Best Practices for Optimizing DB2 Performance

datAvail

Author

This presentation was prepared by:

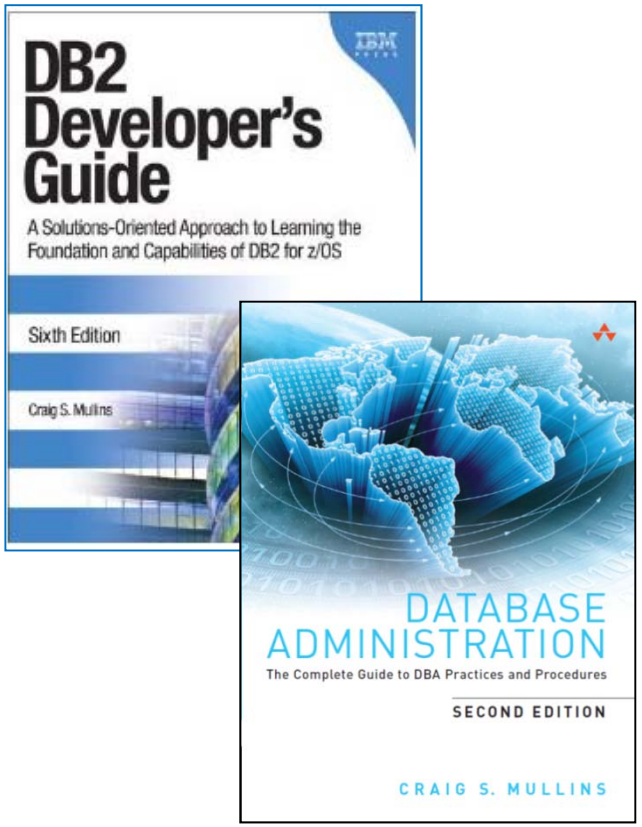
Craig S. Mullins
President & Principal Consultant
Mullins Consulting, Inc.

and

DB2 Practice Consultant
Datavail Corporation

E-mail:
craig@craigsmullins.com

craig.mullins@datavail.com



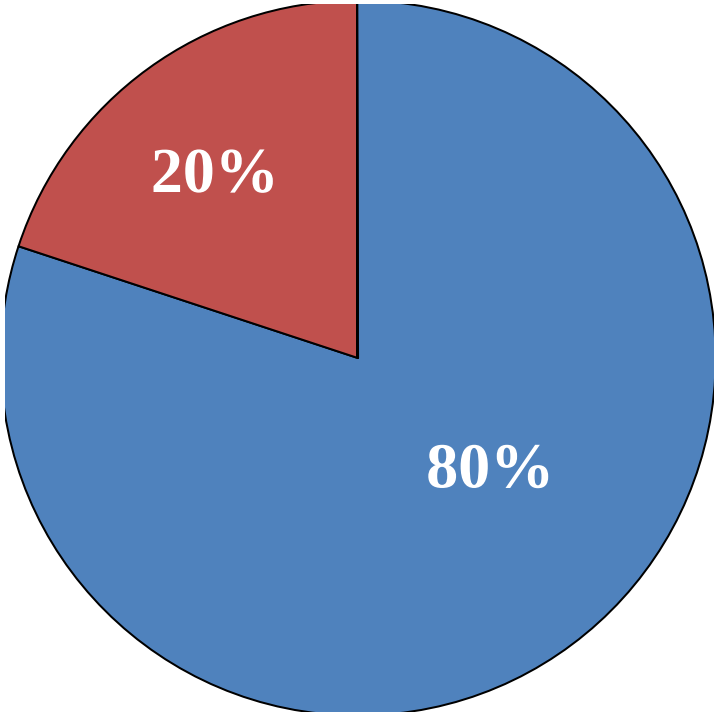
The Tuning Progression

Problem Resolution

- **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

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



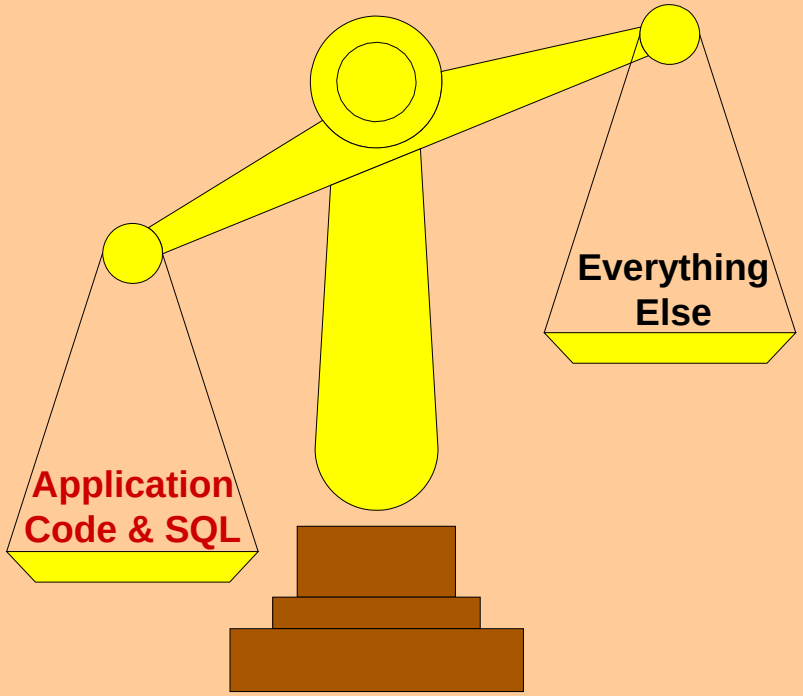
A Few General Performance Themes to Remember

- *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.
- And always remember the Cardinal Rule → "It Depends!"
  - It depends because relational database performance is a complex thing that is not simply explained: situations, implementations & requirements vary.
  - But the secret to optimizing DB2 performance is being able to answer the follow-up question to "It Depends," which is "What does it depend upon?"

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...

