SQL Performance in Today's Digital World

Sheryl M. Larsen

bmc

Executive Briefing Center

Analyzing Online and DDF SQL for Optimal Response Times

COLID	PROGRAM	QUERYNO	NOTES	
EDAP7610	DB2V9PRM	N/A	More popular than CIMB4500 and more GETPAGES Has the same problem as no current or future index can help this query. **Needs new index**	

Interval Date => 11/20/12 Interval Time => 07:00:00 Elapsed Time => 08:00

```
CREATE IXPAMH10 ON TBPAMH AS
(FST_DOS_DT ASC
, DIAG_CD_1 ASC
, PROV_TYP_CD ASC
, SUB_CONT_ID ASC)
```

Stage	Index	CARD	PREDICATES	Filter Factor
Match	1	10,624	FST_DOS_DT >= '2012-01-01'	0.01
1	2	11,392	DIAG_CD_1 LIKE	0.01
1	3	80	PROV_TYP_CD <>	0.98
1	4	669,780	SUB_CONT_ID NOT IN	1.00

Buffer Manager Activity

GETPAGE	**-> 33647744**	GETPFAIL	-> 0
SYNCREAD	-> 621400	SPFETCH	-> 0
LPFETCH	-> 2	DYNPFETCH	-> 1666399
PFPAGES	-> 17872912	PAGEUPDT	-> 61806
IMWRITE	-> 0	FETCHED	-> 11232
INSERTED	-> 0	UPDATED	-> 0

```
SELECT DISTINCT PAMH.SUB_CONT_ID, PAMH.PROV_TYP_CD, PAMH.MBR_NBR,
  PAMH.DCN_ID, PAMH.TOT_CHG_AMT, PAMH.TOT_ALWD_AMT, PAMH.TOT_PMNT_AMT,
  PAMH.DIAG_CD_1, PAMH.DIAG_CD_2, PAMH.DIAG_CD_3, PAMH.FST_DOS_DT,
  PAMH.PROC_DT, PAMH.CAUSE_CD, PRV.PROV_ID, PRV.PROV_CD, PRV.PROV_NM_T,
  PRV.ADDR_1_T, PRV.ADDR_2_T, PRV.PHN_NBR, PRV.PROV_IRS_ID, VWC.MKT_GRP_CD,
  COVD.EFF_DT, COVD.TERM_DT, EXD.CLM_STAT, VWC.POT_CD, SPC.PROV_SPCL_CD,
  SPC.PROV_SPCL_T
  FROM
       CIMP.TBPAMH AS PAMH LEFT OUTER JOIN CIMP.VWCCPMS0 AS VWC ON (PAMH.SUB_CONT_ID =
VWC.SUB_CONT_ID AND PAMH.MBR_NBR = VWC.MBR_NBR
    AND PAMH.DCN_ID = VWC.DCN_ID) LEFT OUTER JOIN CIMP.TBEXDESC AS EXD ON (EXD.CLIENT =
'03' AND PAMH.EX_CD = EXD.EX_CD) LEFT OUTER JOIN CIMP.TBPRVMST AS PRV ON (PAMH.PROV_ID =
PRV.PROV_ID AND PAMH.PROV_CD = PRV.PROV_CD) LEFT OUTER JOIN CIMP.TBSBCOV AS COV ON
(PAMH.SUB_CONT_ID = COV.SUB_CONT_ID) LEFT OUTER JOIN CIMP.TBSBCOVD AS COVD ON
(COV.SUB_CONT_ID = COVD.SUB_CONT_ID) LEFT OUTER JOIN CIMP.TBPRVTCD AS TCD ON
(TCD.PROV_TYP_CD = PRV.PROV_TYP_CD) LEFT OUTER JOIN CIMP.TBSPCLCD AS SPC ON
(SPC.PROV_SPCL_CD = PRV.PROV_SPCL_CD AND SPC.PROV_TYP_ID = TCD.PROV_TYP_ID)
WHERE COVD.TERM_DT >= '2012-11-20' AND COVD.TERM_DT <> COVD.EFF_DT
    AND PAMH.FST_DOS_DT >= '2012-01-01'
    AND VWC.MKT_GRP_CD IN ('C0', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7',
  'C8', 'C9')
    AND ((PAMH.DIAG_CD_1 LIKE ('493%')) OR (PAMH.DIAG_CD_1 LIKE ('465%'))
    OR (PAMH.DIAG_CD_1 LIKE ('466%')) OR (PAMH.DIAG_CD_1 LIKE ('487%'))
    OR (PAMH.DIAG_CD_1 LIKE ('786%')) OR (PAMH.DIAG_CD_1 LIKE ('480%'))
```

Investigating DB2 Batch Jobs to Reduce Overall Runtimes

We found a ***SINGLE job*** ***which was costing 7% of their monthly MLC***!

