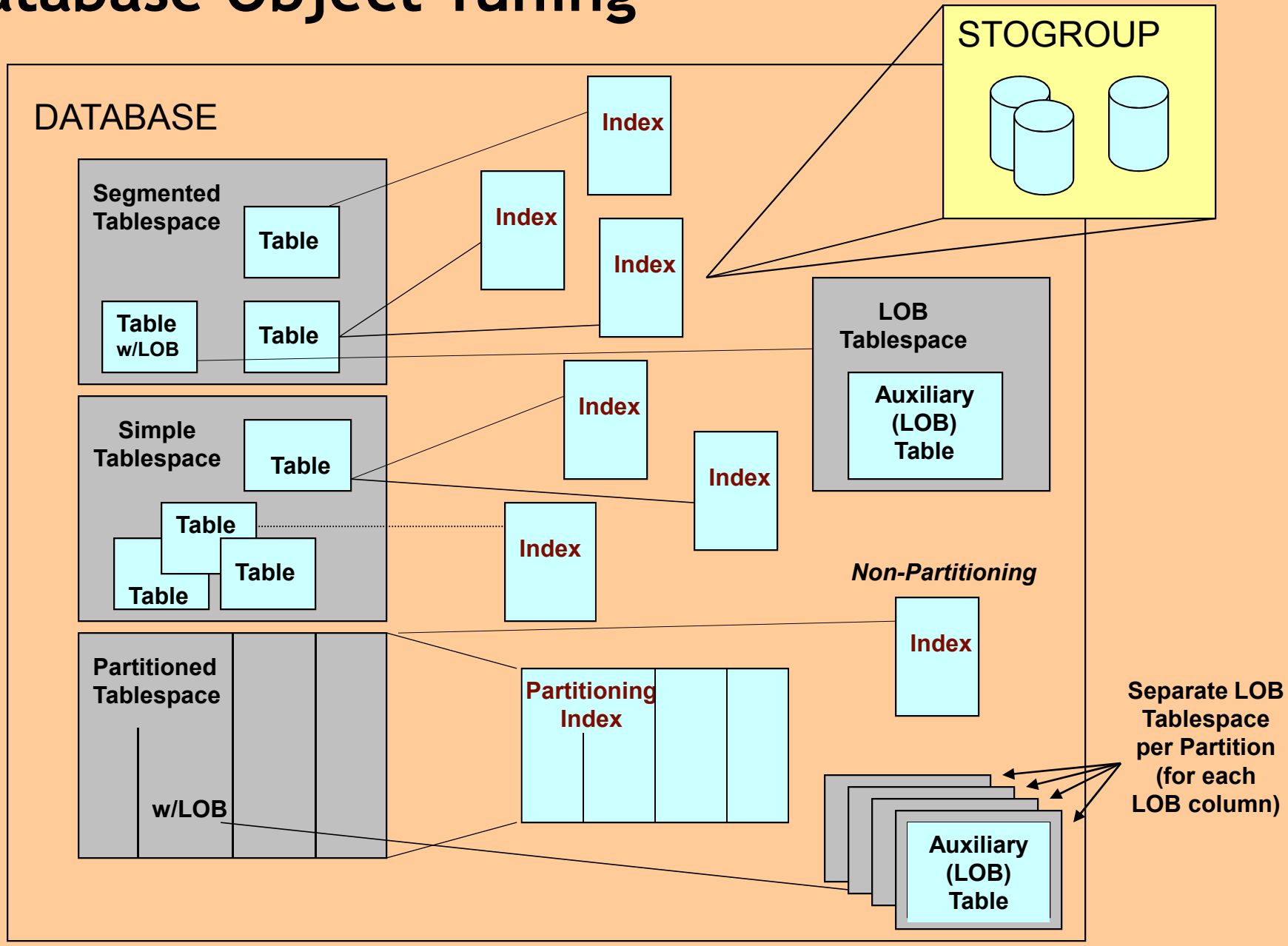
Static vs. Dynamic SQL

(Considerations continued)

Criteria for determining use of dynamic or static SQL (cont.)

- Repetitious Execution
  - As the frequency of execution increases, then you should favor static SQL (or perhaps dynamic SQL with local dynamic statement caching (KEEPDYNAMIC YES).
  - The cost of the PREPARE becomes a smaller and smaller percentage of the overall run time of the statement the more frequently it runs (if the cached prepare is reused).
- Nature of Query
  - When you need all or part of the SQL statement to be generated during application execution favor dynamic over static SQL.
- Run Time Environment
  - Dynamic SQL can be the answer when you need to build an application where the database objects may not exist at precompile time. Dynamic might be a better option than static specifying VALIDATE(RUN).
- Frequency of RUNSTATS
  - When your application needs to access data that changes frequently and dramatically, it makes sense to consider dynamic SQL.

Database Object Tuning



Universal Table Spaces Version 9

Mullins Consulting, Inc.

Combine the space management of segmented table spaces with the organization of partitioned table spaces.

Types of universal table space:

- Partition by range (or range partitioned)
- Partition by growth

Benefits of universal table space:

- Flexibility
- Growth
- Uses space map pages, like segmented
  - More information about free space space than just partitioned
  - Improved mass delete performance
  - Immediate reuse of segments after a table is dropped or mass deleted

Universal Table Spaces
Partition By Growth (PBG)

Version 9

Mullins Consulting, Inc.

Ideal when a table is expected to exceed 64 GB but there is no suitable partitioning key

- Begins as a single-partition table space
- Grows automatically: partitions are added as needed, as the data volume expands
- Can grow up to 128 TB
  - The maximum size is determined by the MAXPARTITIONS and DSSIZE values that you specified and the page size

Consider UTS PBG TS as replacement for simple TS

Universal Table Spaces
Partition By Range (PBR)

Version 9

Mullins Consulting, Inc.

Partition by range, or range-partitioned, universal table spaces are created by specifying both SEGSIZE and NUMPARTS on the CREATE TABLESPACE statement.

- All actions that are allowed on exclusively partitioned or exclusively segmented table spaces are allowed on range-partitioned universal table spaces.

Ranges for range-partitioned universal table space can be specified on subsequent CREATE TABLE (or CREATE INDEX statements).

General Table Space Recommendations

Mullins Consulting, Inc.

As of DB2 V9, favor universal table spaces over segmented or traditional partitioned table spaces

- UTS are the future of DB2 table spaces
- At some point, other table space types are likely to be deprecated (like simple already have been)

In most cases limit yourself to one table per table space

- You can still use a segmented table space when you must have multi-table TS

DSSIZE < 4GB unless you definitely need large TS

Database Organization

Mullins Consulting, Inc.

Be sure to run RUNSTATS

- as data volume changes, new data structures added
- followed by (RE)BIND with /EXPLAIN(YES)

Review statistics (or RTS) to determine when to REORG

- NEARINDREF and FARINDREF
- LEAFDIST, PERCDROP
- For clustering indexes
  - NEAROFFPOSF and FAROFFPOSF
  - CLUSTERRATIOF
- Migrate to Real Time Statistics!
- Analyze access patterns before reorganizing
  - Random vs. sequential
  - Consider automation

Table Design Basics

Mullins Consulting, Inc.

