Oh, The Things I've Seen

Db2 Stories and Best Practices

By Craig S. Mullins
(*with a nod to Dr. Seuss*)

Agenda

- One Stats, Two Stats,
  Old Stats, New Stats
- RID-ing is Fundamental
- The DBA Will Review It
- A Small Matter of Lock Size
- Not Db2 at All
- Some General Best Practices

One Stats, Two Stats, Old Stats, New Stats

Dealing With Outdated RUNSTATS

Things I've Seen...

RUNSTATS aren't run,

Now RUNSTATS are old,

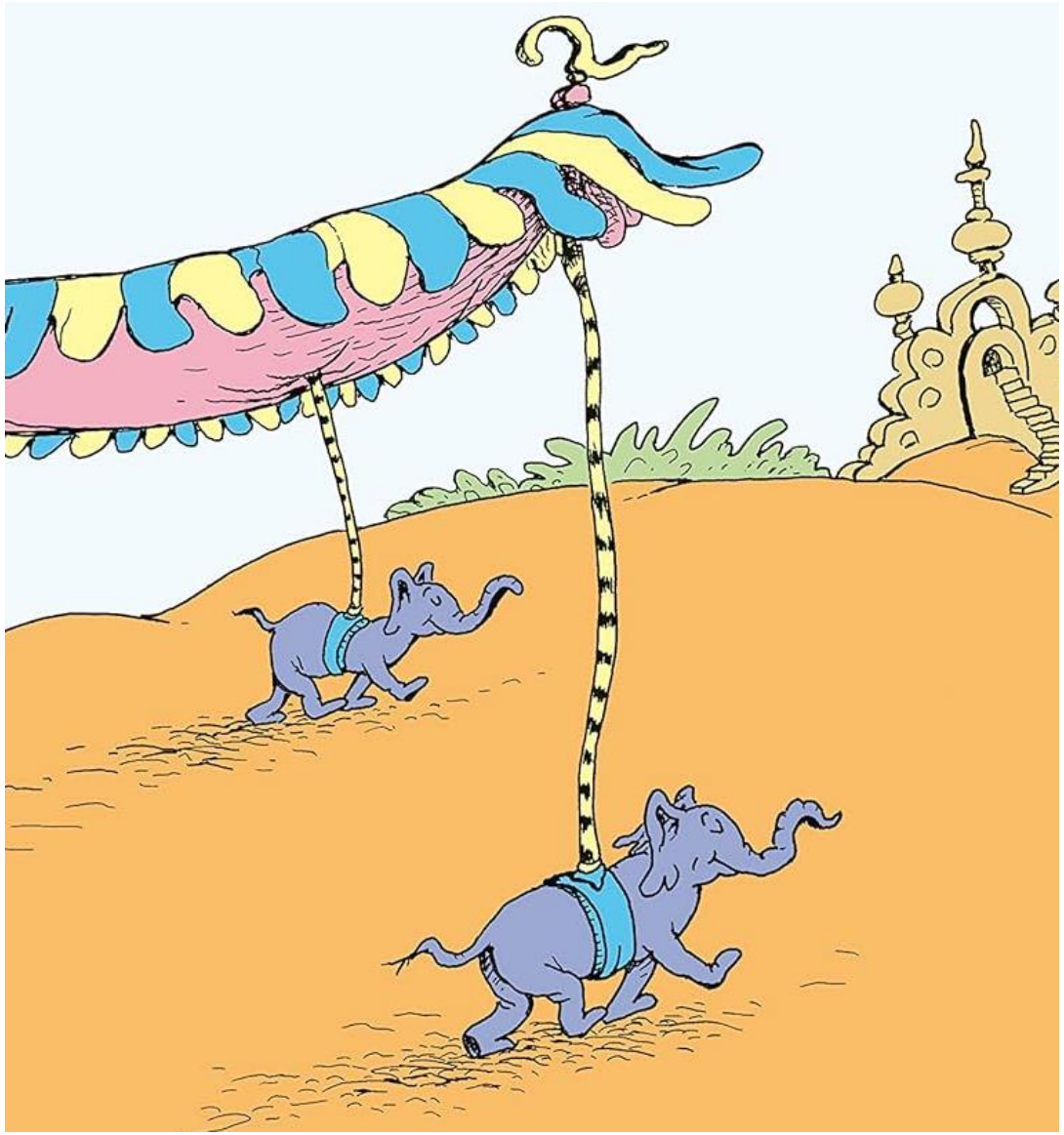
But RUNSTATS aren't needed,

Or so we've been told!

Scenarios

-1

- Some table spaces have never had RUNSTATS run on them.
- Some table spaces had RUNSTATS run once, but never again.
  - Many reach a point of stats stasis but…
- Some table spaces had manual statistics applied.
  - And then never had RUNSTATS run again.
  - FUD sets in, nobody wants to run RUNSTATS!



- Scans data
- Gathers details on data
- Records statistics in the Db2 Catalog
- Used to generate access paths
- Can be used by DBAs, too
  - RTS usually more accurate