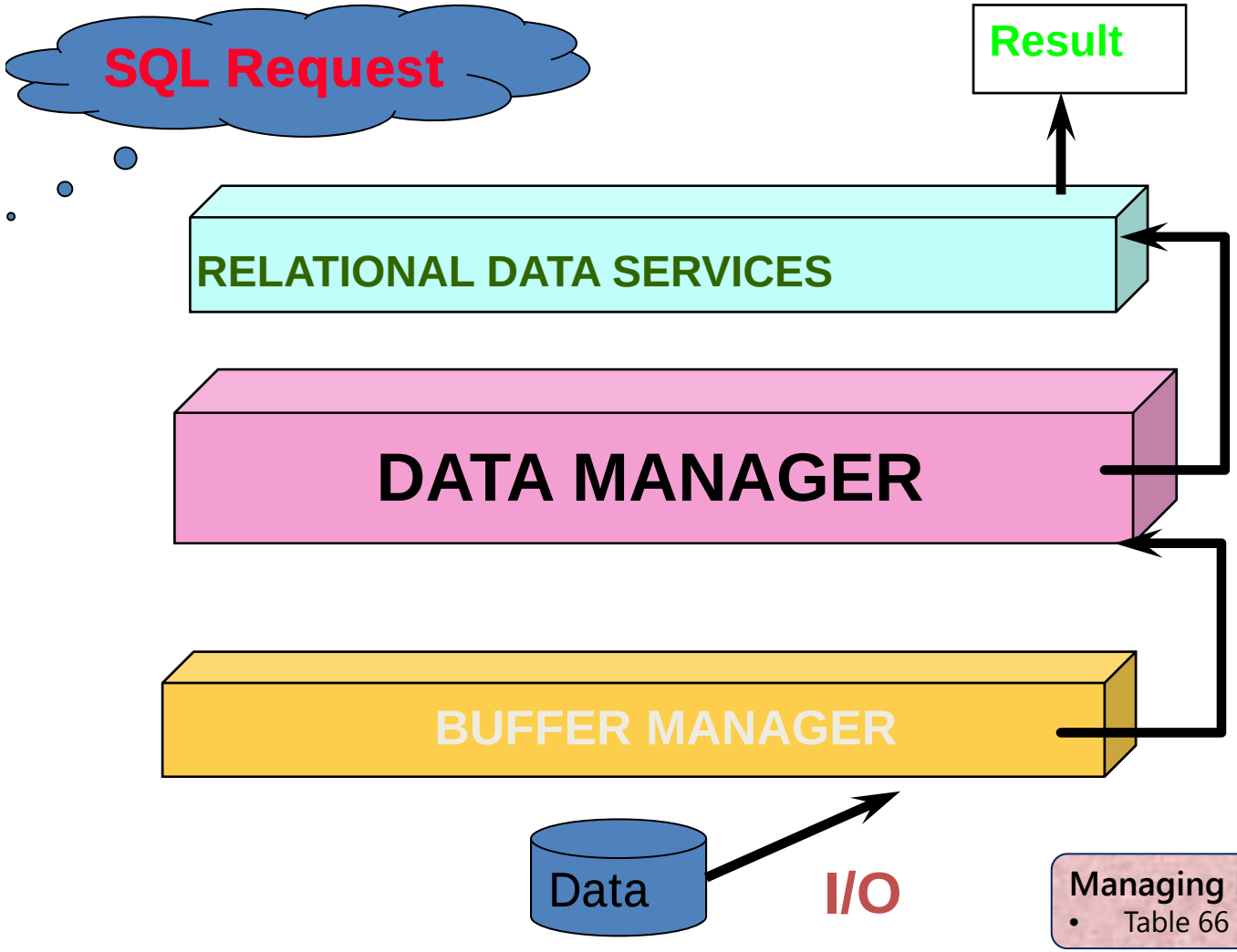
- *as high as 70% to 80%*



Application Tuning: SQL

- Simpler is better, but complex SQL can be efficient
- In general, let SQL do the work, not the program
- Always provide join predicates
  - Avoid Cartesian products
- Favor Stage 1 and Indexable predicates
  - Host variable data type/length should match column
- Avoid table space 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?

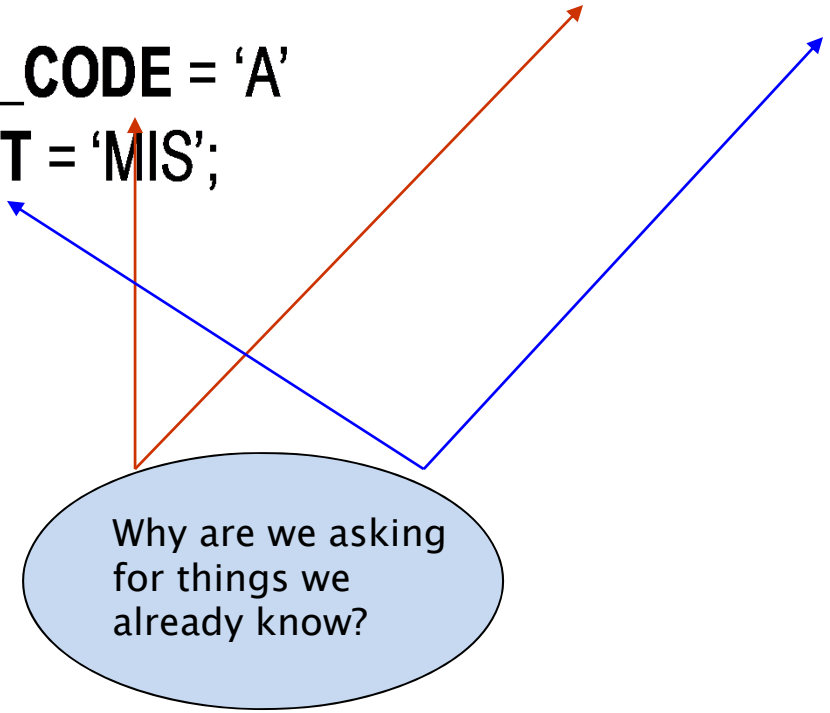
- **Not an actual Stage but…**
  - It can be helpful to think of moving predicates from SQL into your programs as Stage 3
- **Easy to remember:**
  - Stage 1 better than Stage 2 and…
  - Stage 2 better than Stage 3

