The Top Ten Db2 Things You Need to Know

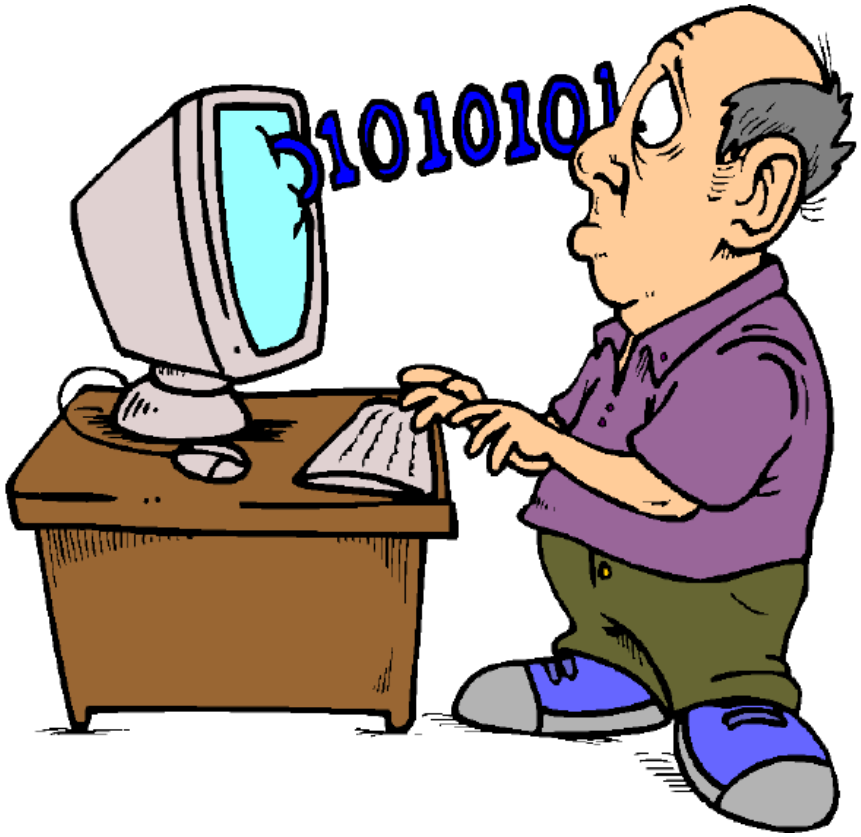
For DBAs and Developers

Craig S. Mullins
Mullins Consulting, Inc.
http://www.MullinsConsulting.com

The Top Ten Things for DB2 Developers

The following countdown will focus on tips, techniques, and tactics for better application development when using DB2 for z/OS.

Most of the "things" will be related to coding practices, but some will focus on career management and communi-cation.



Top Ten Things You Need to Know

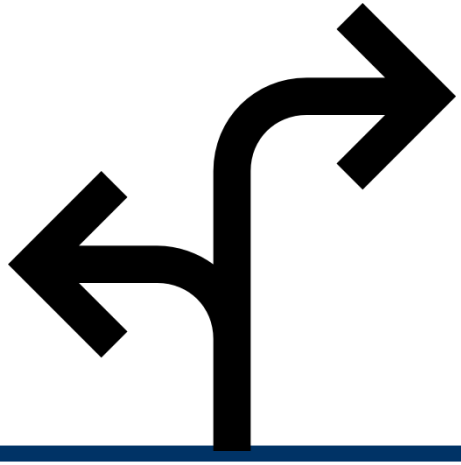
Today's presentation will be delivered in two sections.

One each for:

- Developers
- DBA

Offers up my opinion as to the Top Ten things that are mandatory for ensuring the effective use of DB2 in terms of:

- Ease of use
- Performance
- Availability
- Career growth



1. Master the Basics

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

Favor Stage 1 and Indexable predicates

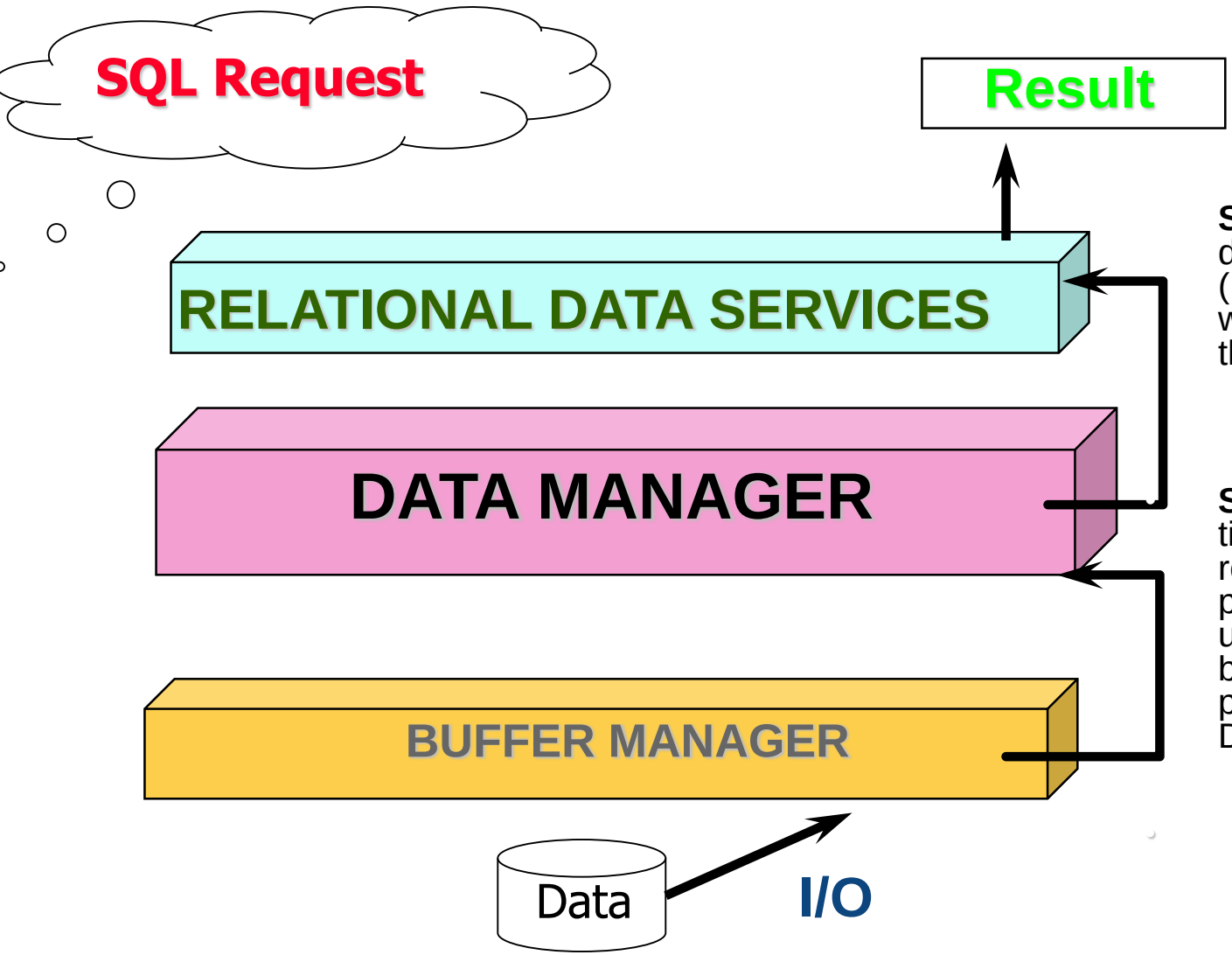
- Host variable data type/length should match column
- Print and post the current list – SC19-2978 Managing Performance – Table 66 – Page 269

Avoid table space scans for large tables

Avoid sorting when possible:

- indexes for ORDER BY and GROUP BY
- judicious use of DISTINCT
- UNION ALL versus UNION

The Difference Between Stages 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

1. More Basics - *It Depends!*

Understand your circumstances & apply what makes sense.

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.

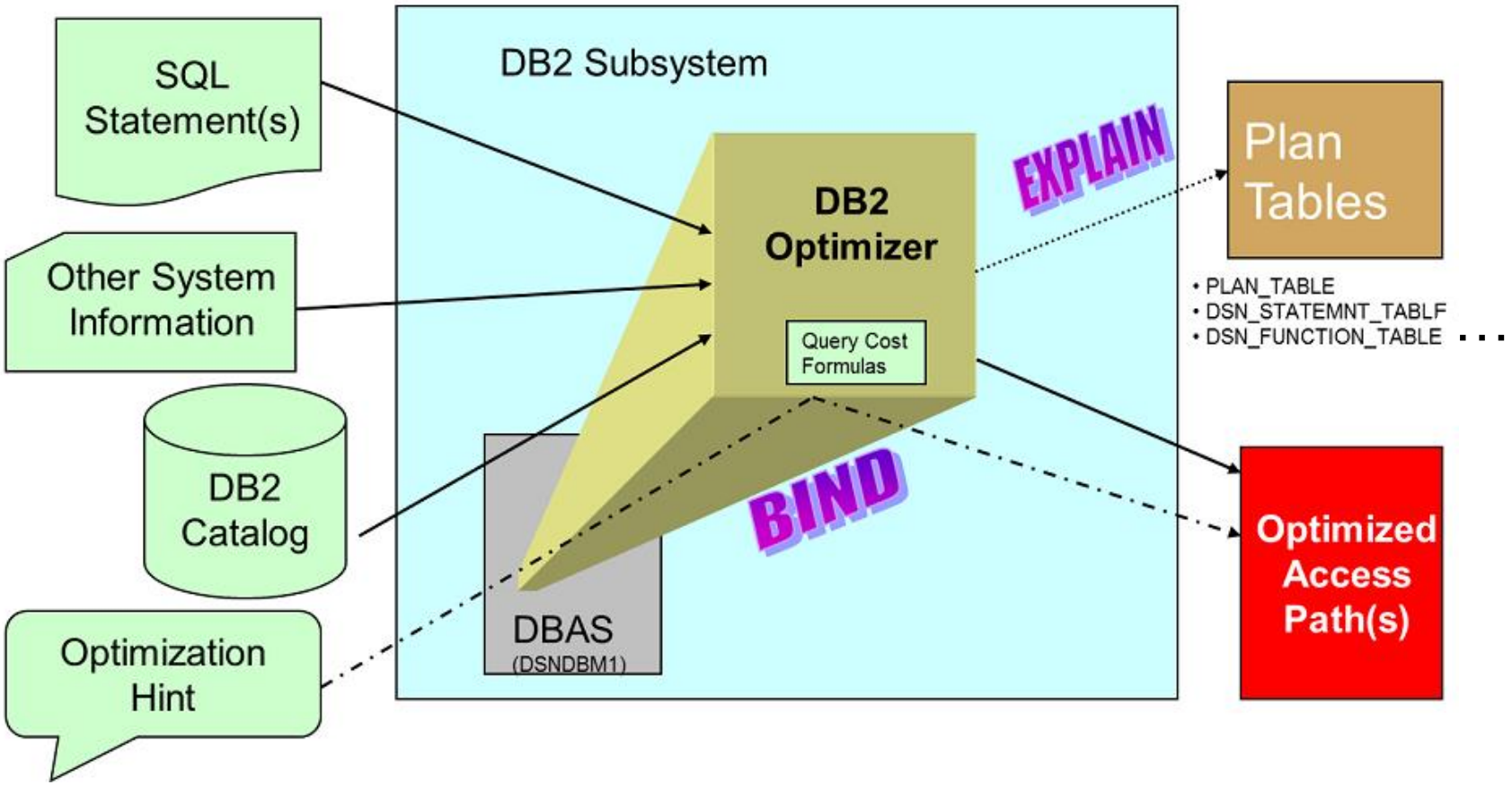
Don't be discouraged when you ask the local expert which statement will perform better, and the answer is "It depends." The expert is just doing his or her job. The secret to optimizing DB2 performance is being able to answer the follow-up question to "It depends"—and that is "**What does it depend on?**"