As normalized as possible, but performance before aesthetics; normalization optimizes "update" at the expense of "retrieval"

- Don't let data modelers dictate "physical" design

Do not create base table views

Avoid the defaults - *they are usually wrong*

Determine amount of free space

- PCTFREE – amount of each page to remain free during REORG
- FREEPAGE – after this many pages of data, keep an empty page
- Based on volatility
- Don't just let everything default (for example, to 10).

Database Design: Rows & Columns

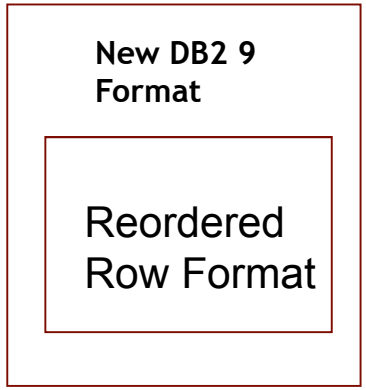
Mullins Consulting, Inc.

Avoid wasted space *(page size?)*

- Row length > 4056 requires larger page size
- Row length 2029 - 4056 = one row per page
  - Ex) 2500 bytes: page size?
- Row length < 15 wastes space (max 255 rows/page)

Sequence columns based on logging

- Infrequently updated non-variable columns first
- Static (infrequently updated) variable columns
- Frequently updated columns last
- Frequently modified together, place next to each other



Database Design: Data Types

Mullins Consulting, Inc.

Use NULL sparingly

Use appropriate DB2 data types

- Use DATE instead of CHAR or numeric for dates
- Store numeric data using a numeric data type
  - INTEGER, SMALLINT, DECIMAL, etc.
- INTEGER versus DECIMAL(x,0)
  - Control over domain vs. storage requirements
- "DATE and TIME" versus TIMESTAMP
  - Ease of use/storage vs. precision/arithmetic

Be Aware Of New DB2 9 Data Types

BIGINT
DECFLOAT
VARBINARY
XML

Compression versus VARCHAR

- Compression = less overhead *(no 2 byte prefix)*
- Compression requires no programmatic handling

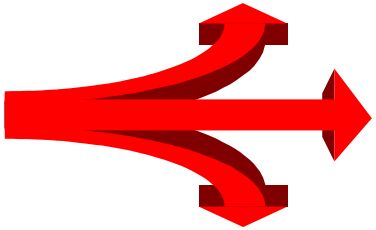
Database Design: Integrity

Mullins Consulting, Inc.

Use DB2 declarative RI instead of program RI *(usually)*

- performance and ease of use
- ensure integrity for planned and ad hoc database modification

Do not use RI for lookup tables *(overkill)*



- consider CHECK constraints vs. lookup tables

Use triggers only when declarative RI is not workable

- Triggers are less efficient *(usually)* than RI
  - but usually better than enforcing in application programs

Specify indexes on foreign keys

Index Usage Basics

Mullins Consulting, Inc.

Know which columns are indexed

- …and favor specifying them in your WHERE clauses
- Obviously, if the column is *not* in an index DB2 can never use an index to satisfy the predicate.
  - There are other considerations.

Specify the leading column of composite indexes

- For example: IX1 (LNAME, FNAME, MIDINIT)

```
SELECT * FROM CUST WHERE LNAME = ?
```

- If instead… `WHERE FNAME = ?`
  - DB2 could not do a direct index lookup

Database Design: Indexes

Mullins Consulting, Inc.

A proper indexing strategy can be the #1 factor to ensure optimal performance

First take care of unique & PK constraints

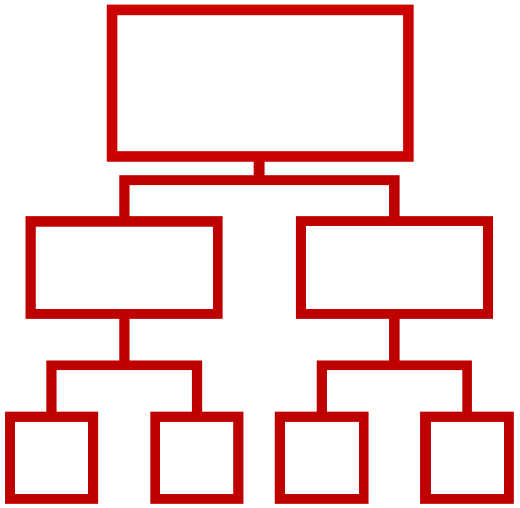
Then for foreign keys *(usually)*

Heavily used queries - predicates

Overloading of columns for IXO

Index to avoid sorting

- ORDER BY, GROUP BY



Consider INS / UPD / DEL implications

Consider how to index variable cols - [PADDED | NOT PADDED]

Index not used? SYSIBM.SYSINDEXSPACESTATS.LASTUSED

Database Design: Buffer Pools, Pt. 1

Mullins Consulting, Inc.

Bufferpool allocations - do not default everything to BP0

- Explicitly specify a buffer pool for every table space and index

Ideas:

- isolate the catalog in BP0
- separate indexes from table spaces
- isolate heavily hit data
- isolate sort work area
- optimize BP strategy for your data & app processing mix: sequential vs. random
- there is no “silver bullet” approach
  - more on bufferpool tuning coming up!

REORG/RUNSTATS/REBIND

BIND and REBIND

Mullins Consulting, Inc.

The BIND and REBIND commands determine the access paths that the SQL in your programs will use to get to the data

- **BIND** – accepts a DBRM (Data Base Request Module) and binds it into a package (or plan, until V10, at which point DBRMs will not be able to be bound directly to plans)
  - SQL can change

- **REBIND** – take a package (or plan) that has already been bound and re-assesses the access paths based on the latest statistics
  - SQL cannot change

Why Rebind?

Mullins Consulting, Inc.

Data volumes have changed

- More data
- Less data

Data skew has changed

Data distribution has changed

DB2 has changed

- PTF
- New version

Environment has changed

Scheduling Rebinds…

Mullins Consulting, Inc.

REBIND is critical for application performance

It is a wise course of action to plan your REBIND strategy

There are several common approaches:

- Regular maintenance: REBIND after RUNSTATS
  - Perhaps not every day, but REBIND are done after RUNSTATS
- Global REBIND after migration to new DB2 version
- Global REBIND after installing new PTFs
  - Above two mean access paths only change when DB2 changes
- REBIND after x days / weeks / months …
- Let it Ride! *("If it ain't broke, don't fix it.")*

Let It Ride

Programs once bound, are (almost) never rebound.

- Reason:
  - Fear of access path degradation
- Result:
  - No improvement to access paths
  - No CPU savings from new DB2 efficiencies
  - Sub-optimal performance
  - Every DB2 program potentially suffers for fear that one or two SQL statements will become inefficient

Regular REBIND

Mullins Consulting, Inc.