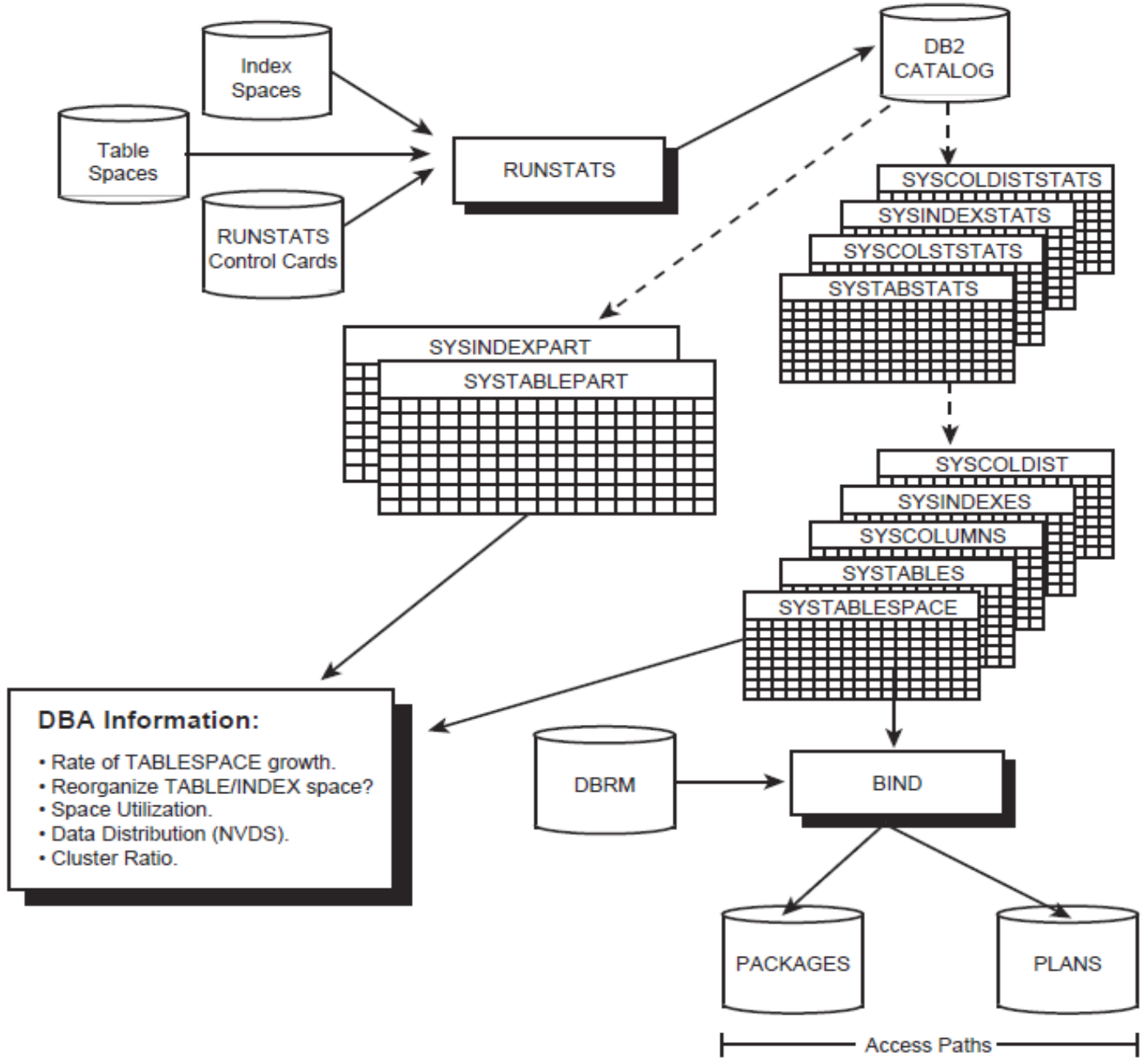


Table Statistics

- CARDF
  - **Number of rows in the table (or partition)**
- NPAGESF
  - **Number of pages containing data**
- NACTIVEF
  - **Number of active pages in the table space**
- PCTROWCOMP
  - **Percentage of rows that are compressed**

Index Statistics

- **NLEAF – number of active leaf pages**
- **NLEVELS – number of levels in the index**
- **CLUSTERRATIOF – the percentage of rows that are in clustered order**
- **CLUSTERING – whether the index is the clustering index**
- **FIRSTKEYCARDF – number of distinct values for the first column of the index**
- **FULLKEYCARDF – number of distinct values for the complete index key (all columns in the key)**

Column Statistics

Single Column

- COLCARDF – Number of distinct values for a column
  - Assumes data is uniformly distributed
- HIGH2KEY – Second highest key value
- LOW2KEY – Second lowest key value

Multiple Correlated Columns

- SYSCOLDIST.CARDF – Number of distinct values for a group of column
  - Useful for correlated data
  - (TYPE=C, NUMCOLS>1)

Distribution Statistics

- Non-Uniform Distribution Statistics (NUDS), also called FREQVAL statistics, are useful for determining skew.
  - Without them, data is assumed to be uniform.
  - SYSCOLDIST.FREQUENCYF – number of times a given value (or values) occur(s)
    - Can be collected for a single column
      - (TYPE = F, NUMCOLS = 1)
    - Or for multiple columns with COLGROUP
      - (TYPE = F, NUMCOLS > 1)

Histogram Statistics

- Statistics in quantiles over intervals
  - aka Range statistics
  - Maximum of 100 quantiles
    - Fewer than 10 quantiles, reverts to distribution statistics
  - A column value belongs to only one quantile
    - NULL has its own separate quantile
  - Db2 will attempt to:
    - keep the quantiles of similar size
      - same # of rows, not same # of values
    - avoid big gaps between the quantiles

What Should You Collect?

- Always collect basic statistics.
  - Table and Index
- Collect column stats for important predicates and ORDER BY clauses.
- Collect NUDS when data is skewed.
  - e.g.) cigar smokers skew male
- Collect correlation stats when two or more columns are highly correlated.
  - e.g.) CITY, STATE, ZIP
- Collect histogram statistics when data skews by range.
  - e.g.) Lunch rush or Christmas shopping season

RUNSTATS Guidelines (1)

- Collect RUNSTATS for all indexes

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81D
    INDEX (ALL)
```

- Collect RUNSTATS for columns in sensitive WHERE and ORDER BY clauses

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E
  TABLE (DSN8810.EMP)
  COLUMN (FIRSTNME, LASTNAME, SALARY)
```

RUNSTATS Guidelines (2)

- Collect RUNSTATS for skewed data

```
RUNSTATS TABLESPACE
DSN8D81A.DSN8S81E
  TABLE (DSN8810.EMP)
  COLGROUP(JOB) FREQVAL COUNT 10
```

RUNSTATS Guidelines (3)

- Collect RUNSTATS for correlated data

```
RUNSTATS TABLESPACE
  DSN8D81A.DSN8S81E
    TABLE (DSN8810.EMP)
    COLGROUP(CITY, STATE, ZIP)
    FREQVAL COUNT 10
```