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.

There is a corollary to the "It depends" rule that also is important. Hard and fast rules that rigidly enforce or forbid usage are not a good idea. In other words, this corollary is simply stated: "Almost never say always or never." Notice that even the wording of this corollary embraces flexibility.

2. Understand The Basics of the DB2 Optimizer

- The DB2 Optimizer makes all decisions on how data will be accessed, not you and not your application code.



2. Understand The Basics of the DB2 Optimizer (cont.)

Expert Db2 tuning professional try to understand all of the ways that the Db2 Optimizer works.

- Some DBAs aspire to this

Application developers will likely never master all of the intricacies of the Db2 Optimizer.

- And that is okay!
  - No one (outside of IBM) fully understands every nuance of the optimizer
  - We can learn general rules, one of which is to entrust the optimizer to create more efficient access than we can write ourselves

› Guideline: Do as much work as possible within the SQL

› Guideline: DB2 code is more efficient than my/your code

3. Avoid "Flat File"/Master File Processing With DB2

Unlearn the "flat file" mentality… learn to Join!

- "Master file" processing is not appropriate for optimal DB2 applications
  - Open Cursor is NOT the same as Open File

The database is not a set of files

- Files have no relationships set within and among them

Tables are not files, they are based on sets:

- Sets are not ordered; members of a set are all of the same type.
- When you perform an operation on a set, the action happens "all at once" to all the members of the set.

Rows are not records: Records are sequential, rows have no physical order

Columns are not fields: Columns are typed (and can be NULL); not so for fields.

- Without a program a field has no meaning.

4. Build Better Programs by Avoiding Stages 3 & 4

We learned about Stages 1 and 2, with Stage 1 being more efficient than Stage 2.

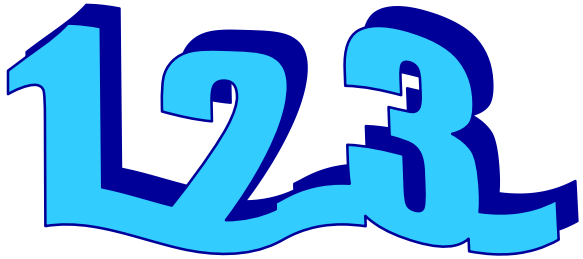
- Now let's learn about Stages 3 and 4.

Stage 3 can be thought of as moving predicates from SQL into your programs.

- Instead of coding WHERE clauses in SQL, pulling the data into the program and checking it there (IF…THEN)

If the data has to be brought into the program before it is filtered out, performance will suffer!

- Stage 1 better than Stage 2
- Stage 2 better than Stage 3



4. Build Better Programs by Avoiding Stages 3 & 4 *(cont.)*

Avoid Black Boxes

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

Stage 4?

5. Learn How to Code for NULL

NULL represents the absence of a value. It is not the same as zero or an empty string.

A null is not a "null value" – there is no value.

- Maybe a "Null Lack of Value"

Consider the following columns:

- TERMINATION_DATE – null or a valid date?
- SALARY – null or zero?
- SSN – non-US resident?
- HAIR_COLOR – what about bald men?