Better Approach: Regular REBINDing

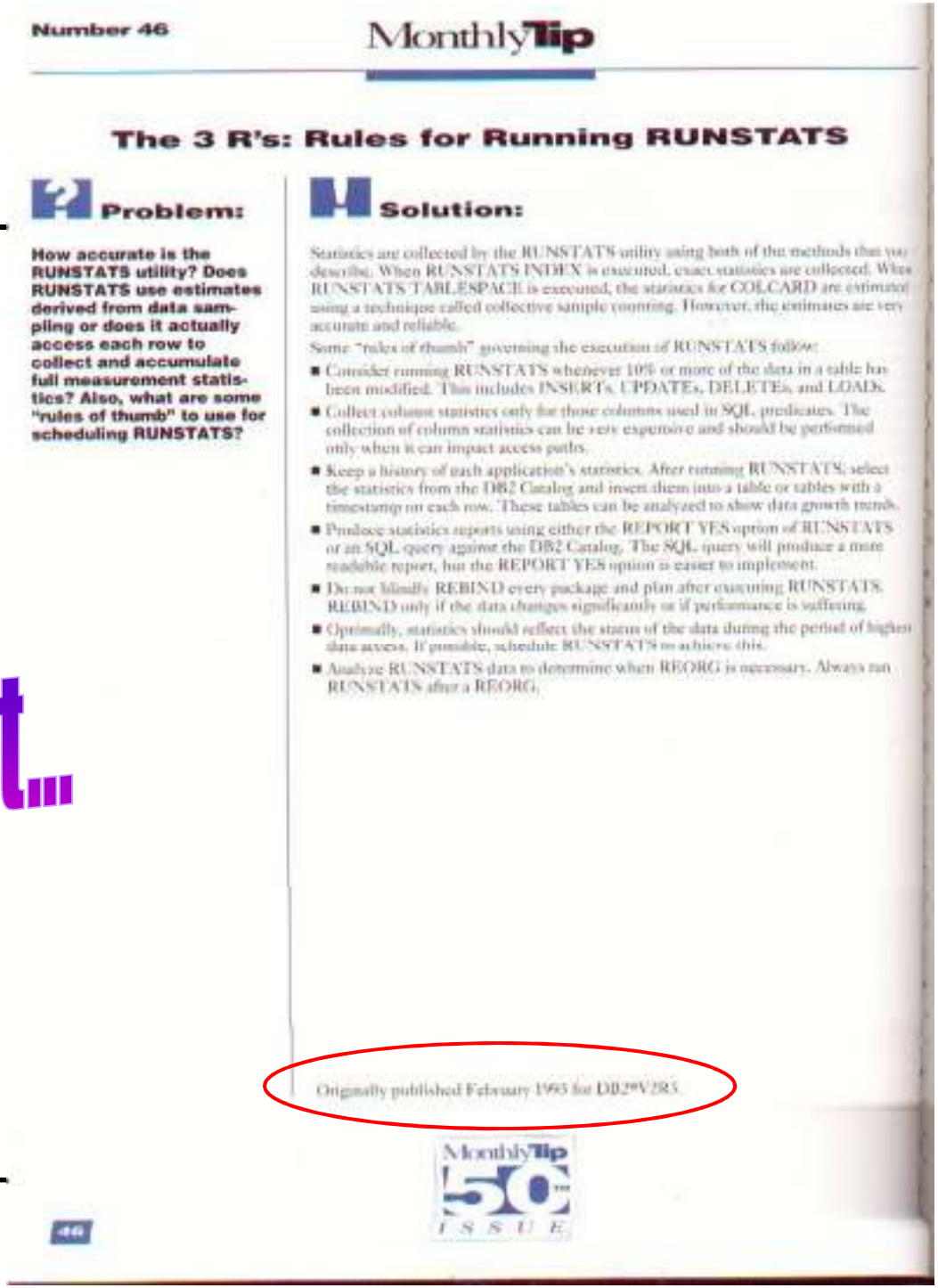
- The Three R's *(next slide)*
- Why is this better?
  - Access paths are more up-to-date based on the current state of the data.
- Result:
  - Generally, improved access paths
  - CPU savings from new DB2 efficiencies
  - Optimal performance
- Of course, you can still get those "problem" access paths.

The Three R's

- REORG
- RUNSTATS
- REBIND

Good, old advice but...

Still a couple of R's short.



Number 46

MonthlyTip

The 3 R's: Rules for Running RUNSTATS

Problem:

How accurate is the RUNSTATS utility? Does RUNSTATS use estimates derived from data sampling or does it actually access each row to collect and accumulate full measurement statistics? Also, what are some "rules of thumb" to use for scheduling RUNSTATS?

Solution:

Statistics are collected by the RUNSTATS utility using both of the methods that you describe. When RUNSTATS INDEX is executed, exact statistics are collected. When RUNSTATS TABLESPACE is executed, the statistics for COLCARD are estimated using a technique called collective sample counting. However, the estimates are very accurate and reliable.

Some "rules of thumb" governing the execution of RUNSTATS follow:

- Consider running RUNSTATS whenever 10% or more of the data in a table has been modified. This includes INSERTs, UPDATEs, DELETEs, and LOADs.
- Collect column statistics only for those columns used in SQL predicates. The collection of column statistics can be very expensive and should be performed only when it can impact access paths.
- Keep a history of each application's statistics. After running RUNSTATS, select the statistics from the DB2 Catalog and insert them into a table or tables with a timestamp on each row. These tables can be analyzed to show data growth trends.
- Produce statistics reports using either the REPORT YES option of RUNSTATS or an SQL query against the DB2 Catalog. The SQL query will produce a more readable report, but the REPORT YES option is easier to implement.
- Do not blindly REBIND every package and plan after executing RUNSTATS. REBIND only if the data changes significantly or if performance is suffering.
- Optimally, statistics should reflect the status of the data during the period of highest data access. If possible, schedule RUNSTATS to achieve this.
- Analyze RUNSTATS data to determine when REORG is necessary. Always run RUNSTATS after a REORG.

Originally published February 1993 for DB2®V2R3

MonthlyTip 50th ISSUE

46

Problems With the Three R's

Mullins Consulting, Inc.

They pose a lot of questions…

- When should you REORGanize?
  - To properly determine requires statistics (available in RTS).
  - So perhaps it should be RTS, REORG, RUNSTATS, REBIND?
- When should you run RUNSTATS?
  - To properly determine you need to know the make-up, usage, and volatility of your data.
- When should you REBIND?
  - When statistics have changed significantly enough to change access paths.
  - Knowing when this happens can be tricky.

The Importance of Accurate DB2 Catalog Statistics

Mullins Consulting, Inc.

Why correct statistics are so important

- The DB2 Optimizer makes all access path decisions
- Accurate stats help the Optimizer make the correct decisions
- Incorrect statistics tend to degrade performance due to bad access paths

"More than half of the bad access paths sent to IBM support are caused by incorrect statistics."

- According to Terry Purcell (IBM, SVL)

Getting Correct Statistics

Mullins Consulting, Inc.