Ask Only for What You Absolutely Need

- **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';
```

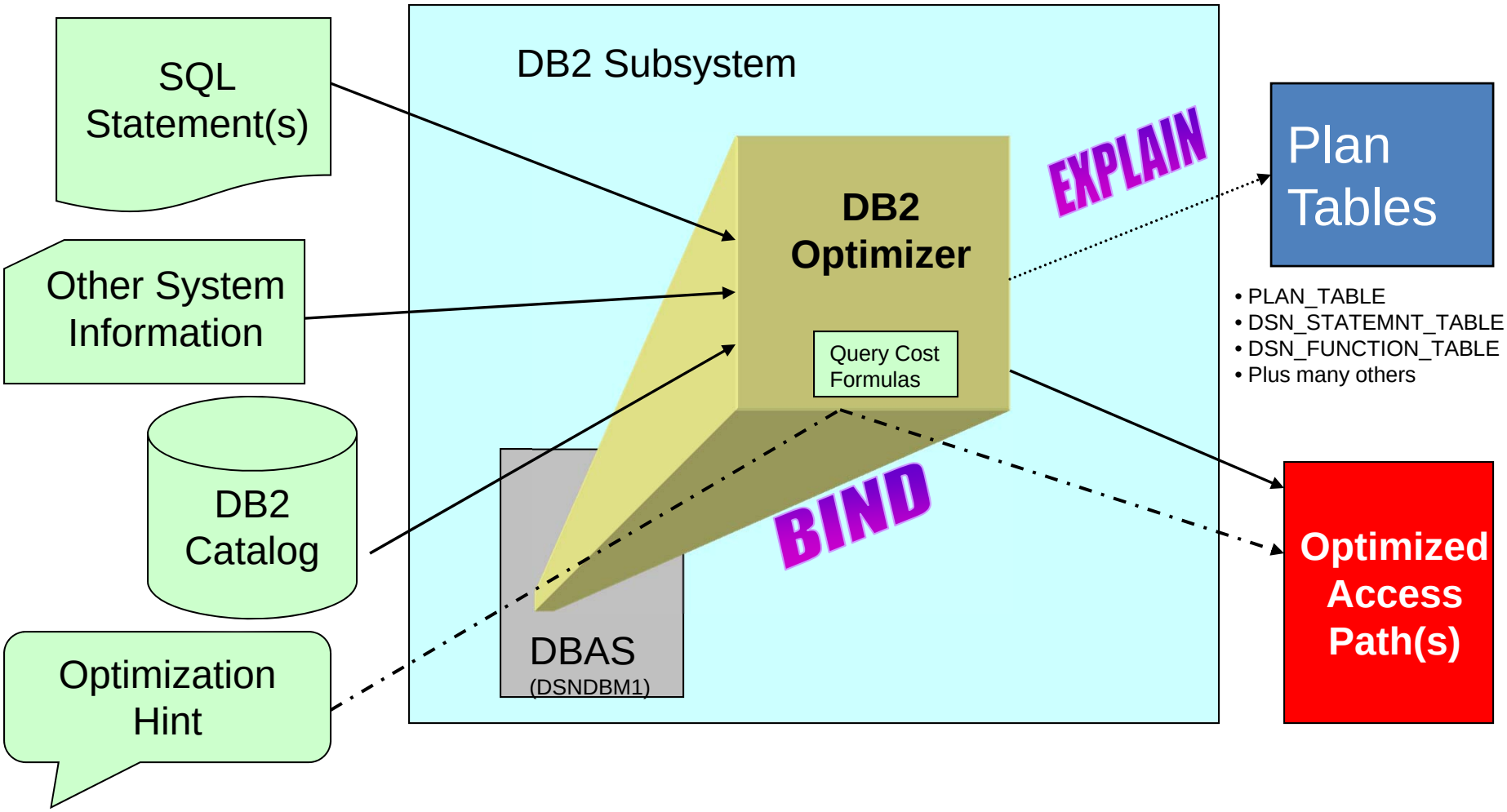


Avoid Black Boxes

Stage 4?

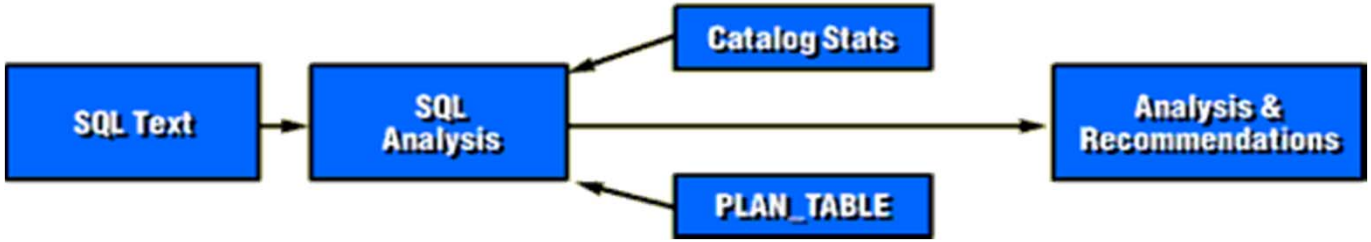
http://www.craigsmullins.com/dbu_0703.htm

Application Tuning: Optimization



OPTHINT in PLAN_TABLE

Application Tuning: *EXPLAIN Analysis*



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

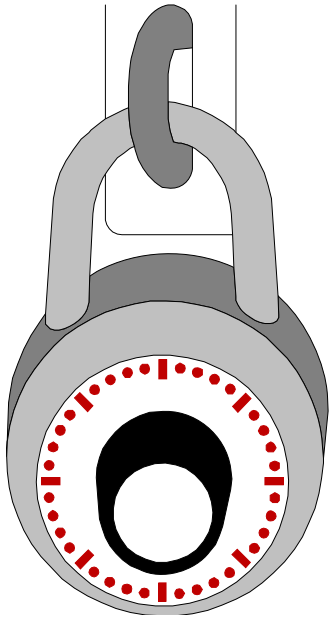
Sort required?

- Join, Unique, Group By, Order By

Locking

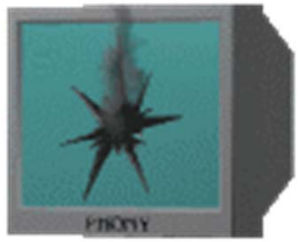
Application Tuning: Locking

- 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
- Plan and implement a COMMIT strategy
  - or experience TIMEOUTs and DEADLOCKs
- Encourage Lock Avoidance
  - ISOLATION(CS) / CURRENTDATA(NO)
  - Can be used only by read only cursors
- Consider ISOLATION(UR) to avoid locking

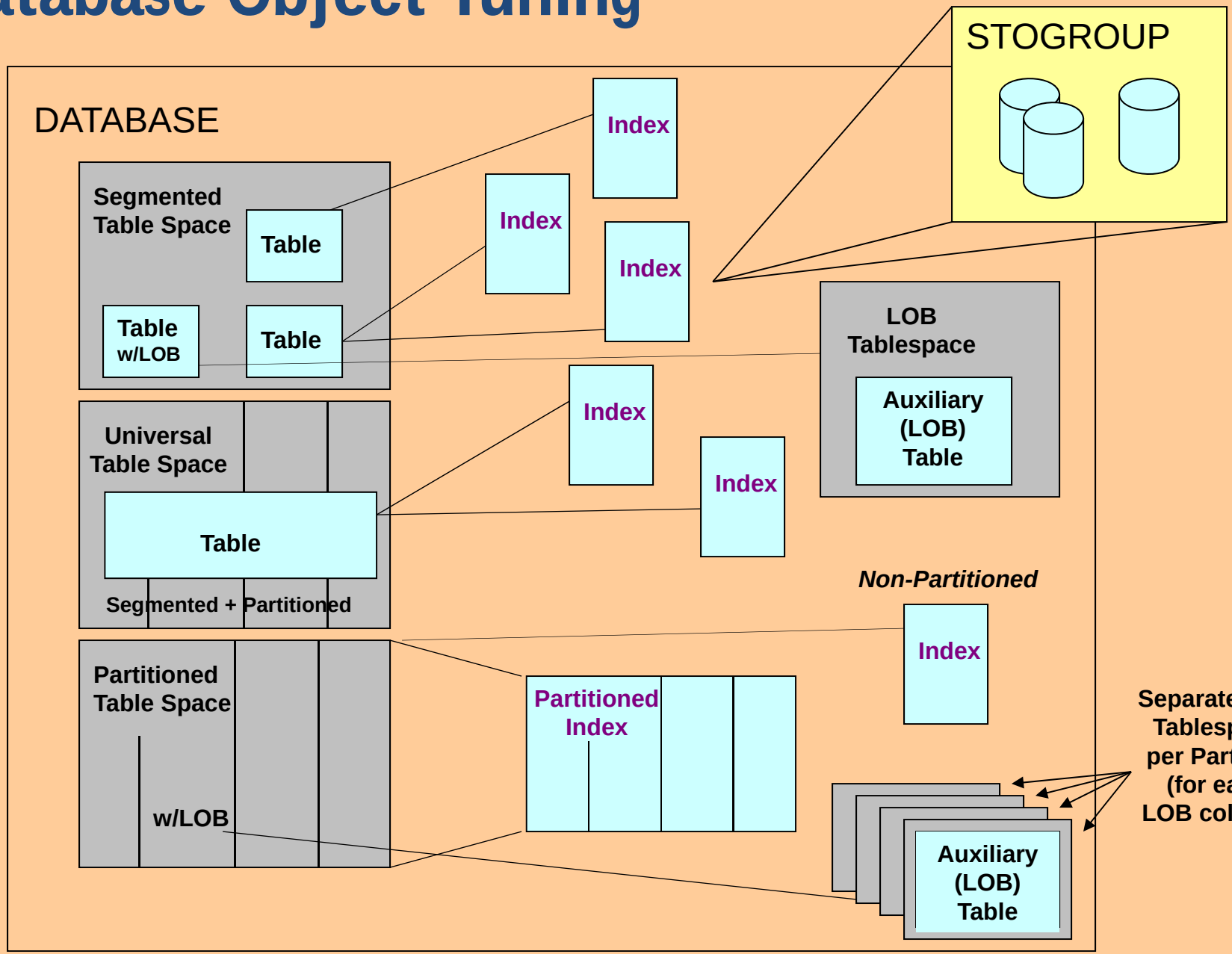


Application Tuning: Program

- **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
- **Online versus Batch**
  - Limit the amount of data to be retrieved to a reasonable amount
  - Consider OPTIMIZE FOR 1 ROW to disable list prefetch



Database Object Tuning



General Table Space Recommendations

- **Universal table spaces are the future (and present) of DB2 table spaces**
  - Favor using universal table spaces over segmented or traditional partitioned table spaces
    - Partition by growth (PBG) is ideal when a table is expected to exceed 64 GB but there is no suitable partitioning key
    - PBR, or range-partitioned, universal table spaces are ideal for partitioning data and should be considered the standard instead of classic partitioning