5. Coding for NULL

DB2 uses a null-indicator to set a column to null

- One byte
  - Zero or positive value means not null
  - Negative value means null

Not stored in column, but associated with column

Programs have to be coded to explicitly deal with the possibility of null

```
EXEC SQL
  SELECT   EMPNO, SALARY
    INTO     :EMPNO,
             :SALARY:SALARY-IND
    FROM     EMP
    WHERE    EMPNO = '000100'
END-EXEC.
```

5. Coding for NULL *(cont.)*

If you do not specify a null indicator or structure for a nullable column you will get an error if a null is returned:

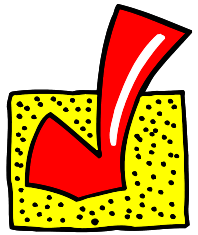
- SQLCODE
- SQLSTATE

THE NULL VALUE CANNOT BE ASSIGNED TO OUTPUT HOST VARIABLE NUMBER *position-number* BECAUSE NO INDICATOR VARIABLE IS SPECIFIED

5. Coding for NULL *(cont.)*

This is Correct

```
SELECT   EMPNO, WORKDEPT, SALARY
FROM     EMP
WHERE    SALARY IS NULL;
```



This is Incorrect

```
SELECT   EMPNO, WORKDEPT, SALARY
FROM     EMP
WHERE    SALARY = NULL;
```

5. Coding for Null – Comparing "Values"

Inconsistent Treatment?

- Not equal in predicates
  - That is, WHERE NULL = NULL is not true, but unknown
- "Equal" in terms of sort order though
  - That is, all nulls group together for ORDER BY and GROUP BY
- Also NULLs are considered to be equal in the following situations:
  - When duplicates are eliminated by SELECT DISTINCT or COUNT(DISTINCT column)
  - In unique indexes without the WHERE NOT NULL specification
- Is this inconsistent?
  - Would you really want a separate row for each null in your output for GROUP BY and ORDER BY?

5. A Few More Things About NULL

What is the result of NULL/0?

AVG, COUNT DISTINCT, SUM, MAX and MIN omit column occurrences that are set to null.

- AVG(COLA) not necessarily equal to SUM(COLA)/COUNT(*)

Consider this SQL:

```
SELECT  SUM(SALARY)
FROM    EMP
WHERE   DEPTNO > 999;
```

- Even if SALARY is defined as NOT NULL this can return a NULL
  - When there is no DEPTNO greater than 999

6. Avoid Bachelor Programming Syndrome

Proper Application Development Technique

Fear of COMMITing

- Plan and implement a COMMIT strategy
  - or experience TIMEOUTs and DEADLOCKs

6. COMMIT Guidelines

Application requirement	COMMIT recommendations
No concurrent access required and unlimited time for reprocessing in the event of an ABEND	Code program for COMMITs, but consider using a parameter to control commit frequency
No concurrency required but limited reprocessing time	COMMIT in batch approximately every 5 minutes or so
Limited batch concurrency required; no concurrent online activity	COMMIT in batch every 1 to 5 minutes (more frequently to increase concurrency)
Online concurrency required	COMMIT in batch every 5 to 15 seconds

Your mileage may vary...

7. Know the SQL Tweaks

There are shorthand tweaks that can be applied to SQL statement to "coerce" the optimizer to make specific decisions:

- OPTIMIZE FOR 1 ROW
  - Tells the optimizer that the intent is to retrieve only a small amount of data
  - Should eliminate list prefetch
  - Might cause the optimizer to eliminate sorts (especially for joins)
- No Op -- +0, -0, /1, *1, || ''
  - Arithmetic expressions are non-indexable
  - By adding the no-op to the predicate you do NOT want DB2 to choose an index for you can help to convince DB2 to choose a different index
  - Non-column expressions are indexable (as of V5), but IBM has made an exception for the no-ops

7. Example of a No-Op SQL Tweak

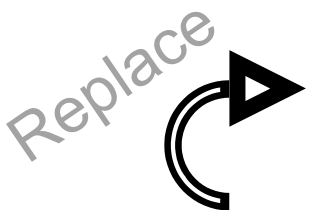
In this example, concatenating an empty string in the last predicate of the SQL statement disables matching index access but leaves the predicate Stage 1:

```
SELECT   EMPNO, WORKDEPT, EDLEVEL, SALARY
FROM     EMP
WHERE    EMPNO BETWEEN '000100' AND '000570'
AND      WORKDEPT > 'A01' CONCAT '';
```

7. Another Type of SQL Tweak

Scalar Functions

- Specifying a scalar function on a column renders the predicate non-indexable
- Can work like a no-op (unless an index is available on the expression)
- Example, if two indexes exist, one on LASTNAME and one on SALARY, and DB2 is choosing the SALARY index instead of LASTNAME:




```
SELECT    EMPNO, FIRSTNME, LASTNAME
FROM       EMP
WHERE    LASTNAME LIKE 'M%'
AND           SALARY > 50000.00;

AND           DEC(SALARY,7,2) > 50000.00;
```

7. Tweaking SQL - continued

What about join order?

- Sometimes you want to influence DB2 to choose a different order for the tables being joined
- Usually, we want the table with the smallest number of rows that qualify to be the starting table
  - You can add more predicates to make the optimizer think the number of qualifying rows have been reduced

    -or-
  - You might make the join predicate on TableA non-indexable

```
WHERE    (A.COL = B.COL OR 0=1)
```

8. Understand the Difference Between Dynamic SQL and Static SQL

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.

8. Static vs. Dynamic SQL Considerations

Criteria when 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:

    Non-uniformly distributed. (e.g. cigar smokers skews male)

    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.