Ways to update statistics

- RUNSTATS utility
- REORG with inline statistics
- LOAD with inline statistics
- Using SQL for statistics manipulation
- Transferring statistics from another system
- Using tools for manipulation

OK, Then…
When Should we REBIND?

Mullins Consulting, Inc.

When do we REBIND?

- The best answer to this questions is: "Whenever data has changed significantly enough that it may impact the performance of the existing access paths."
  - The problem is knowing *exactly* when this happens.

DB2 application performance can be negatively affected by uncontrolled REBINDs.

- Causes
  - Optimizer inefficiency
  - Volatile tables
  - Catalog pollution
  - Inefficient use of RUNSTATS

So The Best Approach: The ~~3~~ 5 R's

Mullins Consulting, Inc.

RTS **(or RUNSTATS)**

REORG

RUNSTATS

REBIND

Recheck

- In other words, what did the REBIND do?
  - Did any access paths change?
  - Are they better or worse?
  - Does anything need attention?

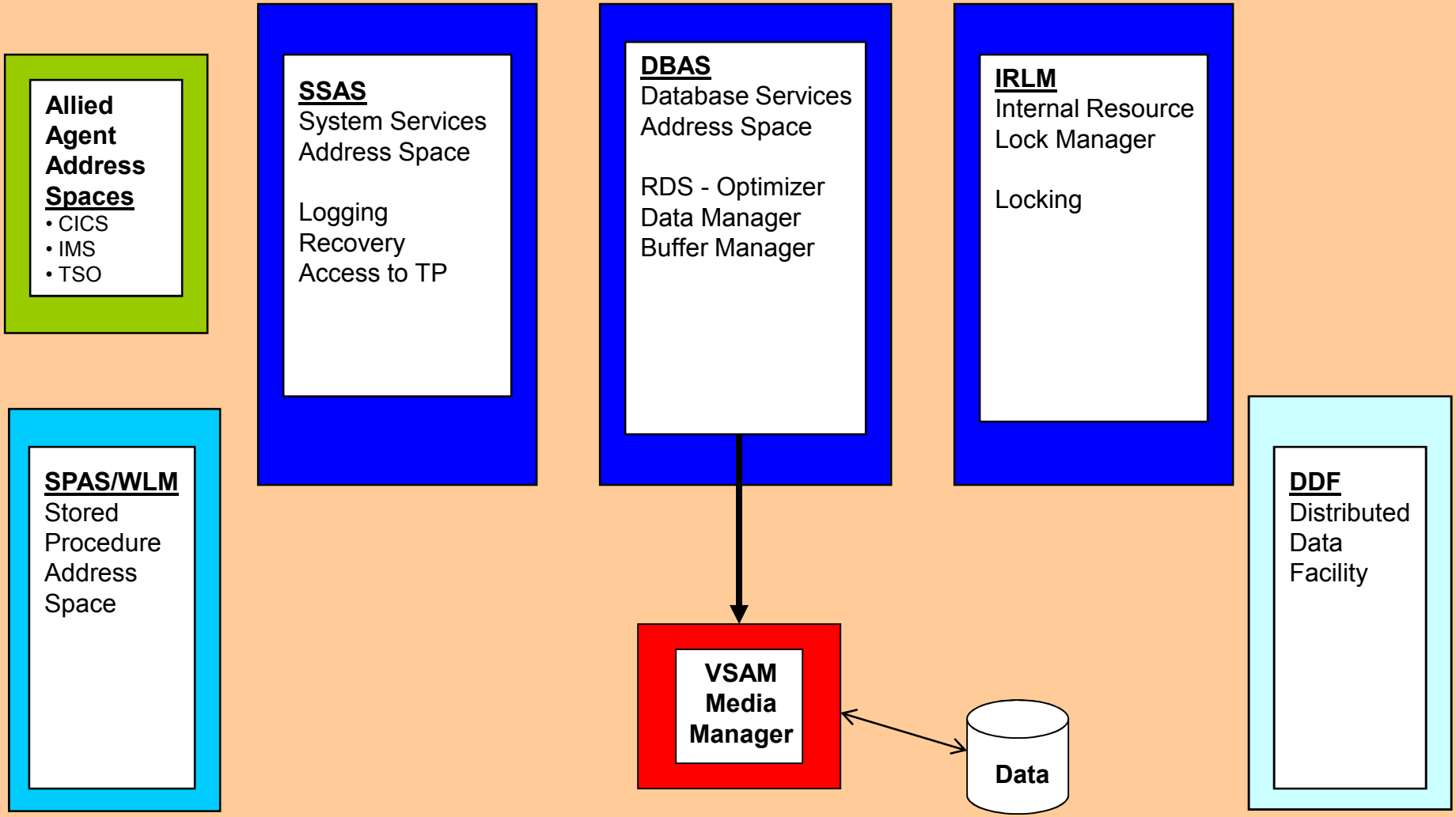
Access Path Degradation Correction

Mullins Consulting, Inc.

Problem access paths can still occur. If so:

- Absolutely determine you are not at fault by re-re-checking
  - Statistics up-to-date?
  - Correct statistics run?
  - Package rebound with latest statistics?
- If problems persist, one of the following approaches could work for you:
  - Plan Stability (V9)
  - Tweak SQL (ways to "coerce" the optimizer)
  - Code and Use an Access Path Hint
  - Manipulate statistics (*caution*)