– traced this back to a ***single individual*** (lovely lady) who had NO idea her queries were driving up the MLC peak by 7%.

We found a ***SINGLE job*** ***which was costing 10% of their monthly MLC***!

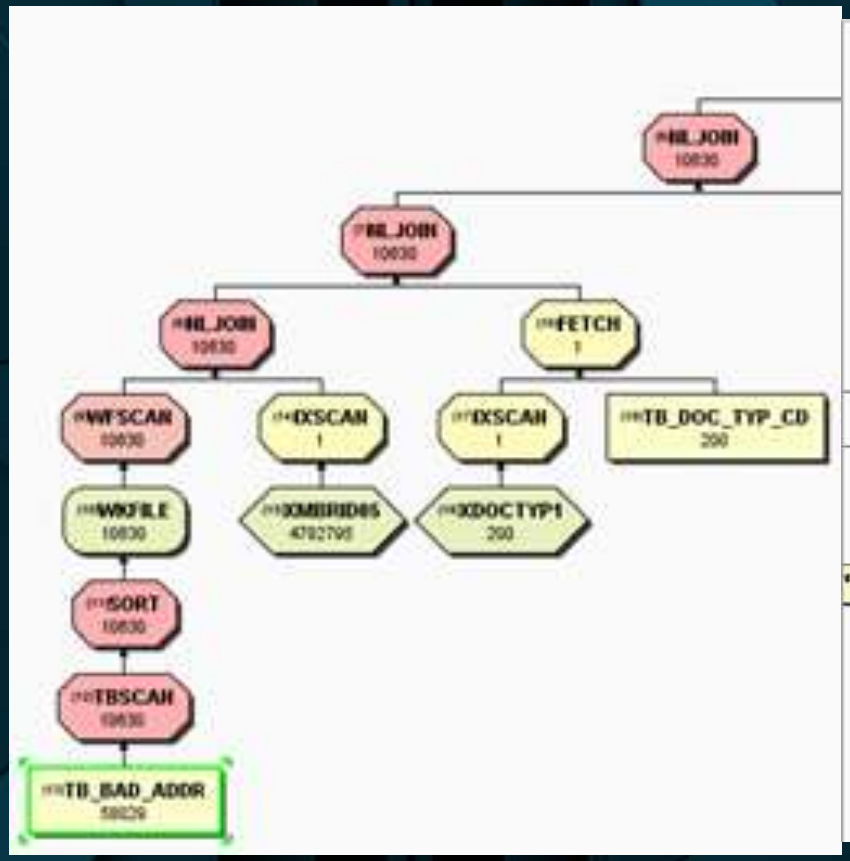
– delayed the job 3 hours and reduced the MLC by 10% reducing the MLC bill by $10,000/month

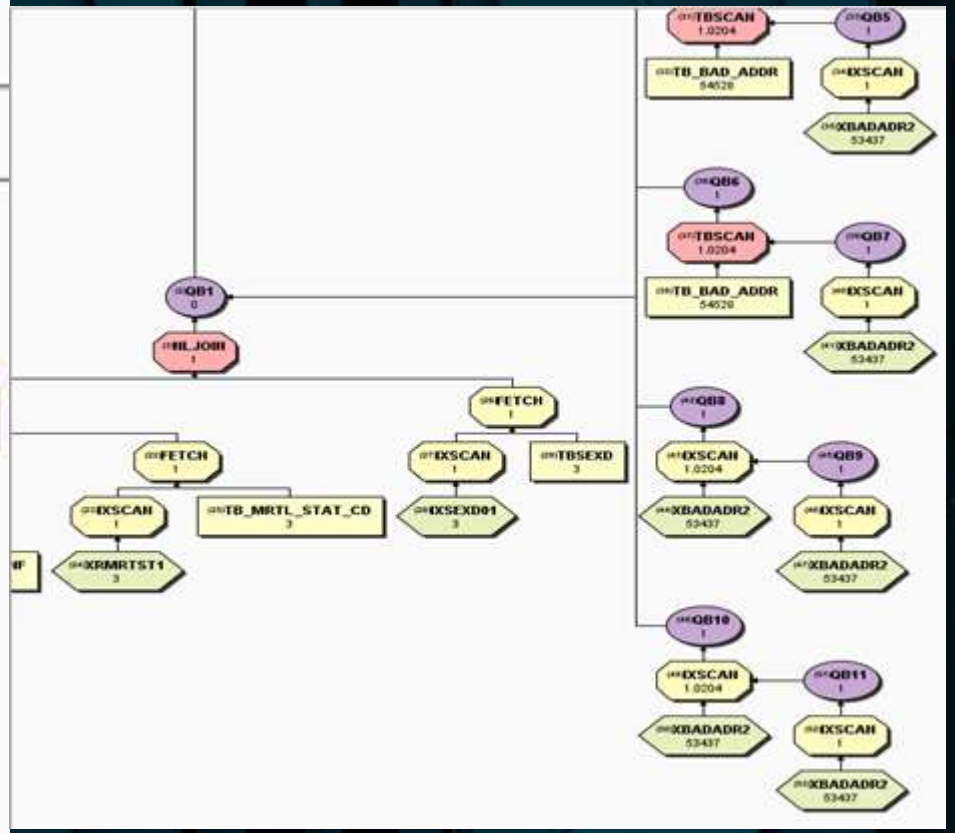
Exploring Access Path Efficiency by Examining the EXPLAIN