  - At some point, other table space types are likely to be deprecated (like simple table spaces, which 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

Database Organization

- **Be sure to run RUNSTATS**
  - as data volume changes, new data structures added
  - followed by (RE)BIND with /EXPLAIN(YES)
- **Review statistics (or better yet 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

Scheduling Regular Rebinds: The Five R's

- **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

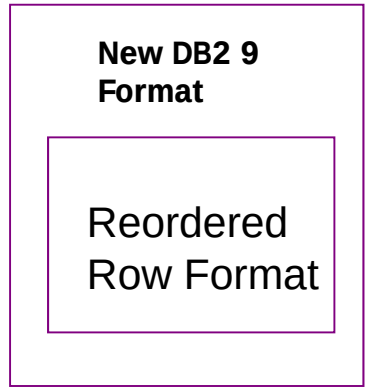
- **Problem access paths can still occur. If so:**
  - Absolutely determine you are not at fault by re-re-checking
    - Statistics up-to-date?
    - Correct statistics run?
    - Package rebound with latest statistics?
  - If problems persist, one of the following approaches could work for you:
   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

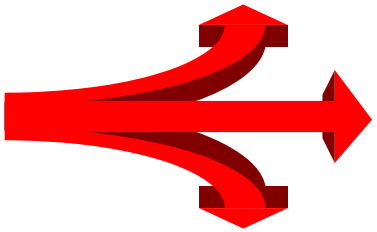
Table Design Guidelines

- As normalized as possible, but performance before aesthetics; normalization optimizes "update" at the expense of "retrieval"
  - Don't let data modelers dictate "physical" design
- 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
  - Don't just let everything default (for example, to 10).
    - Should be based on the volatility of the data
- Use appropriate data types
  - Character, numeric, date-time, binary, XML, BLOBs…
- Compression versus VARCHAR
  - Compression = less overhead *(no 2 byte prefix)*
  - Compression requires no programmatic handling

Database Design: Columns and RI

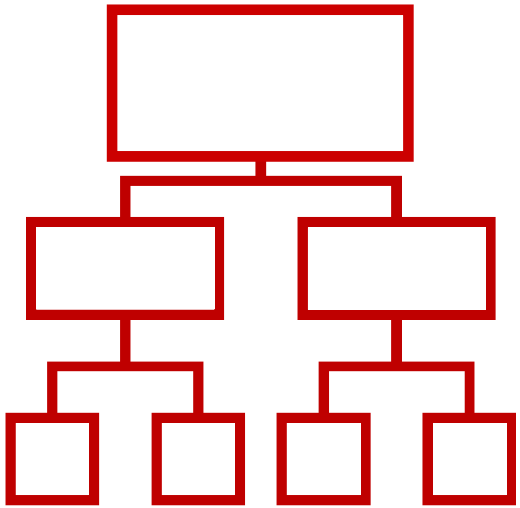
- **Sequence columns based on logging**