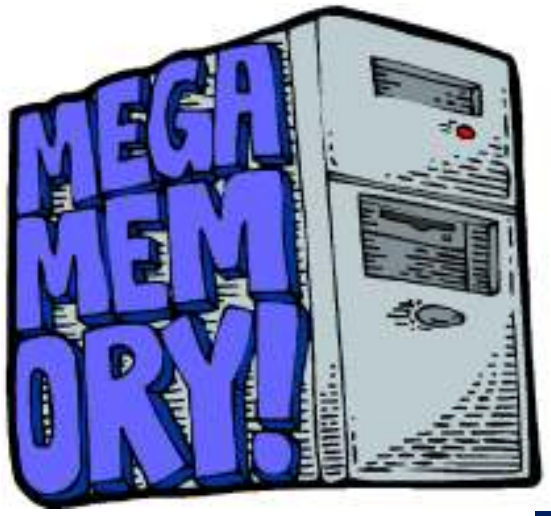
System & DB2 Subsystem Tuning



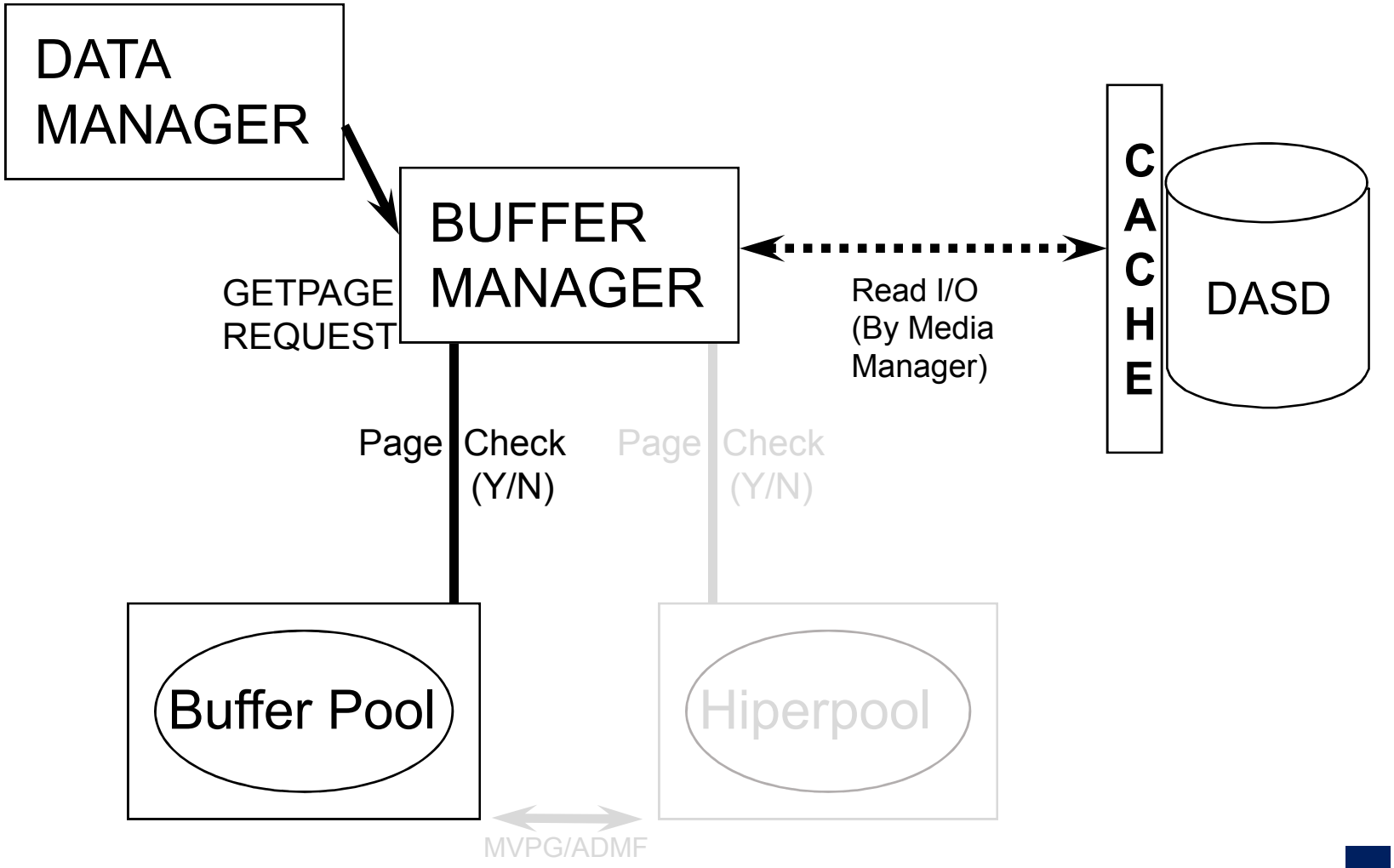
Memory

Relational database systems “love” memory

- Performance improves if important information and run-time details are cached in memory instead of being read from disk every time they are needed.



GETPAGE Processing



Swimming in DB2's " Pools"

Mullins Consulting, Inc.

DB2 uses four types of "pools" - or memory structures to cache data and information to avoid costly disk I/O

- **Buffer Pools** - used to cache data in memory when it is read from disk.
- **RID Pool** - used to sort RIDs (record identifiers) for List Prefetch, Multiple Index Access, and Hybrid Joins.
- **EDM Pool** - used to cache program details (access paths, dynamic PREPARE, authorization) and database structural information (DBD).
- **Sort Pool** - used when DB2 must sort data.

Subsystem: Buffer Pools, Pt. 2

Mullins Consulting, Inc.

DB2 provides up to 80 buffer pools - USE THEM!

- 4K: BP0 thru BP49
- 8K: BP8K0 thru BP8K9
- 16K: BP16K0 thru BP16K9
- 32K: BP32K thru BP32K9

Consider reserving a bufferpool for tuning

- Move problem objects there to isolate for tuning

DB2 V8 significantly increased buffer pool storage

- 1TB new limit for buffer pools
- No more hiperpools; no more bufferpools in data spaces