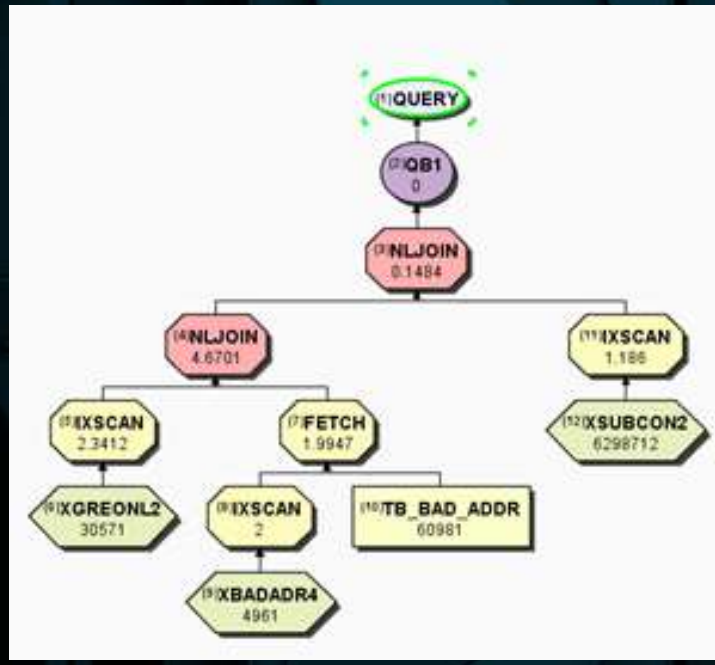
COLLID	PROGRAM	QUERYNO	NOTES	
M		1377080	Query did 8 million GETPAGES in 4 executions on TB_BAD_ADDR. Data Studio showed misleading filter factors. Verified Detector is telling the truth by running 2 queries. The Cache verifies Detector's output for those expensive queries. Needs a new index on TB_BAD_ADDR and Histogram statistics. Rewrote query to use EXISTS subquery Rewrote again to use a table expression Rewrote again to only return a flag	Y

TB_BAD_ADDR Query has Misleading Stats
Update: **26 s: 177 ms** original query run time
499 ms Query rewrite run time with the new index
78 ms Query rewrite run time with a table expression
16 ms Query rewrite run time to only return a flag

Access Path Analysis

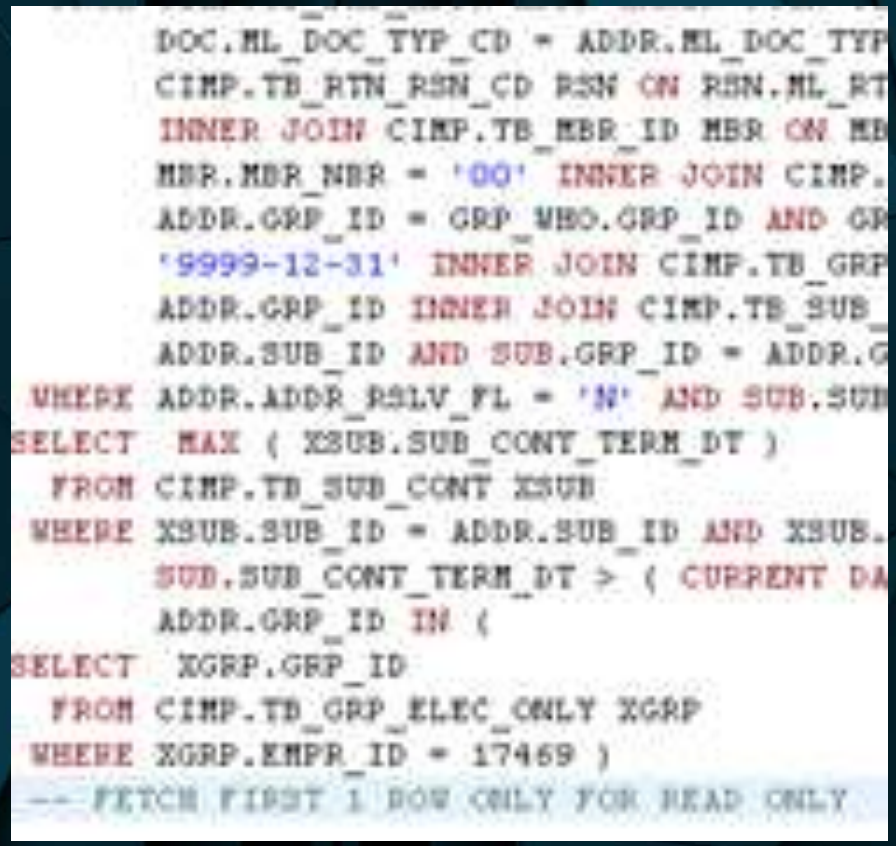






The larger the graph and the more rows involved, the more costly it is.