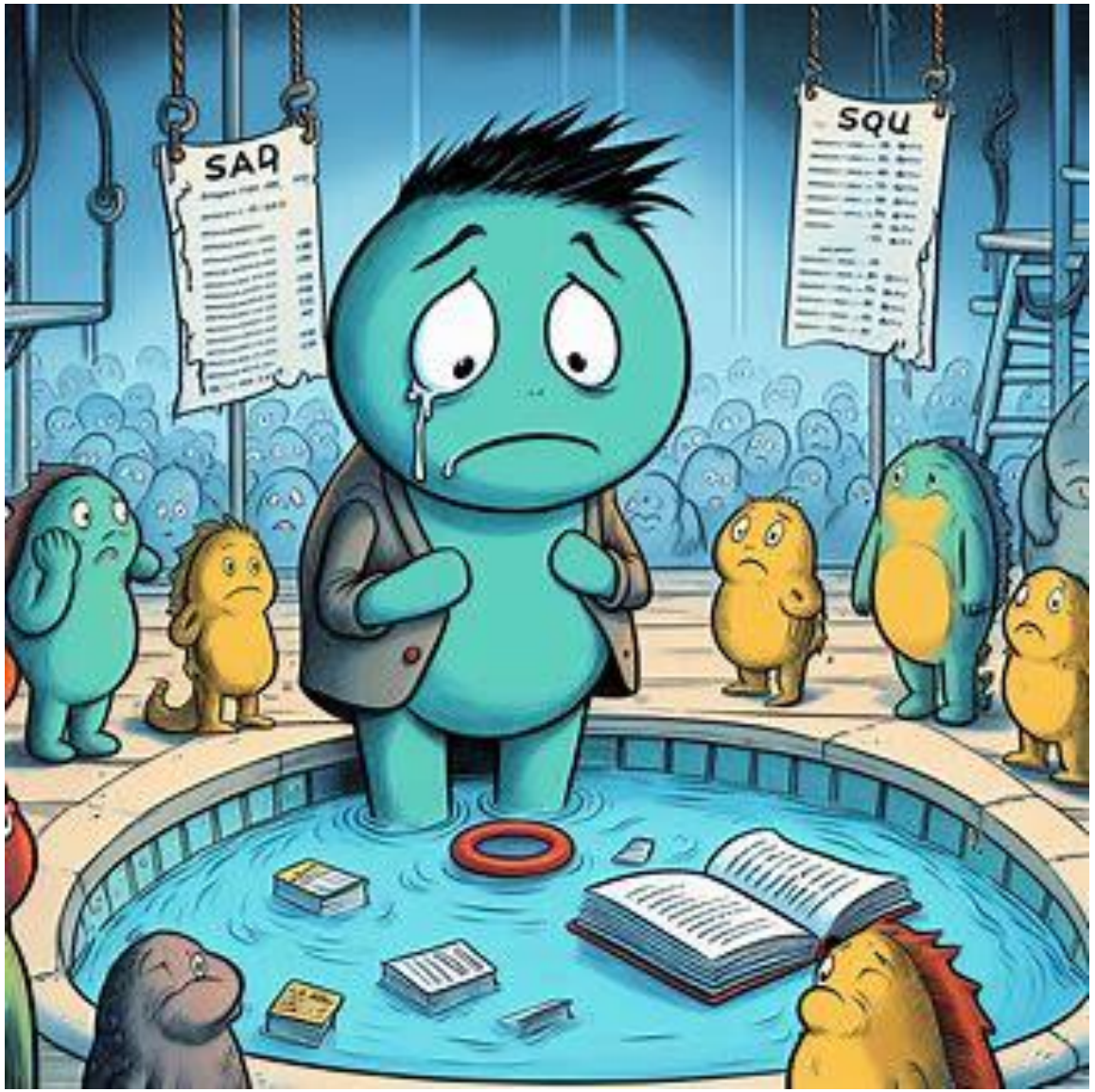
RUNSTATS Guidelines (4)

- Collect Histogram RUNSTATS for range skews
  - Stores QUANTILENO, LOWVALUE, and HIGHVALUE for up to 100 quantiles

```
RUNSTATS INDEX (OWNER.XTRG)
   HISTOGRAM NUMCOLS 2
```

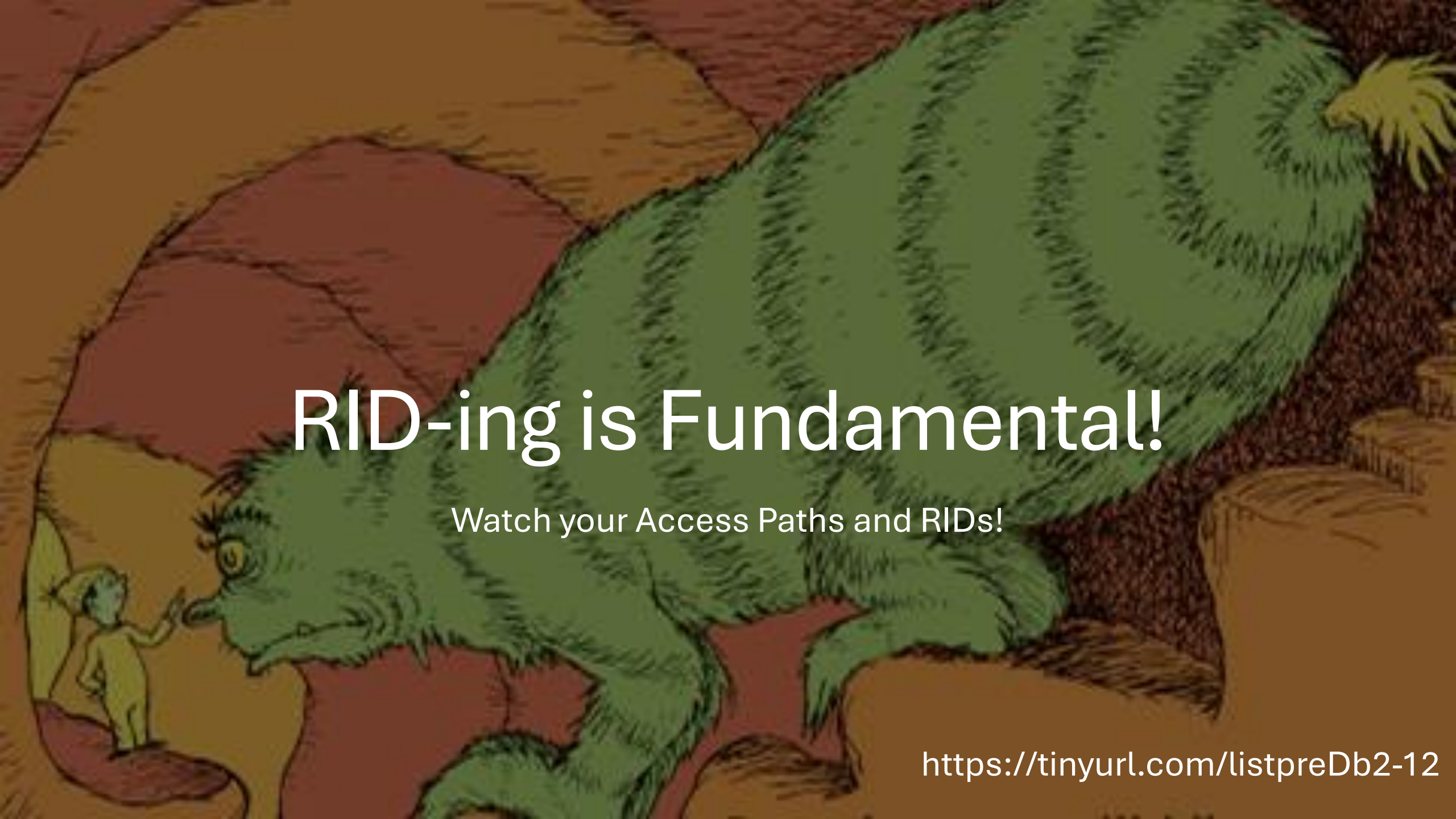
e.g.) DATE, TIMESTAMP

Breakfast	**6:00-6:45**	**Slow**
	6:46-8:10	**Busy**
	8:11-11:25	**Slow**
Lunch	**11:25-1:15**	**Busy**
	1:16-1:45	**Moderate**
	1:46-4:55	**Slow**
Dinner	**4:56-5:45**	**Moderate**
	5:46-7:15	**Busy**
	7:16-11:00	**Slow**



Things I've Seen

Outdated statistics,
a silent foe,
Caused the RID pool
to overflow.
Queries stumbled,
indexes failed,
As SQL's prowess
became impaled.