  - Infrequently updated non-variable columns first
  - Static (infrequently updated) variable columns
  - Frequently updated columns last
  - Frequently modified together, place next to each other
- **Use DB2 declarative RI instead of program RI *(usually)***
  - Declarative usually outperforms program RI
  - Ensures integrity for both planned and ad hoc database modification
  - Do not use RI for lookup tables *(overkill)*
    - consider CHECK constraints vs. lookup tables
  - Specify indexes on foreign keys



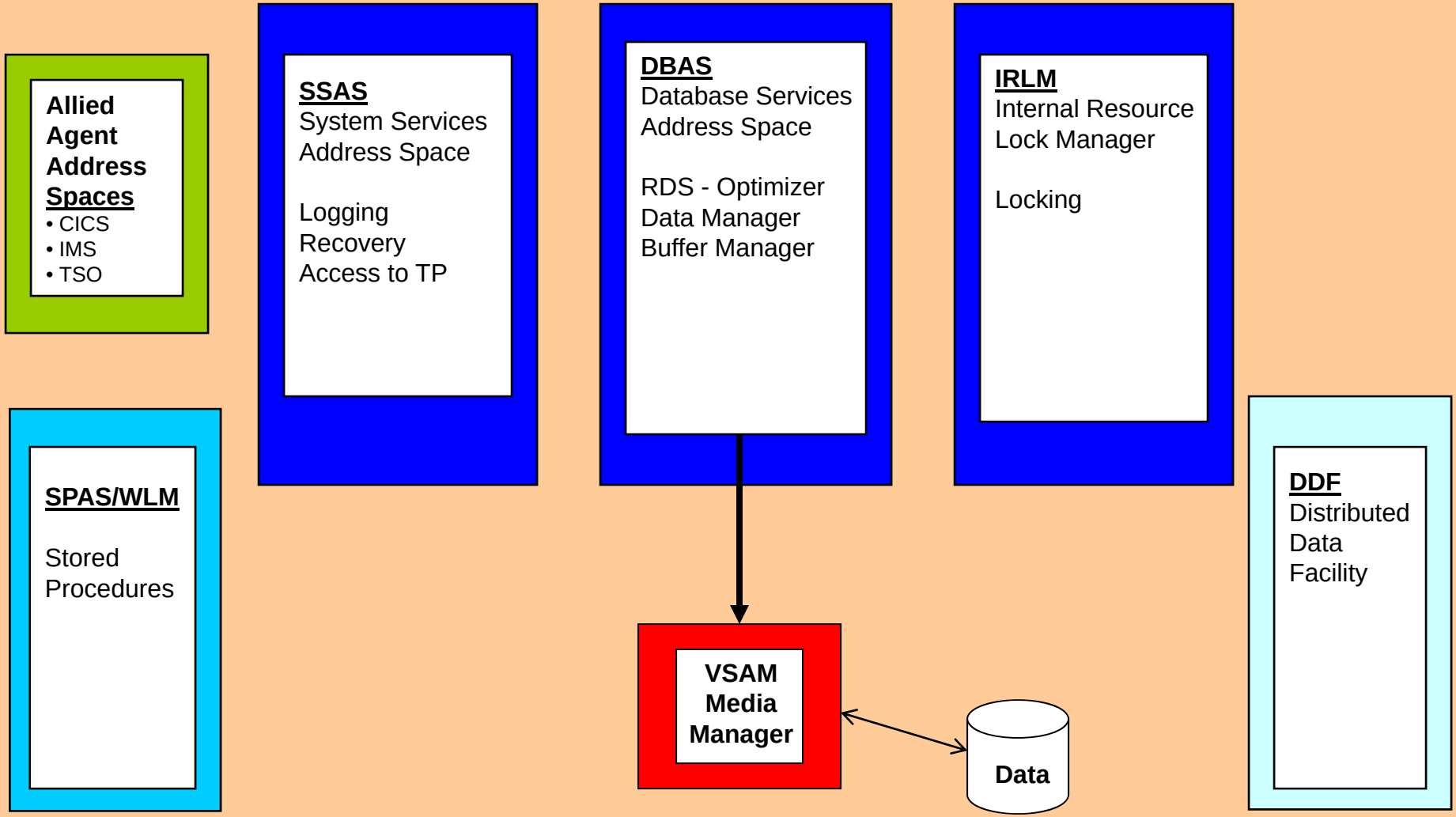


Database Design: Indexes

- 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

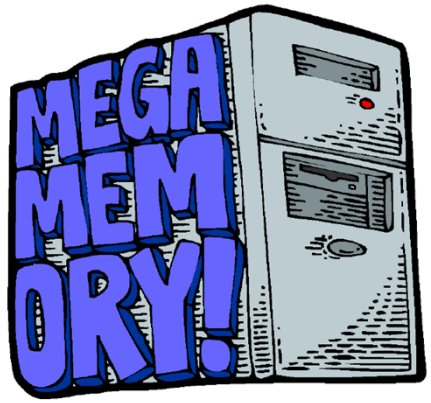


System & DB2 Subsystem Tuning



Swimming in DB2's " Pools"

- 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.
- 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 Tuning: Buffer Pools

- DB2 provides up to 80 buffer pools - USE THEM!
  - - 4K: BP0 thru BP49
  - - 8K: BP8K0 thru BP8K9
  - - 16K: BP16K0 thru BP16K9
  - - 32K: BP32K, and BP32K1 thru BP32K9
- Consider reserving a buffer pool for tuning
  - Move problem objects there to isolate for tuning
- Monitor hit ratio: % times a page is found in the buffer pool
  - The higher the ratio the better

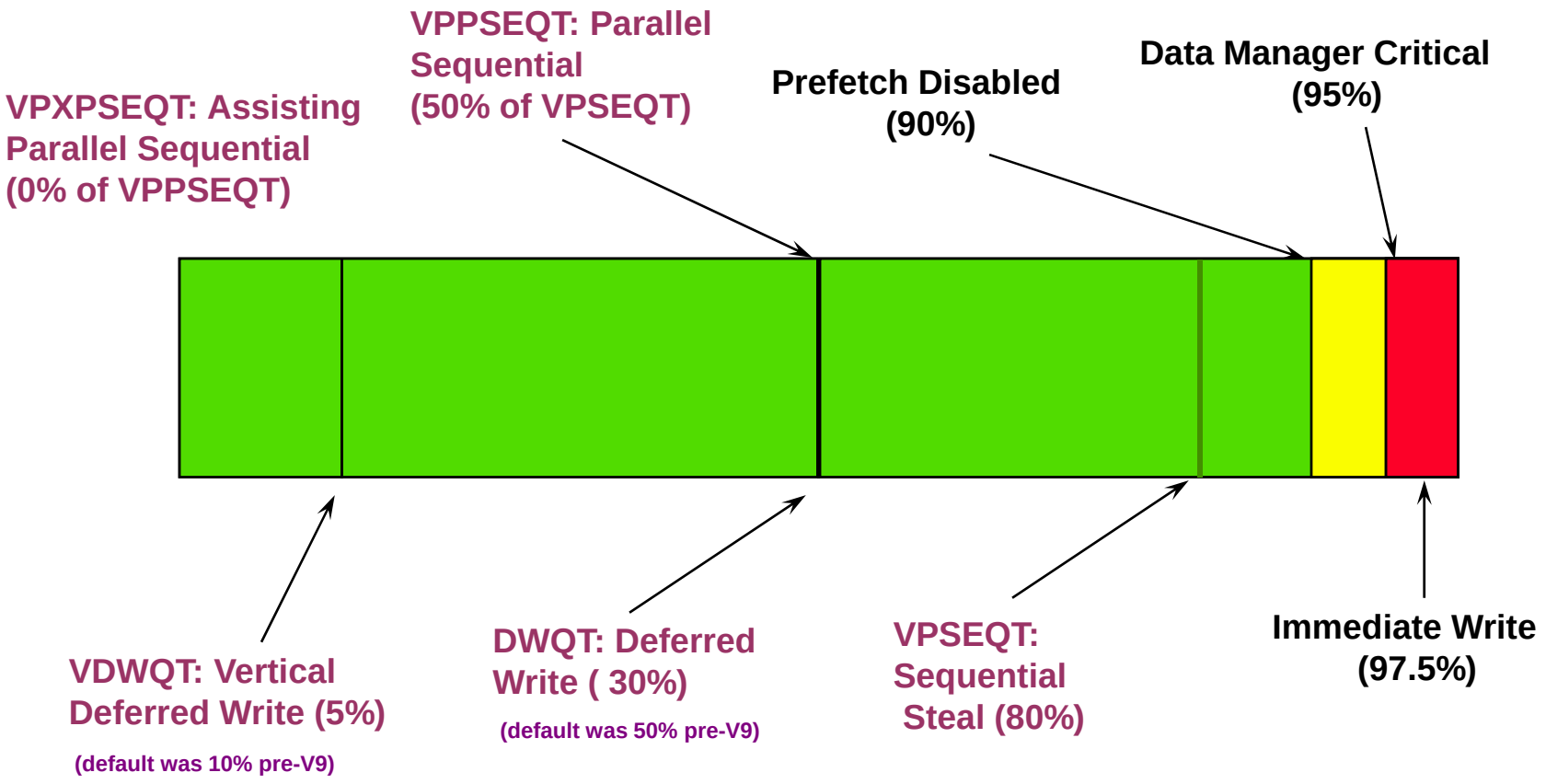
(GETPAGES – PAGES READ) / GETPAGES

→ SYNC I/O + ASYNC I/O

- Monitor and tune buffer pools thresholds
  - Variable and Fixed

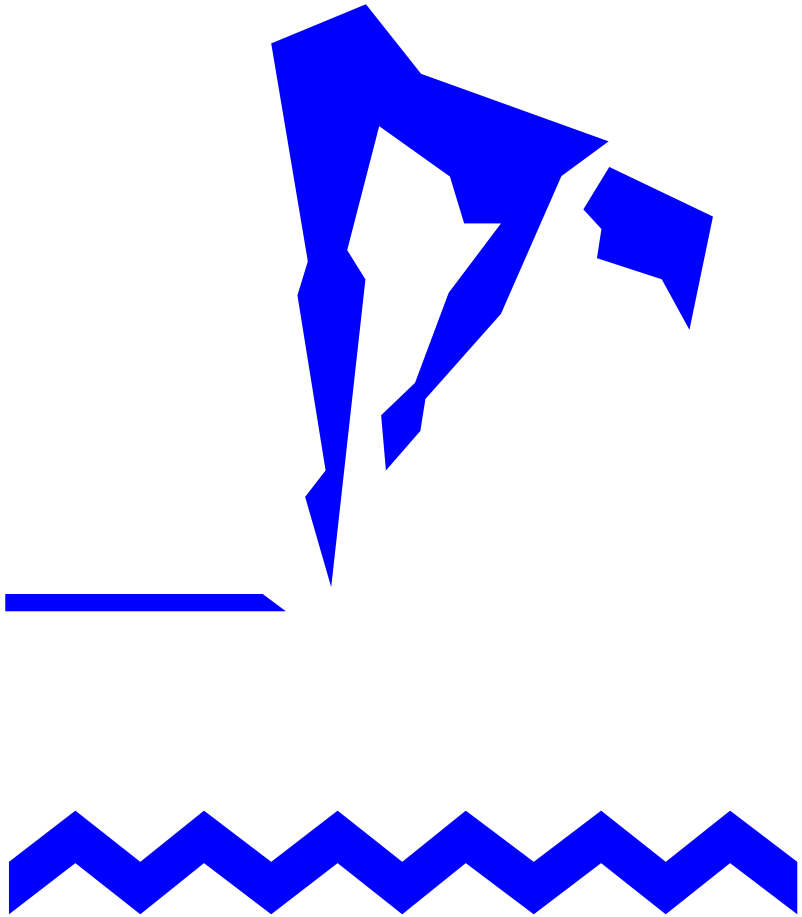
Buffer Pools: Tune Thresholds

- *Variable Thresholds*
- *Fixed Thresholds*



RID Pool Overview

- 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