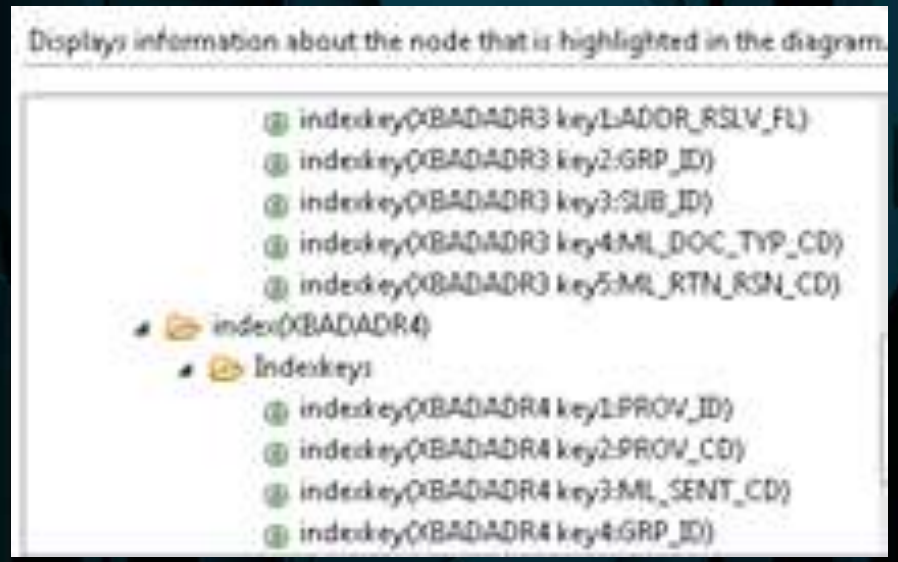
Code Analysis

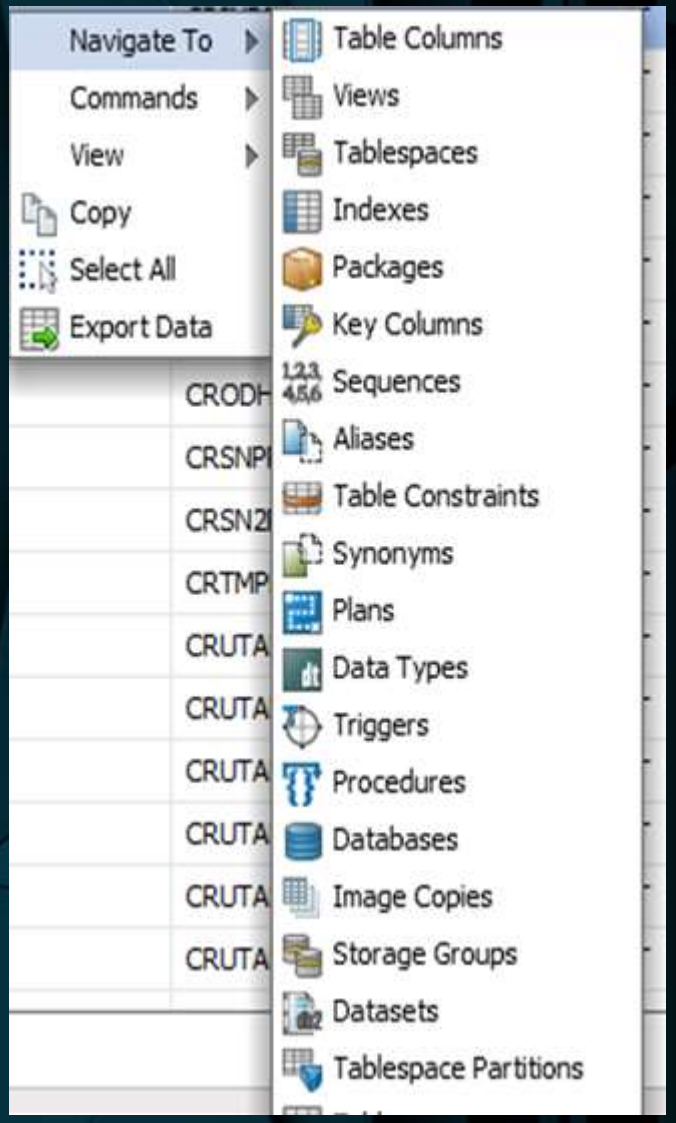


```
        DOC.ML_DOC_TYP_CD = ADDR.ML_DOC_TYP
        CIMP.TB_RTN_RSN_CD RSN ON RSN.ML_RT
        INNER JOIN CIMP.TB_MBR_ID MBR ON MB
        MBR.MBR_NBR = '00' INNER JOIN CIMP.
        ADDR.GRP_ID = GRP_WBO.GRP_ID AND GR
        '9999-12-31' INNER JOIN CIMP.TB_GRP
        ADDR.GRP_ID INNER JOIN CIMP.TB_SUB_
        ADDR.SUB_ID AND SUB.GRP_ID = ADDR.G
 WHERE  ADDR.ADDR_RSLV_FL = 'N' AND SUB.SUB
SELECT   MAX ( XSUB.SUB_CONT_TERM_DT )
  FROM CIMP.TB_SUB_CONT XSUB
 WHERE XSUB.SUB_ID = ADDR.SUB_ID AND XSUB.
        SUB.SUB_CONT_TERM_DT > ( CURRENT DA
        ADDR.GRP_ID IN (
SELECT   XGRP.GRP_ID
  FROM CIMP.TB_GRP_ELEC_ONLY XGRP
 WHERE XGRP.EMPR_ID = 17469 )
 -- FETCH FIRST 1 ROW ONLY FOR READ ONLY
```

```
SELECT 'Y'
FROM
    CIMP.TB_BAD_ADDR ADDR
    INNER JOIN CIMP.TB_GRP_ELEC_ONLY XGRP
                ON XGRP.GRP_ID = ADDR.GRP_ID
    INNER JOIN CIMP.TB_SUB_CONT SUB
                ON SUB.SUB_ID = ADDR.SUB_ID
                AND SUB.GRP_ID = ADDR.GRP_ID
WHERE
    ADDR.ADDR_RSLV_FL = 'N'
    AND
    XGRP.EMPR_ID = 17469
    AND
    (SUB.SUB_CONT_TERM_DT > (CURRENT DATE - 90
DAYS))
    FETCH FIRST 1 ROW ONLY
    FOR READ ONLY       ;
```