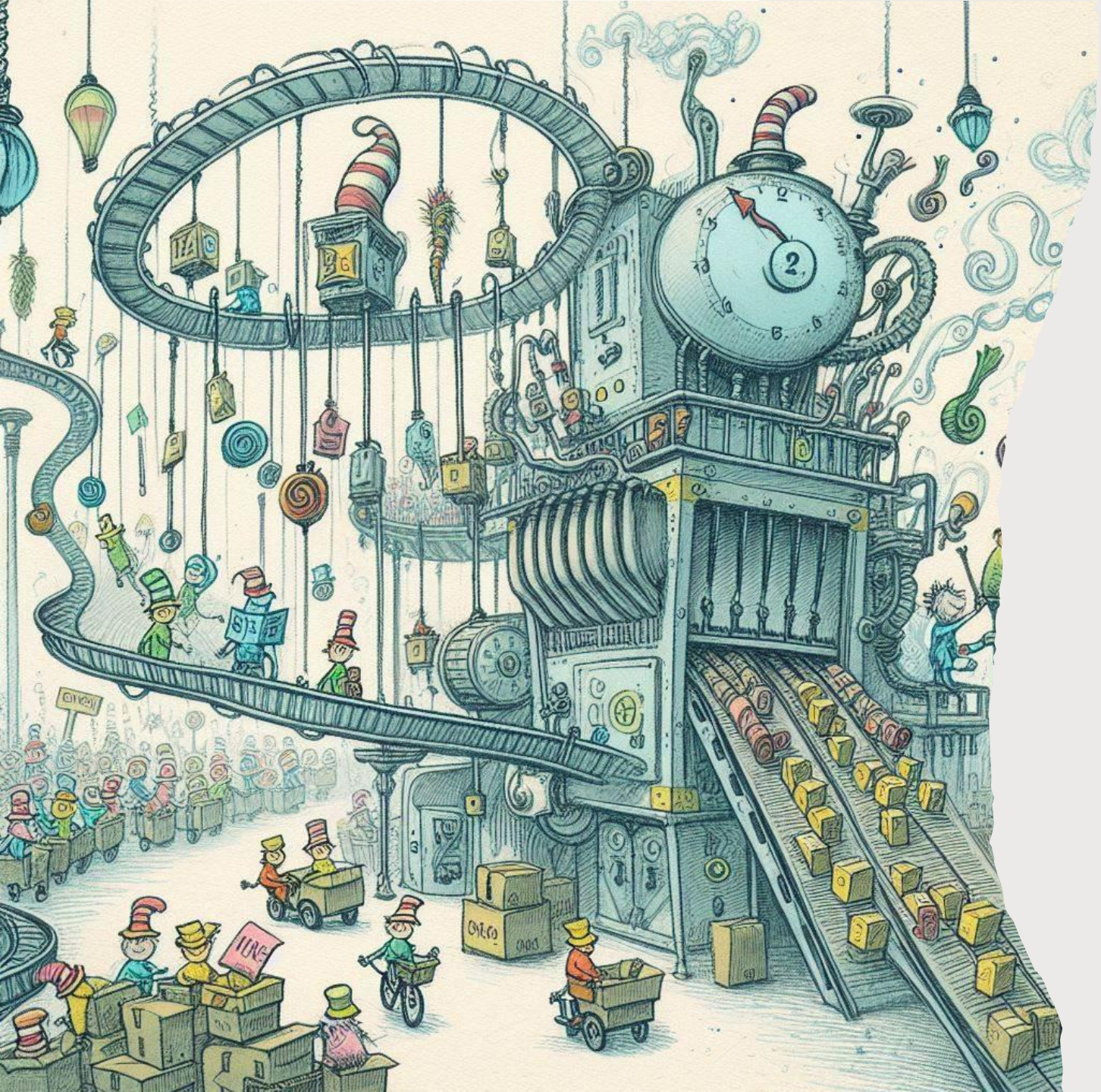

RID-ing is Fundamental!
Watch your Access Paths and RIDs!
https://tinyurl.com/listpreDb2-12

What is a RID?

- RID = **R**ecord **ID**entifier
- Unique identifier assigned to each row in a table
  - Page number and location on the page
- A physical address used by Db2 to…
  - Locate specific rows
  - Access specific rows efficiently
- RID Pool - used to minimize I/O in certain access paths

When does Db2 use RIDs in access paths

- The RID pool is used for:
  - Enforcing unique keys for multi-row updates
  - List prefetch
  - Multiple index access
  - Hybrid joins
- The RID pool is allocated dynamically as it is needed
  - MAXRBLK zparm defines size of the RID pool
    - Setting MAXRBLK to 0 disables RID list processing
- Work file can be used for RID lists when RID pool storage is insufficient
  - MAXTEMPS_RID zparm defines the maximum number of RIDs that can be stored in the work file



What is List Prefetch?

- When list prefetch is involved in an access path:
  - Db2 retrieves a list of RIDs through a matching index scan on one or more indexes.
  - The list of RIDs is sorted in ascending order by page number.
  - Pages are prefetched in order, using the sorted list of RIDs.

- Ideal for non-sequential qualified rows
  - Skip sequential or sparse
  - Large DATAREPEATFACTORF

When is List Prefetch Used?

- Typically, with a single index that has a cluster ratio lower than 80%
- Sometimes on indexes with a high cluster ratio, if the estimated amount of data to be accessed is too small to make sequential prefetch efficient, but large enough to require more than one regular read.
- Always to access data by multiple index access.
- Always to access data from the inner table during a hybrid join.
- Typically for updatable cursors when the index contains columns that might be updated.
- When IN-list predicates are used through an in-memory table as matching predicates (ACCESSTYPE='IN').

https://www.ibm.com/docs/en/db2-for-zos/12?topic=u-list-prefetch-prefetchl

But List Prefetch Can Cause Problems

Types of RID Problems

RDS Limit exceeded *

The number of RIDs that can fit into the guaran-teed number of RID blocks was **greater than 25% of the table**.

DM Limit Exceeded *

The number of RID entries was greater than the **physical limit** of approximately **26 million** RIDs.

Proc.Lim. Exceeded *

RID pool storage was exceeded.

Overflow

RID list overflowed to a work file.

* Index access abandoned and replaced with tablespace scan.