8. Static vs. Dynamic SQL Considerations (continued)

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

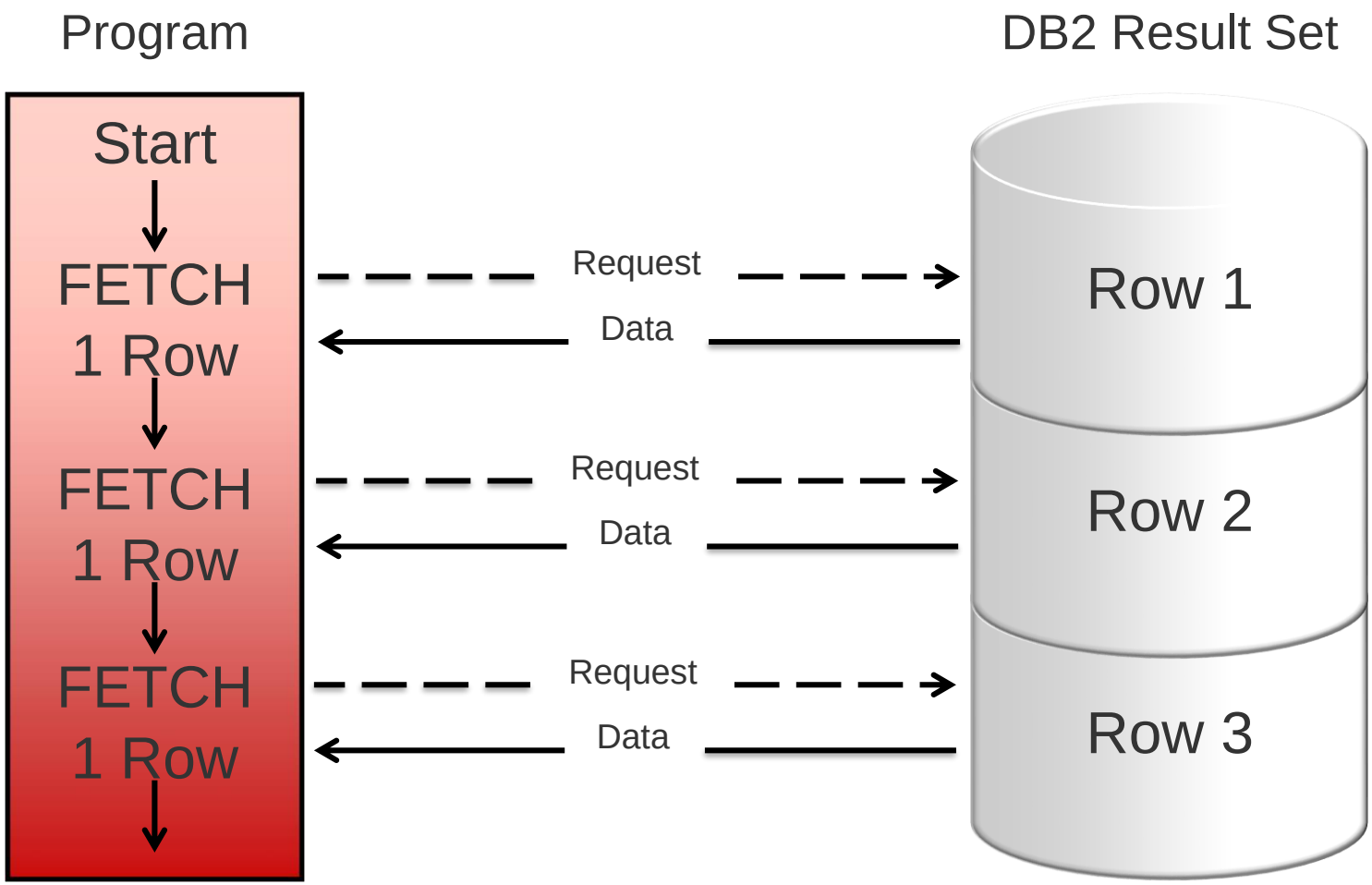
- Repetitious Execution
  - As the frequency of execution increases, favor static SQL (or perhaps dynamic SQL with local dynamic statement caching (KEEPDYNAMIC YES).
  - The cost of PREPARE becomes a smaller percentage of the overall run time of the statement the more frequently it runs (if cached prepare 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.