Monitor hit ratio: % times a page is found in the buffer pool

- The higher the ratio the better

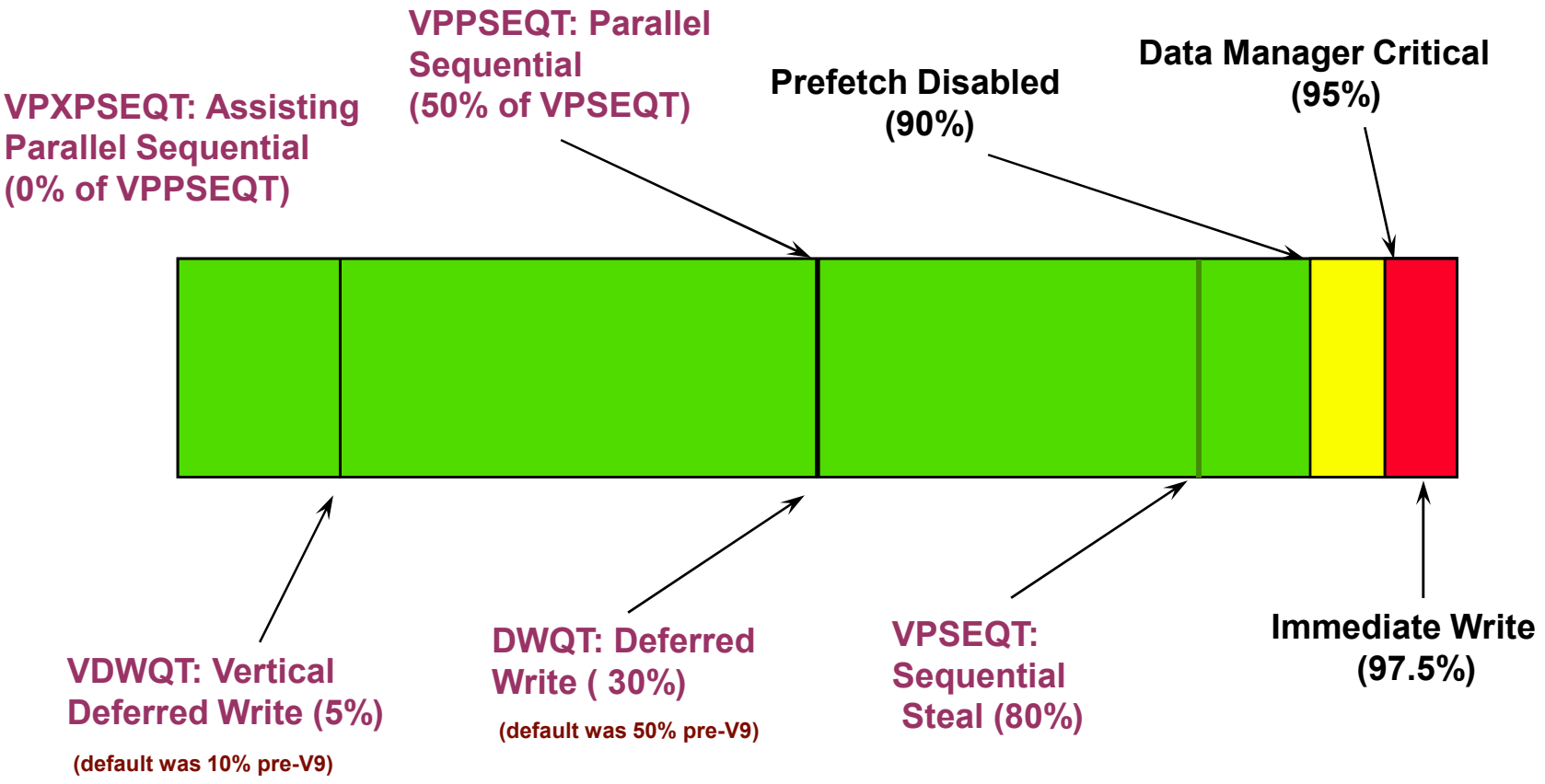
(GETPAGES – PAGES READ) / GETPAGES

SYNC I/O + ASYNC I/O

Buffer Pools: Tune Thresholds

Mullins Consulting, Inc.

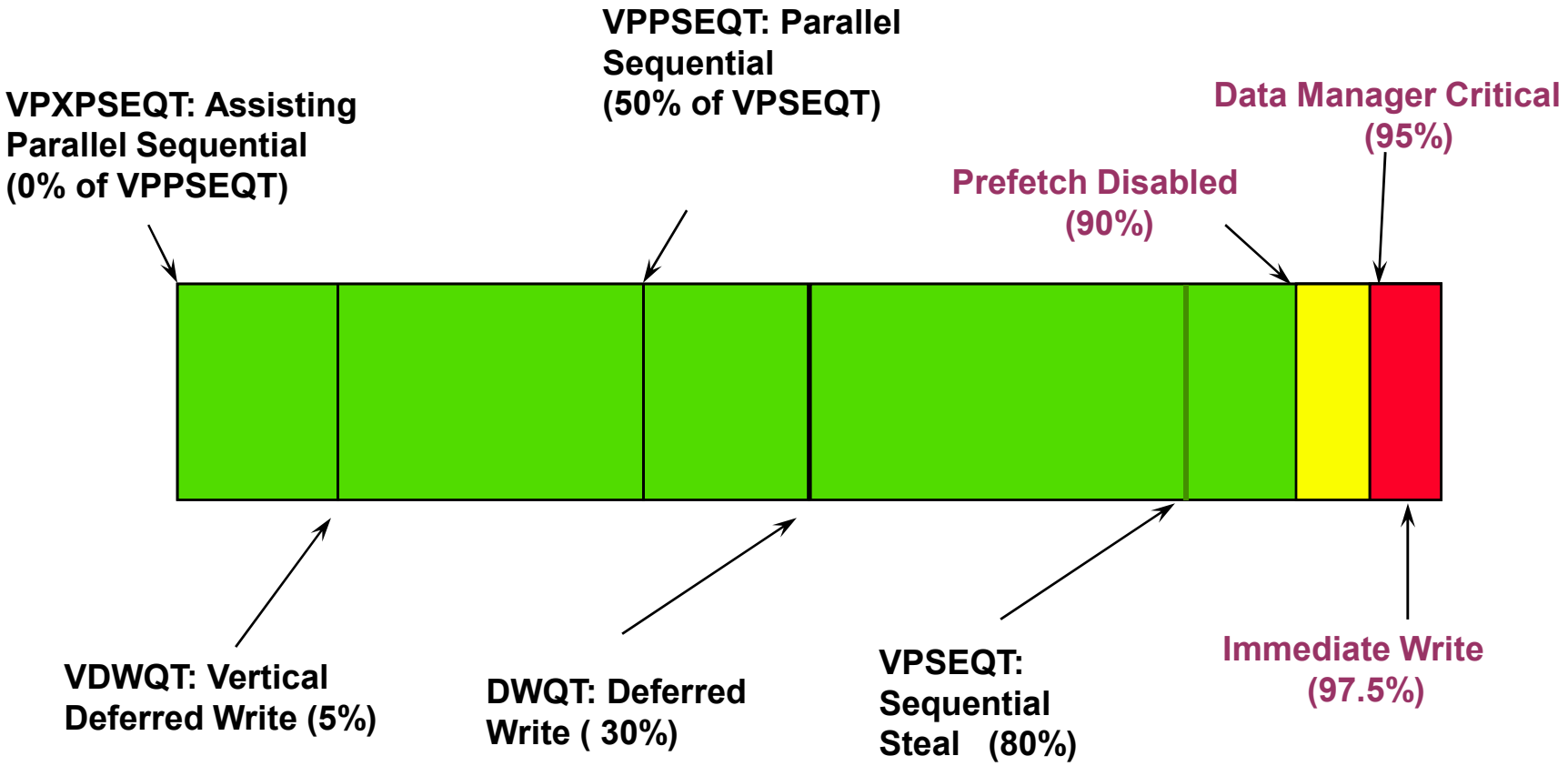
Variable Thresholds



Buffer Pools: Monitor Thresholds

Mullins Consulting, Inc.

Fixed Thresholds



Subsystem: RID Pool

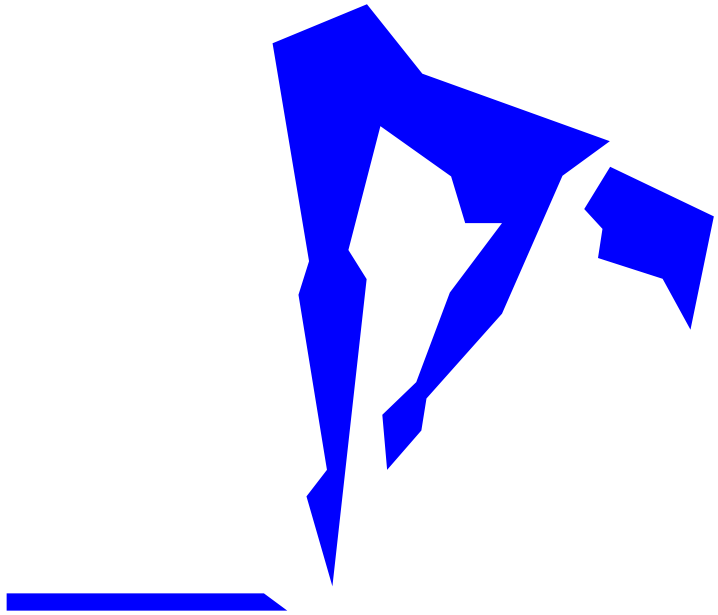
Mullins Consulting, Inc.