RDS Problems Identified

```
RID Processing Failures...
  RID Pool Shortage.......          0              0
  No Virtual Storage......          0              0
RID Processing Interrupted
  Exceeded RDS Limit......        245         957462
  Exceeded DM Limit.......          0             61
  Append: No Storage......          0              0
```

RDS Limit Failure Details

10JAN2024 09:18:21 ----- BMC AMI Ops WINDOW INTERFACE (V7.1.00) ----------
COMMAND ===> SCROLL ===> CSR
CURR WIN ===> 1 ALT WIN ===>
>W1 =RIDLIST==========(ALL=======*========)10JAN2024==09:15:38=====MVDB2=====D=3693

Time of Record	Date of Record	Db2 ID	RIDS Used	Number Of RIDS	Stg Fail	RID Fail	Plan Name	Package Name	Collect ID
05:37:00.01	10JAN2024	[redacted]	YES	0	NO	W	[redacted]	[redacted]	[redacted]
05:34:12.95	10JAN2024	[redacted]	YES	0	NO	W	[redacted]	[redacted]	[redacted]
05:33:44.17	10JAN2024	[redacted]	YES	0	NO	W	[redacted]	[redacted]	[redacted]
05:32:48.00	10JAN2024	[redacted]	YES	0	NO		[redacted]	[redacted]	[redacted]
05:31:56.63	10JAN2024	[redacted]	YES	0	NO	W	[redacted]	[redacted]	[redacted]
05:30:47.13	10JAN2024	[redacted]	YES	0	NO		[redacted]	[redacted]	[redacted]
05:30:20.62	10JAN2024	[redacted]	NO	0	NO	M	DISTSERV	SYSLH200	NULLID
05:29:54.78	10JAN2024	[redacted]	YES	0	NO		[redacted]	[redacted]	[redacted]
05:29:44.44	10JAN2024	[redacted]	YES	0	NO		[redacted]	[redacted]	[redacted]
05:27:10.96	10JAN2024	[redacted]	NO	0	NO	M	DISTSERV	BIDBCT2	QMFW
05:26:41.16	10JAN2024	[redacted]	YES	0	[redacted]				
05:26:21.57	10JAN2024	[redacted]	YES		NO		[redacted]	[redacted]	[redacted]
05:25:37.91	10JAN2024	[redacted]	YES		NO	W	[redacted]	[redacted]	[redacted]
05:25:27.92	10JAN2024	[redacted]	YES		NO	W	[redacted]	[redacted]	[redacted]
05:25:19.57	10JAN2024	[redacted]	YES		NO	W	[redacted]	[redacted]	[redacted]
05:25:10.75	10JAN2024	[redacted]	YES		NO	W	[redacted]	[redacted]	[redacted]
05:23:28.19	10JAN2024	[redacted]	YES		NO	W	[redacted]	[redacted]	[redacted]
05:23:11.83	10JAN2024	[redacted]	YES	0	NO		[redacted]	[redacted]	[redacted]

MA +
»
02/016

me of	Statment	Pre-Cor
cord	Number	Token
:15:19.55	1705	002029E
:15:19.49	1705	002029E
:15:18.84	1609	002029E
:15:18.76	1609	002029E
:15:18.61	1609	002029E
:15:18.55	1609	002029E
:15:18.28	1609	002029E
:15:18.21	1609	002029E
:15:17.39	1513	002029E
:15:17.28	1513	002029E
:15:17.06	1513	002029E
:15:16.98	1513	002029E
:15:16.51	1513	002029E
:15:16.14	1513	002029E
:15:16.56	2599	001E809
	716	00181D6
:00:41.24	1136	001F790
:59:49.44	9094	001AED5

Statement Numbers

Sometimes the PLAN_TABLE Data was Not There!

- Best practice was EXPLAIN(YES)
  - Either they didn't follow it or…
  - They deleted PLAN_TABLE data over time as the table grew
- Solution?
  - EXPLAIN PACKAGE
    - Nobody there had ever done this
  - Execution requires (one of):
    - SQLADM, SYSADM, SYSOPR, or SYSCTRL
  - And, of course, I did not have any of those authorities
    - Waited……

Most Common Reason for RID Failures

- RUNSTATS!
  - Never Run
  - Inaccurate
  - Outdated

Dealing with RID Pool Failures

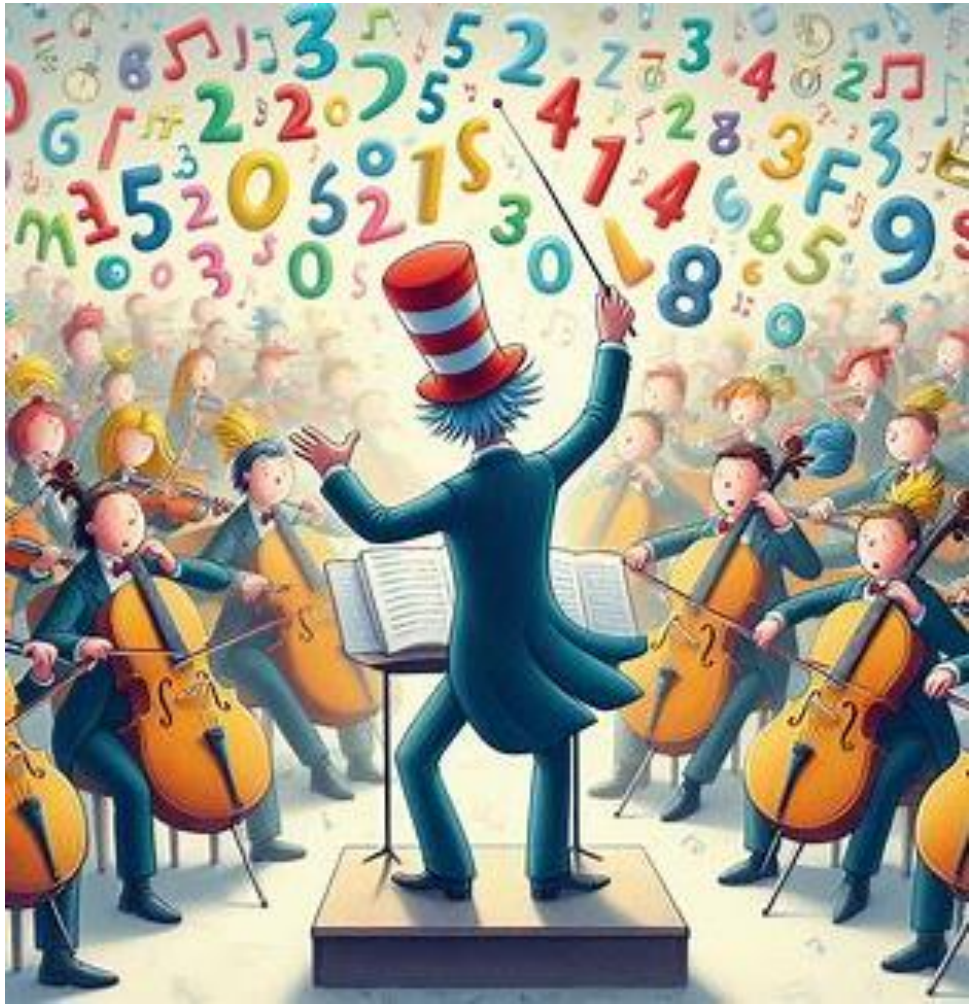
- Run RUNSTATS and REBIND
- Add (or change) an index
- OPTIMIZE FOR 1 ROW
- REOPT(ALWAYS)
- IDAA?