The EDM Pool: Contents & Changes

- **What's in the EDM Pool**

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

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

- **V8: EDM Pool split into three specific pools:**
  - Below the 2GB Bar
    - **EDMPOOL**: EDM Pool stores only CTs, PTs, SKCTs, SKPTs
      - Provides VSCR 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

Sort Pool and Sorting Guidelines

- Sort Pool value is the maximum size of the sort work area allocated for each concurrent sort user.
- Minimize the amount of data that needs to be sorted!
  - DB2 uses a tournament sort unless the Sorted Record > 4075, in which case it uses a tag sort (which is less efficient)
  - Keys + RID are sorted… the data is retrieved from the sort using the RID
- Sorts that don't fit in the Sort Pool overflow to workfile
  - In general, the larger the sort pool, the more efficient the sort.
- 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.

Subsystem: Logging

- 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
- DB2 rollbacks from log data on disk
  - Consider keeping archive logs on disk*

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

Subsystem: System Checkpoint

- **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 (CHKLOGR)
  - Or, as of V7, number of minutes (CHKMINS)
- **Can be changed dynamically using:**
  - SET LOG or *(temporary)*
  - SET SYSPARM *(permanent)*

5 minute intervals for checkpoints during peak processing times.

*CHKFREQ replaced LOGLOAD in DB2 V7

Subsystem Tuning: IRLM

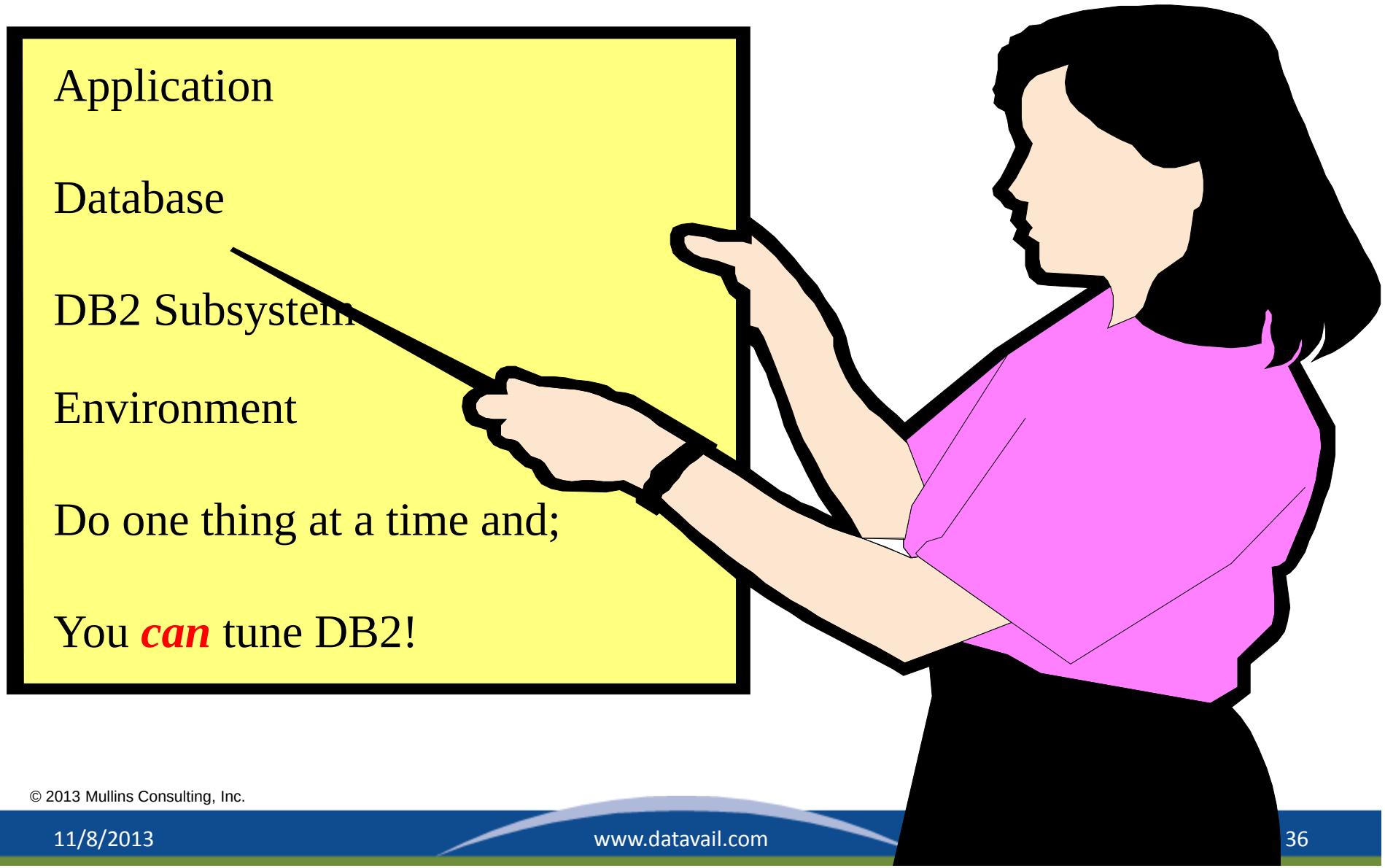
- MAXCSA
  - 250×(LOCKS PER USER)×(MAX USERS)
- ITRACE=NO
  - Do not use ITRACE;