One RID pool for all processing.

The default RID pool size is 4 MB. Can be changed.

RID Pool is used for:

- enforcing unique keys while updating multiple rows
- sorting RIDs during the following operations:
  - List prefetch, including single index list prefetch
  - Access via multiple indexes
  - Hybrid Joins



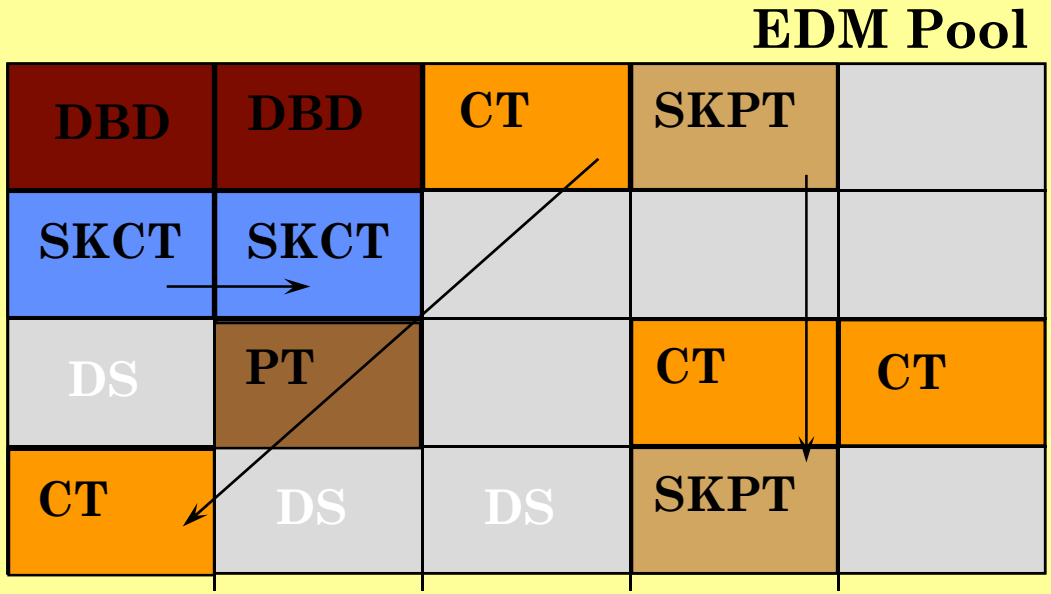
RID Pool Problems?

Mullins Consulting, Inc.

1. RID Pool Overflow (no storage)
   - **Requests exceed ZPARM or DBM1 address space size**
   - **Limit: More than 16 million RID entries used or a single SQL statement consumes more than 50% of the RID Pool**
     - **The SQL causing this condition receives a -904.**
     - **Should not occur frequently.**
2. DB2 anticipates RID pool access is not economical (RDS Limit)
   - **Processing is suspended and access degrades to a table space scan**
     - **DB2 10 changes this behavior; write RIDs to disk instead of degrading to TS scan**
   - **You can disable the access paths listed above by specifying a RID pool size of 0.**
     - **Don't forget to REBIND to change access paths requiring RID pool**

Subsystem: EDM Pool

DB2 Database Services Address Space



What's in EDM Pool

- DBDs
- SKCTs
- CTs
- SKPTs
- PTs
- Auth Cache
- Dyn SQL Prep
- Free pages

V8 breaks each out into separate "pools"

Further break out in V9

General ROT: shoot for 80% efficiency; (1 in 5 DBD/SKPT/SKCT needs to be loaded)

The EDM Pool and V8, V9

Mullins Consulting, Inc.

V8: EDM Pool split into three specific pools:

- Below the 2GB Bar
  - EDMPOOL: EDM Pool stores only CTs, PTs, SKCTs, SKPTs
    - Should be able to reduce the size of this EDM pool
    - Provide some VSCR for below the 2GB Bar storage
- Above the 2GB Bar
  - EDMDBDC: DBDs
  - EDMSTMTC: Cached Dynamic Statements

V9: Introduces additional changes

- Above the 2GB Bar: EDM_SKELETON_POOL
  - All SKCTs and SKPTs
- A portion of the CT and PT is moved above the bar, too

Subsystem: Sort Pool

Mullins Consulting, Inc.

Sort Pool value is the maximum size of the sort work area allocated for each concurrent sort user.

The default Sort Pool size is 2 MB.

- It can be changed on install panel DSNTIPC.

In general, estimate the required storage for a sort pool using the following formula:

```
32000 * (12 + sort key length + sort data length + 4))
```

Sorts that don't fit in SORTPOOL overflow to workfile

- DSNDB07 for non-Data Sharing systems

Sort Performance

Mullins Consulting, Inc.

- In general, the larger the sort pool, the more efficient the sort is.
  - If the data fits into the sort pool, workfile database will not be required
- Allocate additional physical work files in excess of the defaults, and put those work files in their own buffer pool (e.g. BP7).
  - At least 5, sized the same, with no secondary
- The better sorted the data is originally, the more efficient the sort will be.
- Minimize the amount of data that needs to be sorted! →

Minimize Amount of Data to Sort

Mullins Consulting, Inc.

DB2 uses a tournament sort *(next page)* **unless…**

If Sorted Record > 4075, uses a tag sort *(less efficient)* **because the data no longer fits on a 4K page:**

- Data to sort put directly into 32K workfile database
  - For this reason be sure to always allocate at least one 32K workfile in DSNDB07
- Keys + RID are sorted
- Data retrieved from the sort using the RID

Tournament Sort

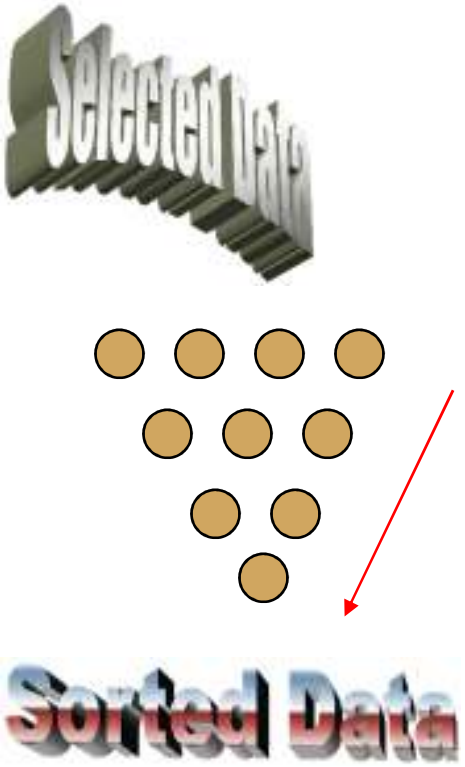
Mullins Consulting, Inc.

DB2 uses a Tournament Sort

- Built into the hardware microcode and very efficient

How Does it Work?