What I Did

- Monitored daily for RDS failures
- Captured package and StmtNo
- Reviewed access paths
  - Not always there
- Recommended fixes
  - New index
  - OPTIMIZE FOR 1 ROW
    - Not always possible
  - REBIND
  - IDAA QUERYACCELERATION(ELIGIBLE)

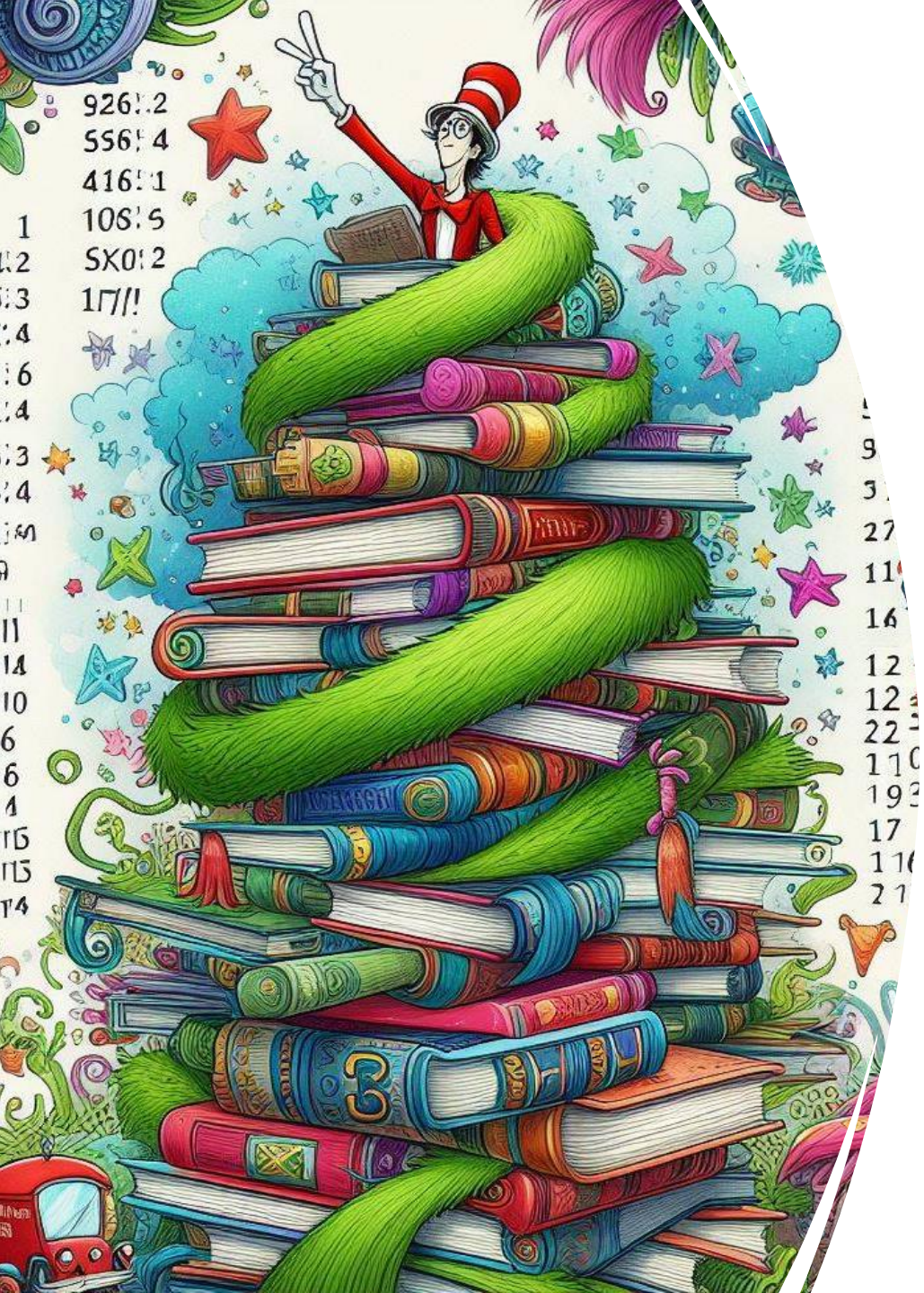
Things I've Seen

So let us heed
this cautionary tale,
And keep our statistics
fresh without fail.
For in the world of data,
chaos can reign,
And success relies
on keeping stats sane.



The DBA Will Review It

No Need to Worry!



Things I've Seen

With scrutinizing eyes
and a meticulous hand,
The DBA reviews all…
a task ever grand.
All queries and updates,
each column and row,
No badness escapes,
this we all know!

The Situation

- I was working on a team of DBAs for a period of time
- Developers send "data fix SQL" to DBAs to review and accept for accuracy
- Done instead of coding programs to correct data problems
  - Instead, just issue some SQL
  - Mostly INSERT and UPDATE statements
- One group did this frequently
- Usually, a dozen or so statements
- And, oh yes, this was production!

Then...

- …I got a request with hundreds and hundreds of INSERT and UPDATE statements!

The Truth

- A DBA will find 10 problems in a 12-line SQL statement…

- But nothing in a 7,500-line program!

What I Did

- I refused to review it.
  - It is an application issue.
- Instead, I took an image COPY of the tables that were being modified before leaving for the day.
  - I knew I'd have something to fall back to if the modifications were somehow wrong.

Also

- I recommended to the DBA team lead that reviewing SQL this way is not a good plan moving forward
- Why?
  - If the DBA reviews it and misses something **it becomes the DBA's fault**!
  - It is better to write "fix" programs to implement changes because it can be **easier to track and back out** problems
  - If data is constantly needing to be modified like this there is some sort of **root cause** that should be analyzed and remediated instead of just constantly fixing data

In the realm of data,
where things are not right,
Lies a daunting task,
a DBA's great fight.
Hundreds of statements,
awaiting review,
But why the DBA? What did he do?

Line by line, the DBA delves,
Through the pages of statements,
where the data is shelved.
But as the pile grows taller,
the hours they wane,
And efficiency falters,
under the strain.

Instead let us honor the
programmer's art,
In crafting a program
to play its true part.
To change so much data
correctly it seems,
A program solves
all of the DBA's dreams!

A Small Matter of Lock Size

Escalating locking problems!

Lock Escalation!

In Db2 land,
so big and so wide,
Lived apps that locked,
oh, how they'd collide!
When queries ran,
all eager and quick,
Locks would escalate,
creating quite the shtick.

Lots of Lock Escalations Occurring

- *Lock escalation* occurs when a threshold is hit
  - DSNZPARMs
    - NUMLKUS
    - NUMLKTS
    - Db2 12 FL 507 delivers built-in global variable so these can be set at the package level
  - LOCKMAX

- What is a lock escalation?
  - Row or page locks, held by an application process on a single table or tablespace, are released.
  - And a tablespace lock, or a set of partition locks is acquired.
  - When locks escalation occurs, Db2 issues message DSNI031I, which identifies the tablespace for which lock escalation occurred, and some information to help identify the plan or package that was running when the escalation occurred.