9. Learn to Code a Multi-Row FETCH

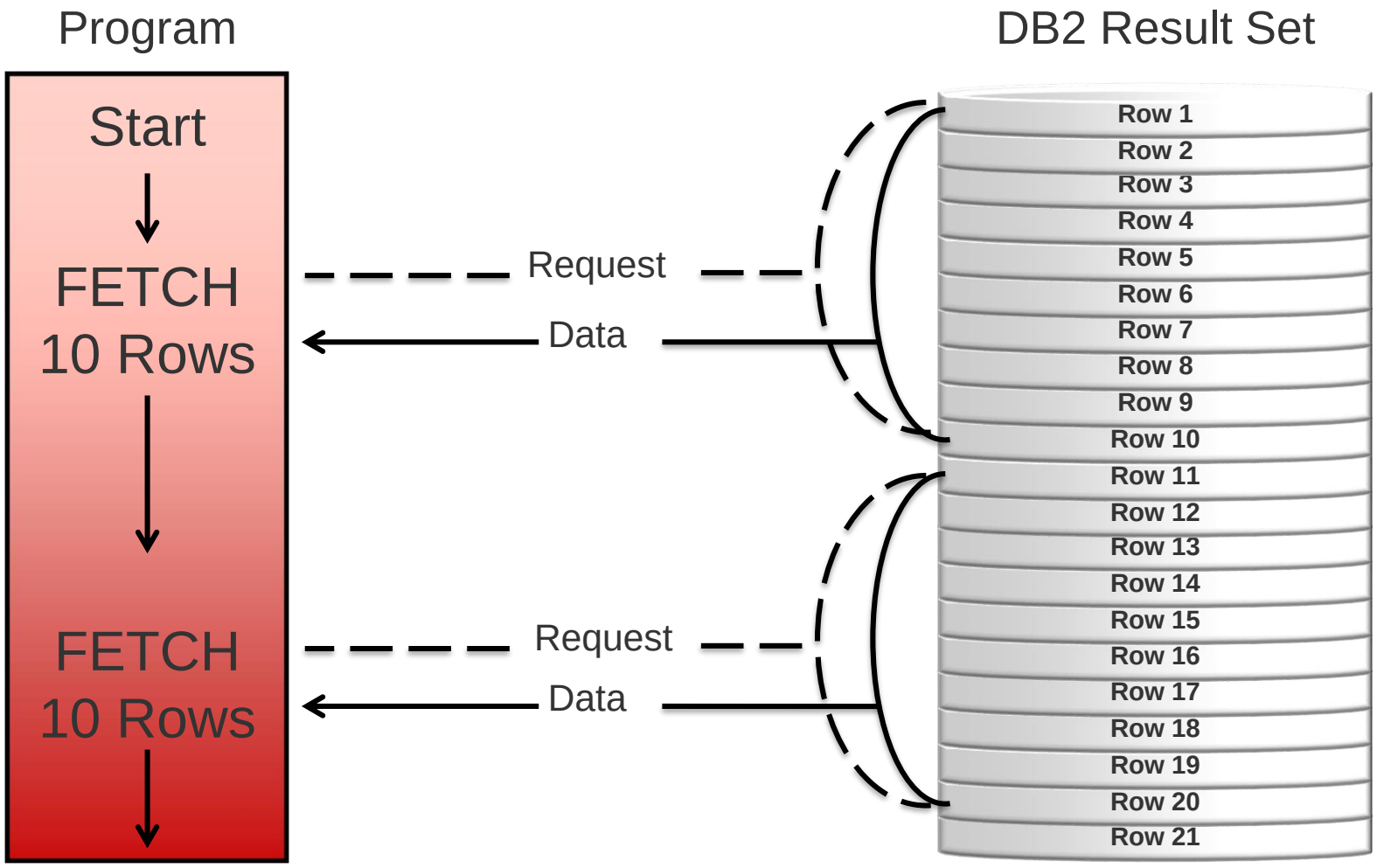
With multi-row Fetch, instead of obtaining data a row at a time to the program, data is obtained in blocks

- That is, multiple rows are obtained into an array in the program
- The program then must navigate through the array and request an additional block of rows when the current block has been processed
- Let's look at high-level depictions of how a multi-row FETCH differs from a traditional, single row FETCH cursor

Typical DB2 Single-Row FETCH



Example DB2 Multi-Row FETCH



9. Benefits of Multi-Row FETCH Over Single Row

CPU can be greatly reduced because moving data across address spaces is expensive

- With a multi-row FETCH multiple rows are moved with one operation

More coding is required in the application program to enable

- Unless it is a distributed program using Block Fetch
- Block Fetch is a distributed technique for more efficiently blocking fetched rows; when data is blocked, less overhead is required as data is sent over the communication lines
  - As of DB2 V8, Block Fetch will automatically use multi-row FETCH
  - Code FOR READ ONLY clause and BIND specifying CURRENTDATA NO to ensure that your distributed read only cursors use Block Fetch.

Depending on each particular case, a number between 10 rows and 100 rows per SQL can be a good starting point for multi-row operations… sweet spot between 100 and 1000 rows.

- Between 50% and 60% performance improvement has been seen.

10. Keep Current

One of the biggest detriments to DB2 performance and application developer efficiency/effectiveness is lack of knowledge about what DB2 can do!

As new versions of DB2 are released, be sure to train yourself on the new SQL, functions, and techniques that can improve:

- Application performance
- Ease of use
- More efficient coding practices

The Top Ten Things for DB2 DBAs

The following countdown will focus on tips, techniques, and tactics to improve the administration of DB2 for z/OS systems.

The list of "things" for DBAs will focus more on HOW the job is done than actual DBA practices and procedures…

- But some will sneak in there…

1. Know Database Design Theory *AND* Practices

Data modeling, normalization and database design are necessary to design useful, efficient databases

No kludging; use the features of DB2 as they were designed and meant to be used. Examples:

- Use DATE, TIME, and TIMSTAMP for chronological data
- Use numeric data types for numbers, character data types for alphanumeric
- Use multiple buffer pools; IBM does not provide 60 buffer pools in order for you to cram all of your data into BP0 (or just a few pools)
- Use referential constraints (declarative RI) and CHECK constraints instead of using programs to enforce constraints

Database constraints improve data integrity.