Scrutinizing Information Contained in the DB2 Catalog





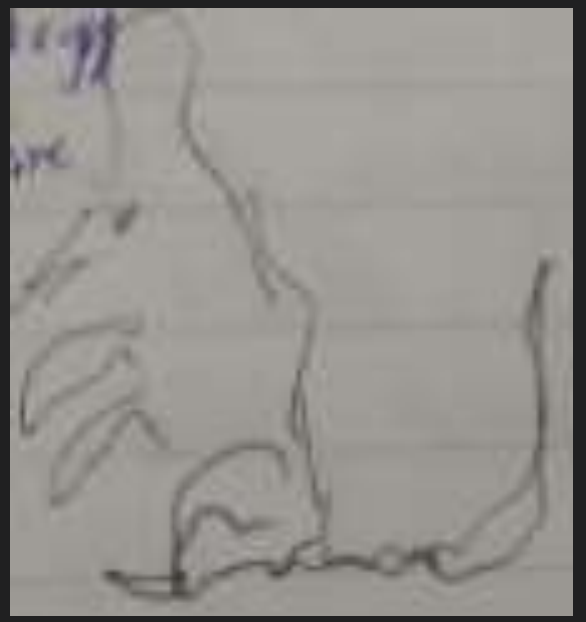
Monetize your savings with compares

Before and After

1. A REORG
2. Any SQL change
3. Any access path change
4. Any index change

Reboot Your Thinking – Paul Sloane

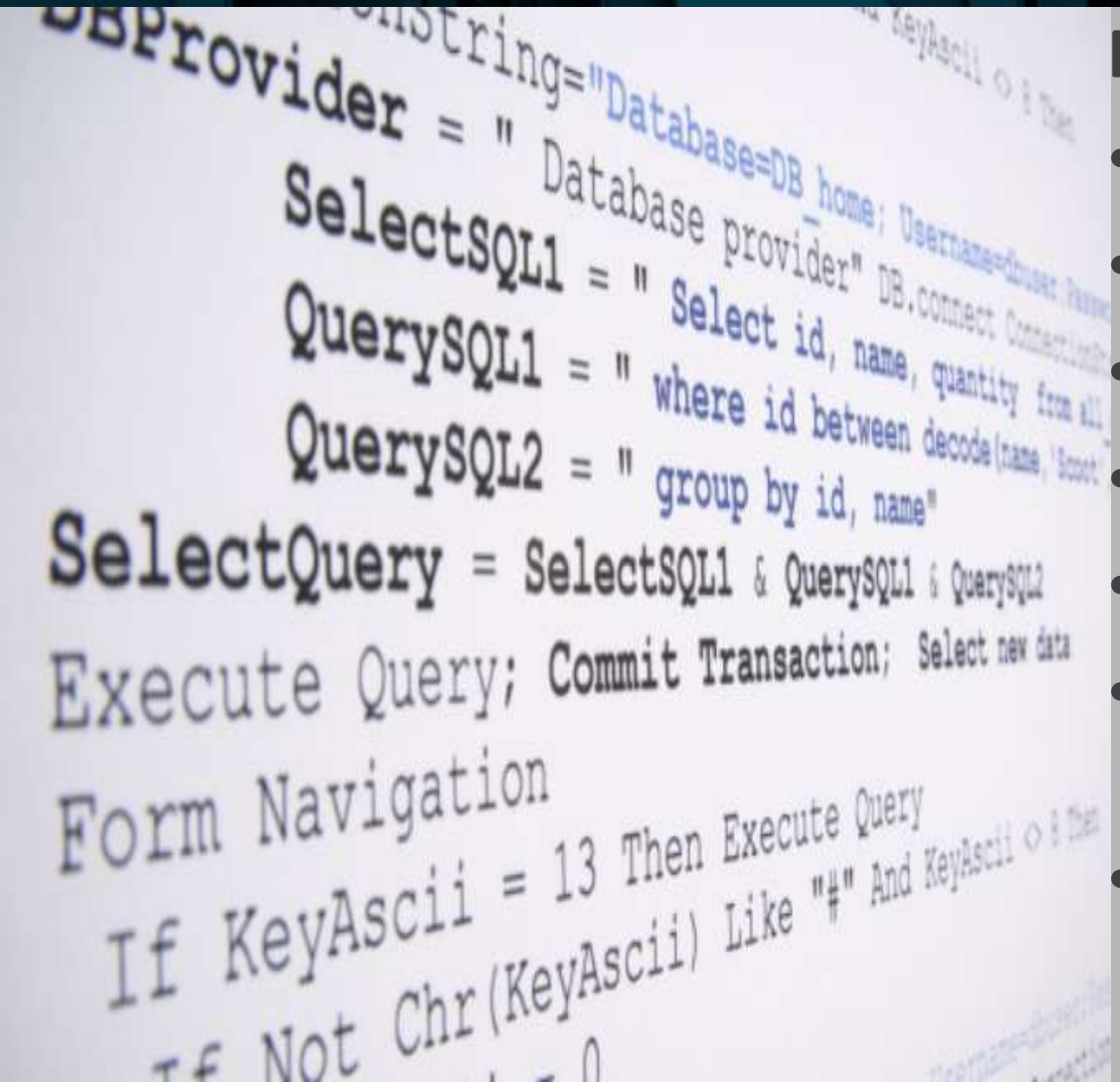
1. Check your assumptions
2. Break the rules
3. Ask searching questions
4. Deliberately take the opposite point of view
5. Generate many ideas
6. Look outside for ideas
7. Manage Risk
8. Empower your team to try new things



Future Proof Your SQL Performance

- Monitor workload impact with a light footprint
- Identify expensive queries, not the highest CPU
- Send alerts for degrading SQL day or night
- Get smart REORG advice
- Get smart index advice
- Track access path changes from migrations to new applications or DB2 upgrades across a workload
- Have many dashboards to enable a broader user audience