Finding Them

- Viewed MSTR log
- Looking for DSNI031I
  - Table space (Resource)
  - Package
  - Statement number

```
05.58.24 S0007433  DSNI031I  - DSNILKES - LOCK ESCALATION HAS OCCURRED  984
   984             FOR
   984               RESOURCE NAME = D
   984               LOCK STATE =  X
   984               PLAN NAME : PACKAGE NAME = D      CH :
   984               COLLECTION-ID =
   984               STATEMENT NUMBER = 00002668
   984               CORRELATION-ID =
   984               CONNECTION-ID = BATCH
   984               LUW-ID = USDISAZN.A      1.DEF8216C2711
   984               THREAD-INFO =
   984             DSSBPGU:BATCH:DSSBPGU:GUADAD98:STATIC:55785992:*:<
   984               PARTITION-INFO = PART 1 AND 0 OTHER PARTS ESCALATED
```

Recurring Theme

- Escalating on tablespaces with LOCKSIZE ROW
- Never re-evaluated LOCKMAX
  - All were set to SYSTEM
- Moving from PAGE to ROW locking means acquiring more locks because there are multiple rows per page
  - But when they changed from ANY/PAGE to ROW no add'l analysis was done!

Check COMMIT Frequency

- Some programs had no COMMIT logic
  - This should NEVER be the case for any program that changes data!
- Adjustments to programs with COMMIT logic
  - COMMIT more frequently to reduce the instances of lock escalation
- Single DELETE
  - Piece-wise DELETE

Consider Setting LOCKMAX

- Number of rows per page multiplied by NUMLKTS
  - Also take into account compression

- Still may have issues with NUMLKUS
  - Consider the host variables added with Db2 FL507
  - SYSIBMADM.MAX_LOCKS_PER_TABLESPACE
  - SYSIBMADM.MAX_LOCKS_PER_USER

Lock De-escalation!

First, it starts small, just a row or two,
But as things heated up, it grew into view.
Shared locks turn exclusive, escalating high,
Blocking other queries as they pass by.
So, programmers and admins dug right in,
To smooth things out keeping problems to the min.
So, remember, dear friend, in Db2's domain,
Lock escalation's a dance, a delicate game.
With finesse and precision, it keeps things in line,
In the wondrous world of Db2, so fine!

Not Db2 at all

Big problems, it must be Db2!

Things I've Seen

In the realm of the mainframe,
a mystery did brew,

A problem perplexed,
with no clue what to do.

Db2, they suspected,
the culprit for sure,

But beneath the surface,
dwelled a different cure!

The Situation

- Hired to help with a "*significant performance problem*"
- System was an entangled mass of parts
  - Old application that ran on batch, CICS, VSAM and flat files
  - Many program from old app converted to Db2 and CICS, but not all
  - Other portions rewritten in .NET using Oracle

A Few More Details

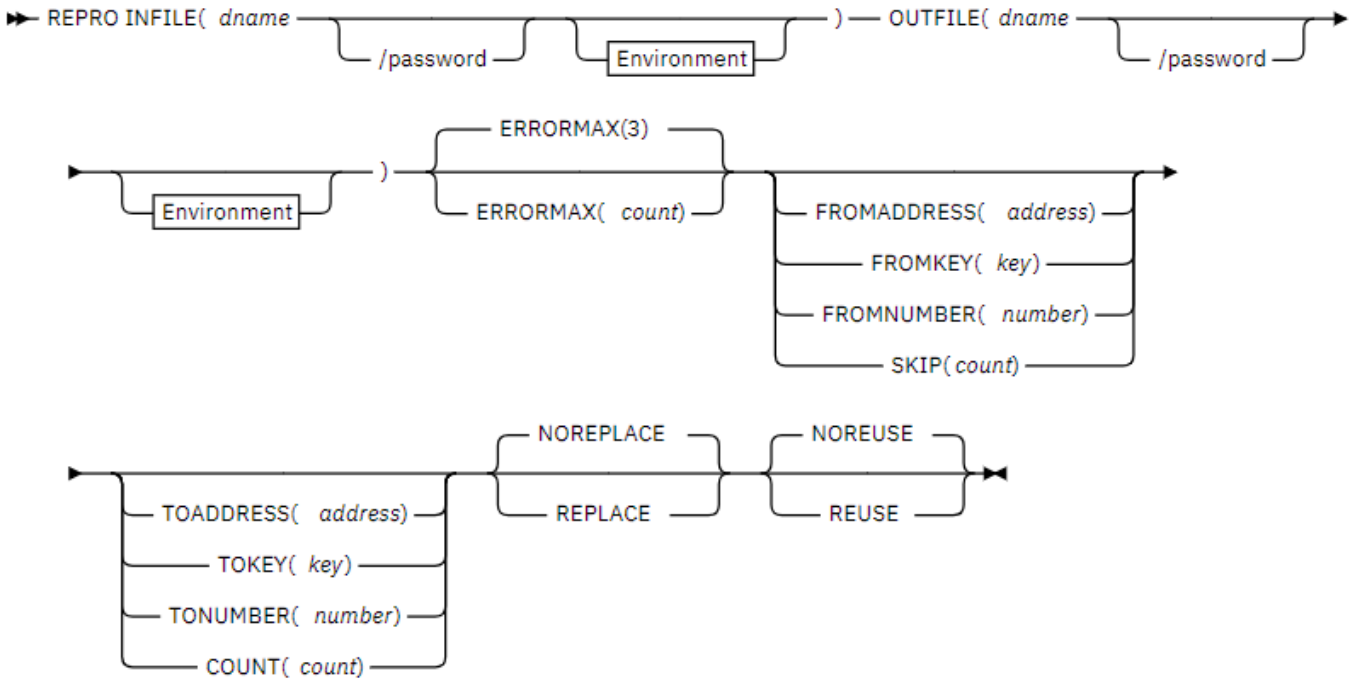
- They had hired other experts for the other parts of the application
- Wanted me to focus on their Db2 SQL
- I reviewed access paths and tuned many statements
  - Had to be implemented by programmers
- Regular tests of the entire application conducted
  - Small performance gains

Then One of the Old Guys They Brought Out of Retirement Had an Idea

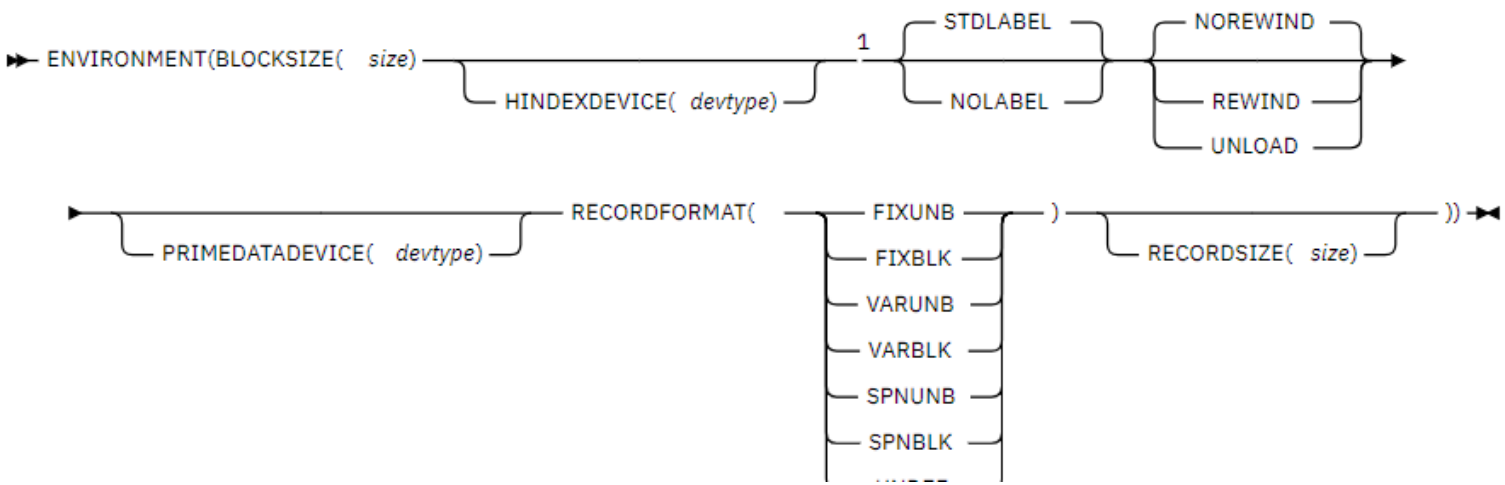
Have you looked at the VSAM files lately?