- When RI, CHECK constraints, and triggers are defined to the database then DB2 is receptive to changes in the state of the data and manages them appropriately.
- Without constraints data quality WILL suffer.

1. Know Database Design Theory *AND* Practices *(cont.)*

Avoid defaults.

- Understand every option at your disposal before choosing any parameter. Use the one that best matches the needs of the database, the application, or the situation.
- Explicitly code the parameter instead of letting it default even if it is the default value that you are choosing.

Free Space

- Choose PCTFREE and FREEPAGE values wisely.
- Don't just default everything to PCTFREE 10 and be done with it.
- Static data does NOT require free space.

Plan for data purging and archiving

- Because not all data is active and if you do not plan for removing it from the database it will NEVER go away.

1. Know Database Design Theory *AND* Practices *(cont.)*

Compression

- Not just for very large objects!
  - Consider compressing table spaces having more than 50 pages.
- Compression can improve application performance:
  - More rows per page
  - Data sticks around in the buffer pool longer
- Use the DSN1COMP utility to evaluate compression gains.

Universal Table Spaces…

- …are the future of DB2
- Use range-partitioned where you used to use classic partitioning
- Use partition-by-growth where you used to use simple and segmented

2. Know Difference Between Importance & Frequency

What is the most important thing that a DBA does?

- Did you just say performance management or tuning?

W R O N G

- The absolute most important thing that a DBA does is ensuring the integrity of the data.
  - Backup and recovery
  - Accuracy
  - Availability

3. Don't Be a Hermit

Many DBAs have reputations as curmudgeons… this is NOT a good thing.

- You do not want developers to be afraid to approach the DBA for assistance.
- Walk around; learn what is going on at your
- Talk to the developers and managers
- Share your knowledge and experience
- Be viewed as helpful

4. Be Careful of Standards

"I followed the corporate standards to the letter, but it doesn't work," complained the frustrated analyst.

"The best thing about standards," said the DBA, "is that there are so many of them to choose from."

Lesson: Although it is advisable to create DB2 standards, for every standard there is a reason to make an exception.

The more laws & restrictions there are the poorer people become.

-Lao Tzu

5. You Can't Know Everything

So instead, know where to look!

- DB2-L
- Google is your friend!
- IBM Knowledge Center (Online)
  - https://www.ibm.com/support/knowledgecenter
- DB2 Manuals
  - http://www-01.ibm.com/support/docview.wss?uid=swg27047206

And who to call for assistance

- Internally at your company
- Externally in your network of contacts

6. The Tuning Progression

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

Problem Resolution ↓

Severity of Problem ↑

7. Basic Tuning Rules

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

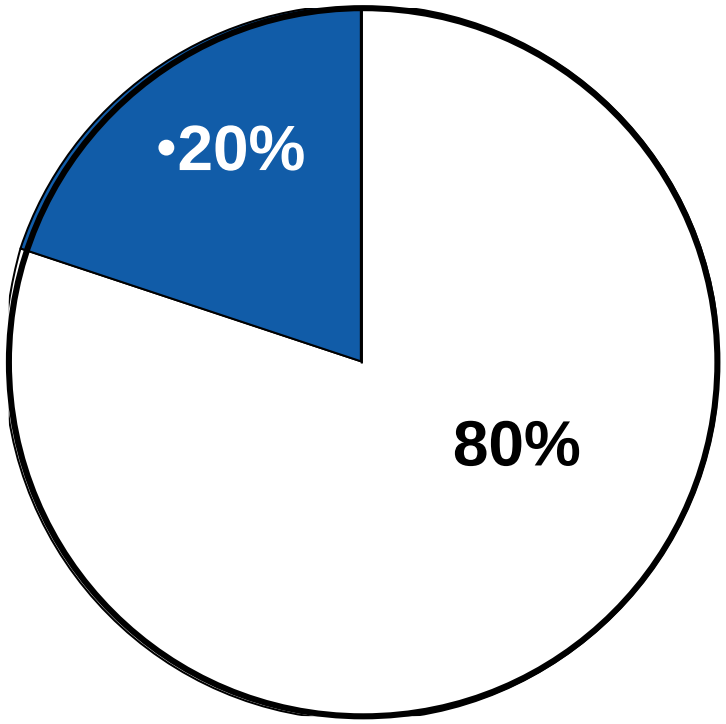
- 20% of your DB2 applications cause 80% of your problems

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**



8. Remain Calm… Do Not Rush to Solutions.

"Oh, I've seen this problem before. All I have to do is to add an index like before and…" said the intern excitedly.

"To a hammer all problems look like nails," replied the DBA.

Lesson:

- Do not rush to a solution. Take in all there is to learn and though you should rely on your past experience, do not force fit that experience to every problem.
  - Every day is an opportunity to learn something new.
  - Humility is a quality that DBAs would do well to exhibit.

To know, yet to think that one does not know is best; not to know, yet to think that one knows will lead to difficulty.

8. Remain Calm… Sometimes Do Nothing!

When all around you is in flux, sip your coffee and contemplate the situation.

- Not every *problem* requires a solution.