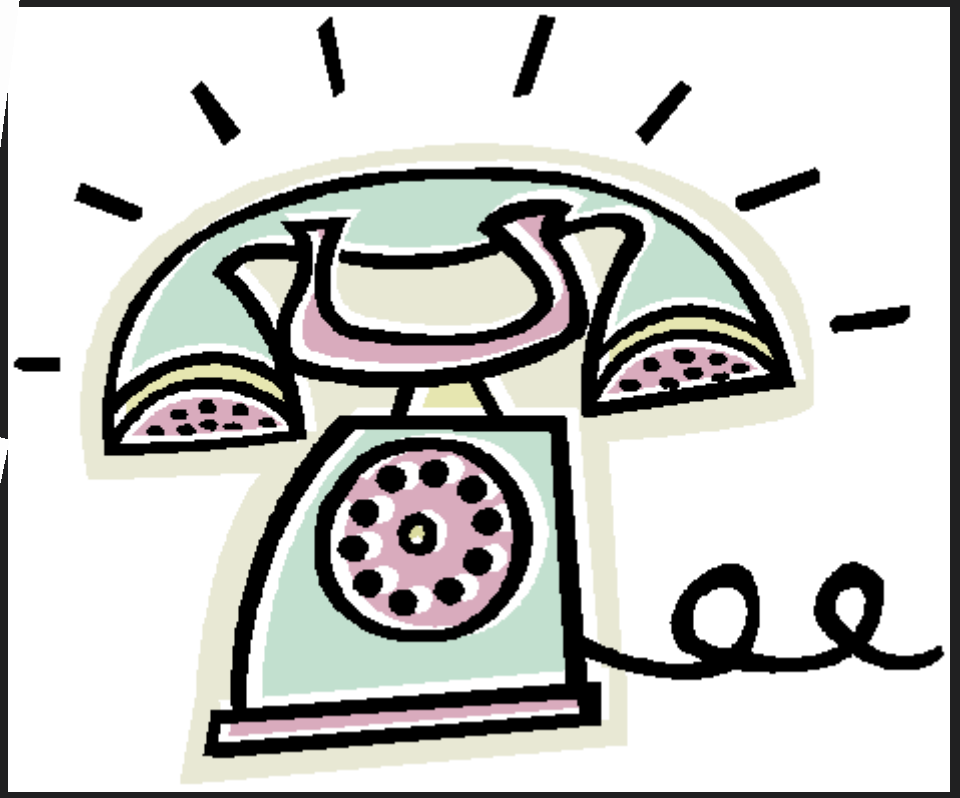
Agenda for SQL Performance Part 2



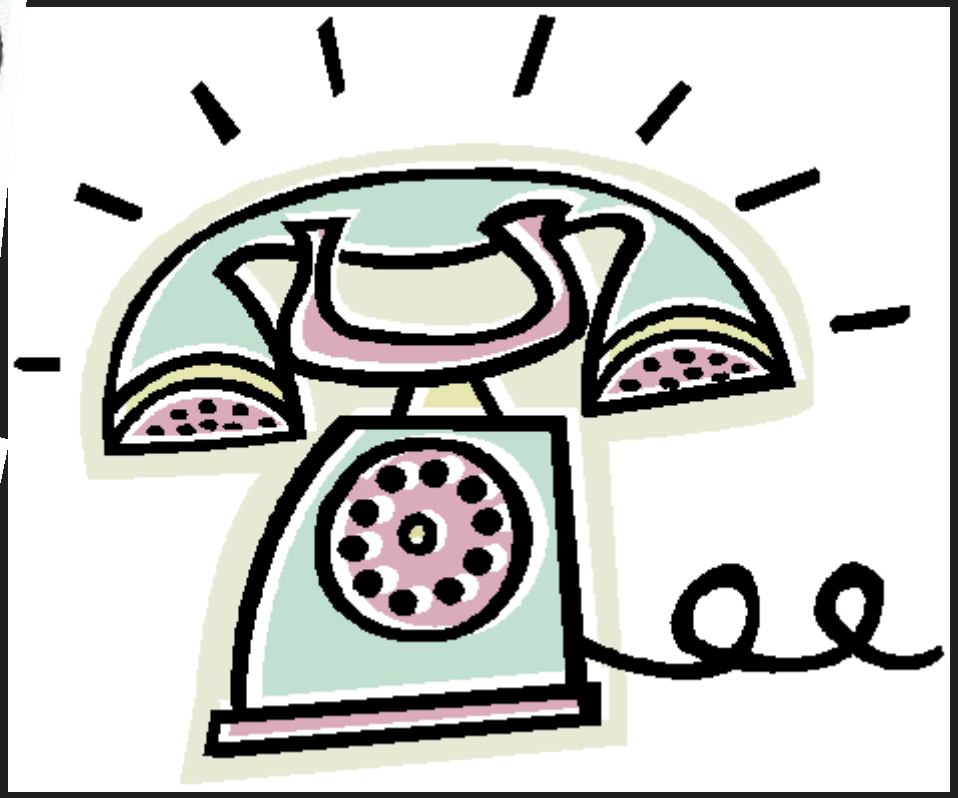
Part 2

- How do you know you have a problem?
- Is it solvable from a DB2 perspective?
- Getting to the bottom of your problem
- Better preparation to solve problems
- “Reactive Tuning”
- What about performance problems you DON’T know about?
- “Proactive Tuning”

How do you know you have a problem?



How do you know you have a problem?



How do you know you have a problem?

- **Something is doing something worse than it was before**
  - Or something is worse than expected
  - Or something

- **Sometimes these reports can be very vague**
- **And may even be inaccurate**

- **But it's always YOUR fault**
  - Until you can prove that it isn't

Is it solvable from a DB2 perspective?

- **More specifically, “Is this even a DB2 problem?”**
- **Performance problems can be caused by a myriad of things**
  - Network issues (including internet)
  - Transaction monitor issues (CICS or IMS/DC)
  - Operating system issues
- **Or perhaps this IS a DB2 issue after all**
  - But how to tell?
- **Our first step should be identifying possible areas to target**

BUT- Before we start

- Lets look at our performance database to see if we are chasing shadows
- What does historical performance look like
- Is it a problem or a perception?

- What do you mean
  “We don’t have a performance database”

BUT- Before we start

- Analysis is also an early indicator of “outside influence”
- If the numbers are not significantly different,
- The problem may well be outside of your control
  - Or even any of your colleagues
- How do you fix an internet performance problem, for example?