  - Instead, if needed, use DB2 lock traces.
- DEADLOK
  1. The number of seconds between two successive scans for a local deadlock
  2. The number of local scans that occur before a scan for global deadlock starts

250 bytes of storage for each lock.

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.
- Hardware
  - storage, RAID, zIIP/zAAP, etc.

Summary



Have You Considered Remote DBA Services?

- Datavail offers remote DBA services and operates 24 x 7 x 365
- With delivery centers in the U.S. and India, Datavail can cover the time zones in which our clients operate
- Support DBAs are working in live shifts round-the-clock
  - DBAs ready to support your business all the time, as needed, already up & working
  - No need to wake up local DBA for support
  - DBAs working with other DBAs – over 350 DBAs
- Datavail's remote services automatically deliver 24x7 coverage
- By combining the 24x7 remote coverage with a US-based primary DBA a client obtains the benefits of an on-staff DBA plus those of Datavail's remote DBA service

Datavail Performance Management

Monitoring Tools

- For customers who have their own preferred monitoring tools and/or scripts, if desired, Datavail leverages those tools– Datavail will configure it, manage it, and seamlessly integrate into our ticketing system
- For customers who do not have a preferred monitoring solution, Datavail leverages our proprietary monitoring tool called Datavail Delta
- We also leverage our collection of custom scripts to perform initial and periodic environmental health checks

Ticketing System

- Our ticketing system is available to any client using our managed services