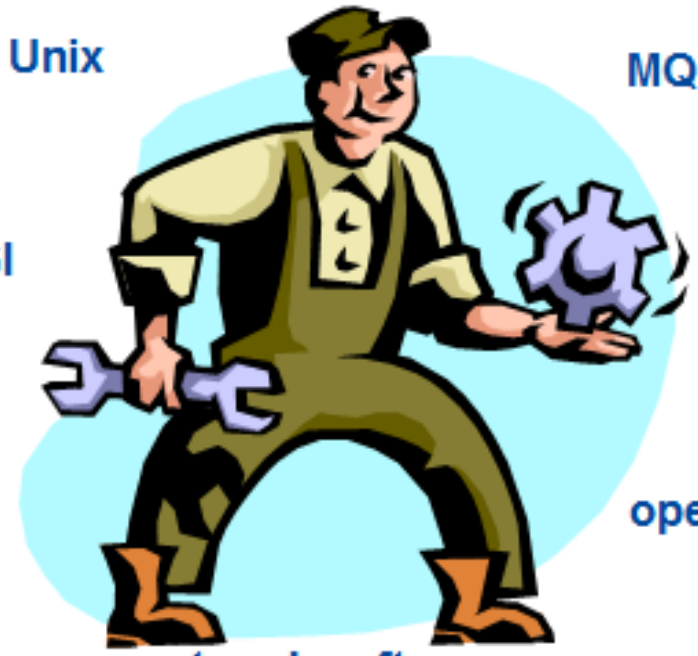
Sometimes the proper course of action is to do nothing. This can be difficult to endure.

- Some "problems" solve themselves…
- Or at least the problem-makers figure out how to solve things themselves.

Pure in heart, like uncut jade,
he cleared the muddy water
by leaving it alone.

-Lao Tzu

9. Don't Fear Being a Jack-of-all-Trades[1]



[1]...and a master of some!

https://datatechnologytoday.wordpress.com/2013/09/07/a-good-dba-should-be-a-jack-of-all-trades/

10. Document and Automate

Write Down Everything

- If you document the processes used to resolve problems and overcome challenges you can be prepared when you encounter a similar problem in the future.
- It is better to read your notes than to try to re-create a scenario from memory.

Keep Everything

- It is a good practice to keep everything you come across during the course of performing your job.
- If not, it always seems like you'll need that stuff the day after you threw it out! I still own some manuals for DB2 Version 2.

Automate

- Why should you do it by hand if you can automate your DBA processes?

The Top Ten Things for Managers (re: DB2)

The following list will focus on tips, techniques, and tactics to improve the way in which managers interact with and manage DB2 professionals.

We won't spend as much time on this list because sometimes you just cannot change management.

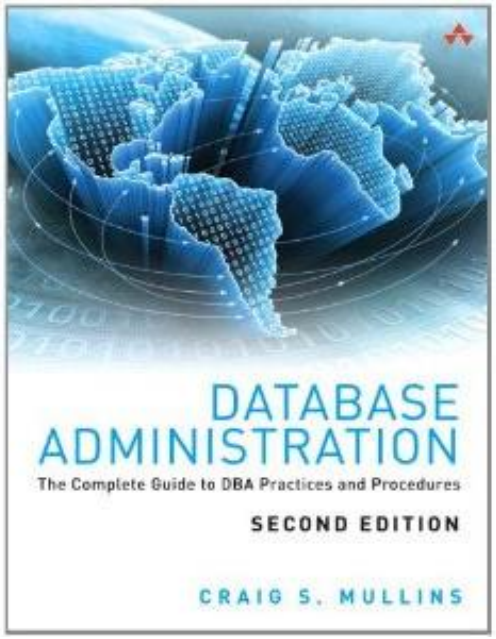
Generally speaking, be supportive and avoid cliches…

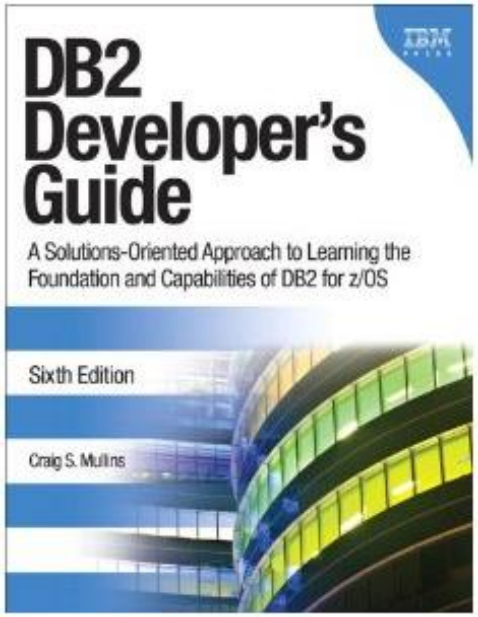
Top Ten Things Managers Can Do

1. Treat programmers/DBAs as individuals – there is no magic bullet for treating every employee the same way.
2. Be available but not omnipresent.
3. Avoid clichés when communicating.
4. Never tell anybody to… "work smarter, not harder."
5. Know what you don't know; You are no longer technically up-to-date; know this & don't pretend otherwise.
6. Be the trusted source for knocking down barriers to success.
7. Embrace educational opportunities for your staff.
8. Limit red tape and minimize the minutiae your staff has to deal with.
9. Trust… but verify.
10. Read Dilbert and don't be the PHB

Contact Information

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





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

http://www.MullinsConsulting.com

craig@craigsmullins.com

Phone: (281) 494-6153

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