And…

- They had not been attended to in years…
  - Actually decades!
- Very disorganized.
- Somebody suggested running AMS REPRO



Environment



What Happens When REPRO is Not Run on VSAM Data Sets

- Disorganization!
  - Data integrity issues
  - Performance degradation
  - Index inconsistencies
  - Increased space usage
  - Risk of duplicate records

The Lesson

Oh, the lesson learned,
in this database tale,
To look beyond the obvious,
without fail.
For sometimes the problem,
though it may seem,
Is lurking elsewhere,
in a different scheme.

Best Practices

Some Good High-Level “Things” to Do

High Level Best Practices

- Keep maintenance up-to-date
- Ensure that appropriate RUNSTATS are being run
  - And kept up-to-date
- Automate as much as you can
  - May require tools
- Make sure backups are taken for all database objects (RTOs)
  - And test your backups
- Make sure every program has a COMMIT strategy in place!
- Document!
  - And make the doc accessible!
- Be proactive.

Always Be Learning

The MORE that you READ,
The MORE things you will KNOW.
The MORE that you LEARN,
The more PLACES you'll GO.
–Dr. Seuss

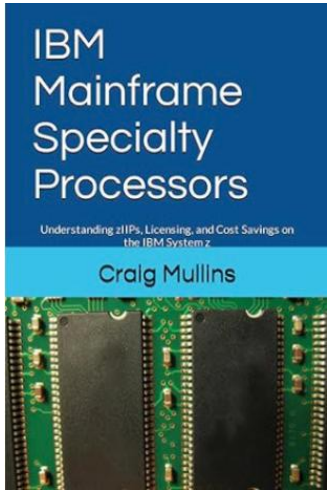
Oh, The Things I've Seen!

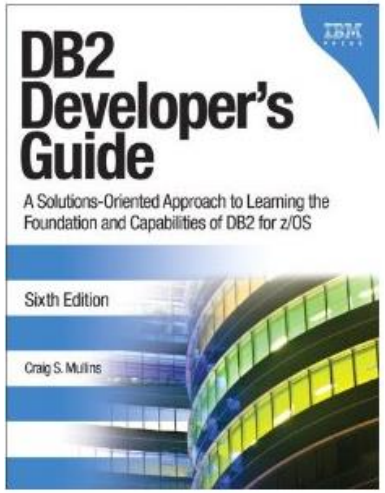
Craig S. Mullins
President & Principal Consultant
IBM Champion for Data & AI
IBM Gold Consultant
mullc@craigsmullins.com

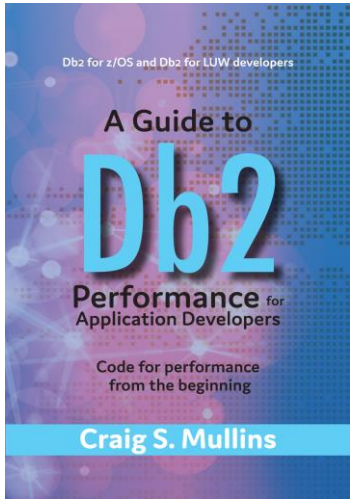
www.mullinsconsulting.com

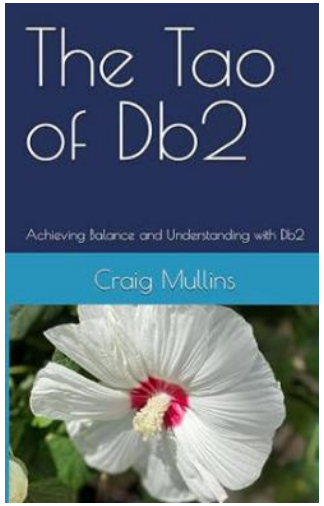
IBM CHAMPION

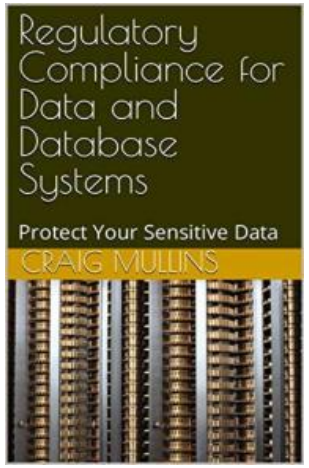
IBM GOLD Consultant

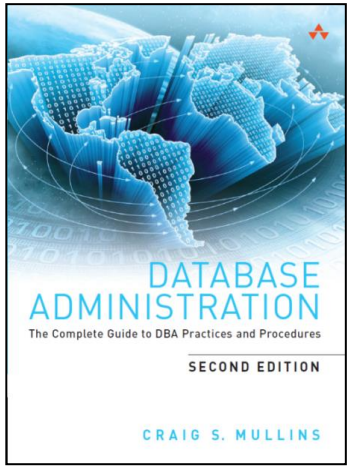


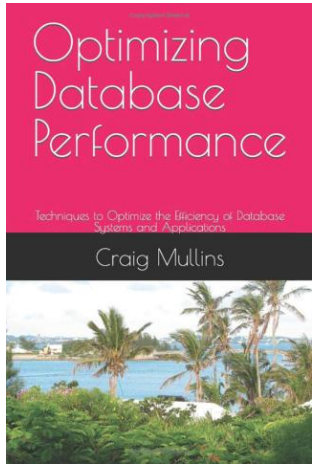


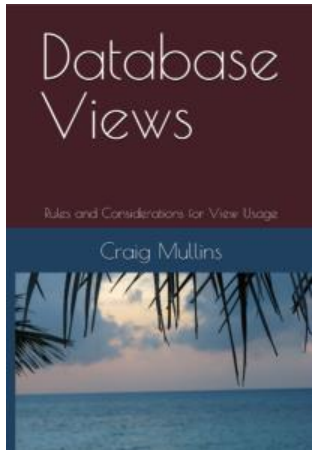












https://www.mullinsconsulting.com/books.htm