- Input data to be sorted passes through a tree structure
- At each level in the tree the data is compared to data already there
- The 'winner' (lowest value for an ASC) moves up the tree
- At the top of the tree, the sorted entries are placed into runs
- Winning entries are removed from the tree and the next value inserted
- If there is more than one run, the runs must be merged



Subsystem: Logging

Mullins Consulting, Inc.

DB2 will only run as fast as the log

Log Configuration

- Dual Active Logging is the preferred configuration
- Each log defined to separate devices and on separate channels

Output Buffer Size

- As BIG as possible please
- Waits occur if OUTBUFF is too small
- Max is 400000K

DB2 rollbacks from log data on DASD

- Consider keeping archive logs on DASD*

* and then migrate archive logs to tape after a specified period of time (HSM)

Subsystem: System Checkpoint

Mullins Consulting, Inc.

Periodically DB2 takes a checkpoint, containing:

- currently open unit of recoveries (UR) within DB2, all open page sets, a list of page sets with exception states, and a list of page sets updated by any currently open UR
- Dirty pages are written out at checkpoint and processing stops until they are written – so make sure DWQT is sized correctly!

Specified in the CHKFREQ* parameter in DSNZPARM

- Number of log records written
- Or, as of V7, number of minutes

Can be changed dynamically using:

- SET LOG or *(temporary)*
- SET SYSPARM (as of V7) *(permanent)*

5 minute intervals for checkpoints during peak processing times.

*CHKFREQ replaced LOGLOAD in DB2 V7

Subsystem Tuning: IRLM

Mullins Consulting, Inc.

MAXCSA

- 250×(LOCKS PER USER)×(MAX USERS)

250 bytes of storage for each lock.

ITRACE=NO

- Do not use ITRACE;
- Instead, if needed, use DB2 lock traces.

DEADLOK

1. The number of seconds between two successive scans for a local deadlock
2. The number of local scans that occur before a scan for global deadlock starts

Consider Specialty Processors: zIIP

The zIIP (System z Integrated Information Processor)

- Transparently (to the application) redirect portions of DB2 distributed (mostly) workload.
  - Remote DRDA access, including JDBC and ODBC access to DB2, including access across LPARs using Hipersockets;
    - Including native SQL stored procedures that are run through DDF (DB2 V9);
  - BI application query processing utilizing DB2 star-schema parallel query capabilities;
  - XML parsing (DB2 V9) and;
  - Certain IBM DB2 utility processing that performs maintenance on index structures.
    - The BUILD portion of LOAD, REORG, and REBUILD

Environment

Operating System

- version, memory, JCL, RACF, etc.

TP Monitors

- CICS, IMS/TM, C/S GUI, web, etc.

Networking

- TCP/IP, SNA, DRDA, stored procedures, etc.

DASD

- storage, ESS/Shark, placement, etc.

Summary

Application

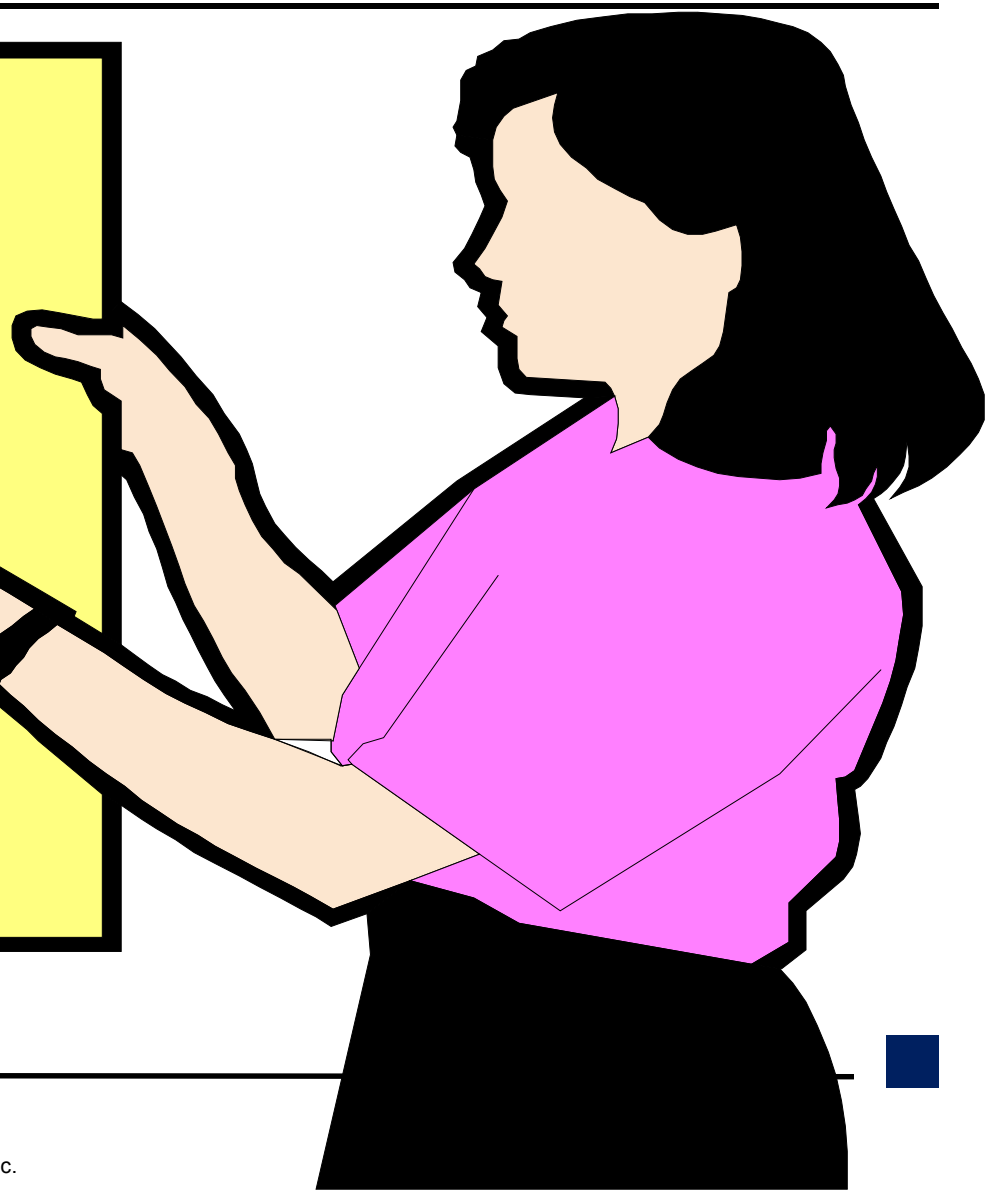
Database

DB2 Subsystem

Environment

Do one thing at a time and;

You ***can*** tune DB2!



Contact Information

Mullins Consulting, Inc.

http://www.craigsmullins.com/dba_book.htm

http://www.craigsmullins.com/cm-book.htm

Craig S. Mullins

Mullins Consulting, Inc.
15 Coventry Court
Sugar Land, TX 77479

http://www.craigsmullins.com

craig@craigsmullins.com

Phone: (281) 494-6153