- For customers who have their own ticketing system we can leverage your toolset and/or create an API integration into Datavail's ticketing system

All cost related to use of the tools and ITSM platform is included with the service

The Value of Datavail Remote DBA Services

datAvail

The world's largest pure-play database support company

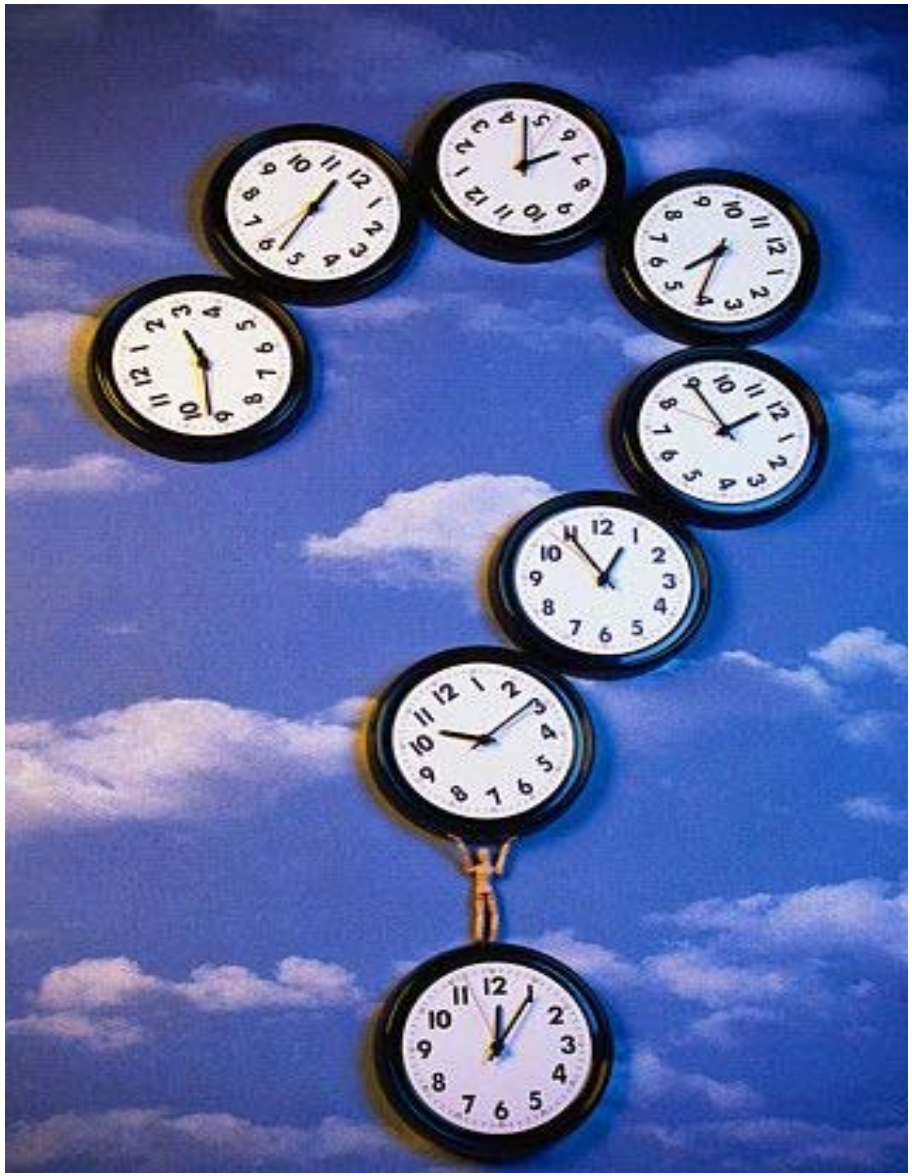
"Datavail delivers a managed services solution -- what you are getting is a small slice of a very sophisticated operation, built over 10 years by people with literally 100's of years of experience in IT operations…"

Datavail Company Profile

Datavail helps you manage your databases, reduce your risk, and lower your cost

- **250+ Customers**
  - Average customer retention of 7 years
- **350+ DBAs**
  - 24x7 Live & Available Tier 1, 2, and 3 DBAs
- **10+ Years Delivering Database Services**
  - Our differentiation is centered in our focus with scale
- **Headquartered in Broomfield, CO**
  - Regional offices Seattle & New York
  - Offshore delivery centers in Mumbai & Bangalore
  - Acquired Blue Gecko & Bluewolf Database Services

- **Strong Year-over-Year growth**
  - Datavail has been growing at a CAGR of 60% over the last 3 years
- **Mature, Proven Delivery Model**
  - Tools, Best Practices and ITIL Processes
  - Shared Services in an on-Shore/Off-Shore Model



Questions

Craig S. Mullins
Mullins Consulting, Inc.
Craig.Mullins@datavail.com
craig@mullinsconsultinginc.com
http://www.mullinsconsultinginc.com

Datavail Corporation
11800 Ridge Parkway – Suite 125
Broomfield, CO 80021
info@datavail.com
http://www.datavail.com
(877) 722-8247