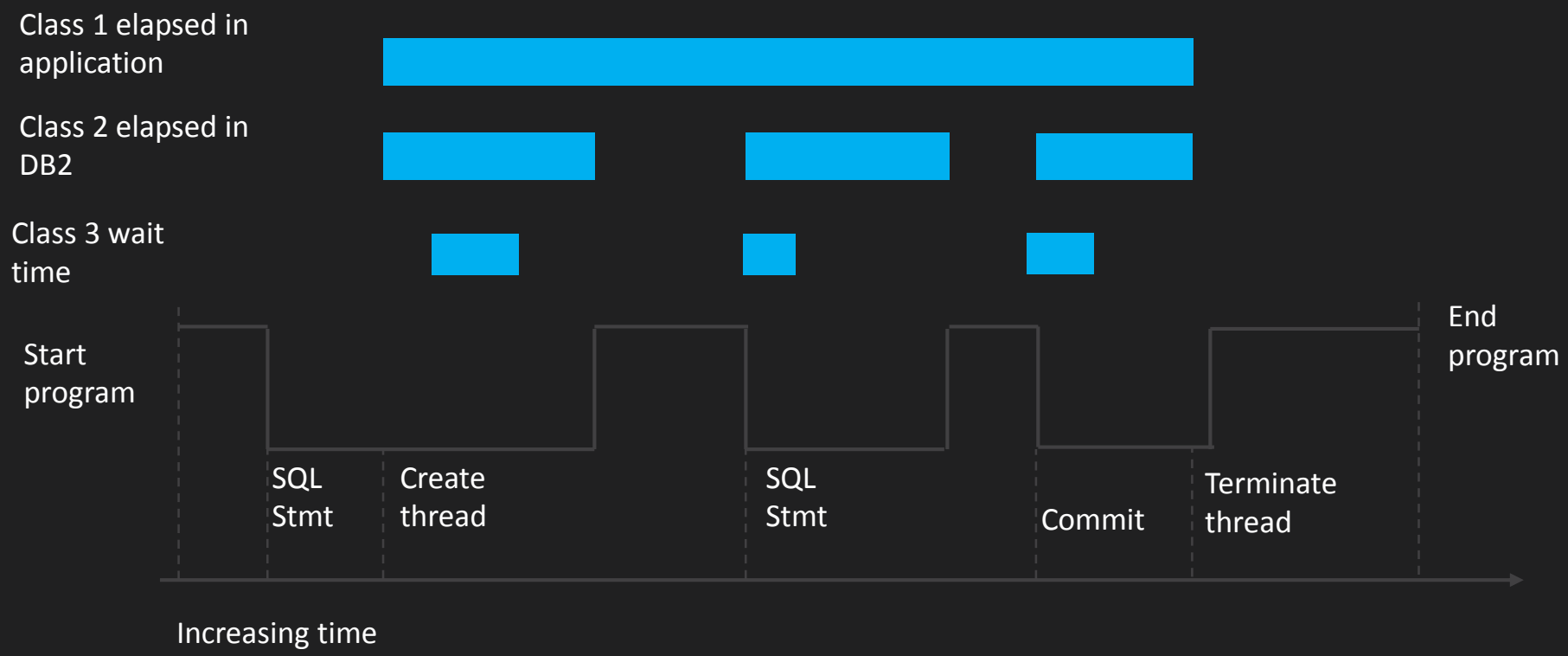
- Your users/customers may not be locally attached any more

How to get to the bottom of your problem

- **We need to identify that it IS a DB2 problem**

- **We can do that with THREE metrics**
  - Elapsed time
  - IN-DB2 time
  - IN-DB2-CPU time

Elapsed Time, IN-DB2 time, IN-DB2-CPU time



Elapsed time

- **If elapsed time is COMPARABLE, there is no problem**
- **If it's relatively HIGH, there IS a problem**
- **Somewhere within the mainframe ecosystem**
  - Typically we're not measuring end-user elapsed times
  - Only internal ones

IN-DB2 time

- **If time spent in DB2 is close to normal, but elapsed was high**
  - The problem lies OUTSIDE DB2
- **So look at CICS, IMS/DC, network/web etc**
- **If time spent in DB2 is HIGHER than normal, then the problem does lie within DB2**

IN-DB2-CPU time

- **If cpu time is close to normal, but total time was high**
  - Then we have a problem outside SQL
  - The access path probably hasn't changed, but the way DB2 is performing it might have
  - Perhaps we have more waits than usual?
    - For locks
    - I/O
- **If the cpu time has increased then DB2 has probably changed access paths**
  - And not for the better

To recap

- In only a few steps we have isolated where we (or someone else) should be looking
- Which sometimes is more than half the battle

- As a side note, the processing (cpu) time should be close to the elapsed time
  - If not, WHY not?

Preparing to solve problems

- **There ARE things you can do to prepare**
- **You know you will have performance problems to diagnose at some point**
- **Make life easier for yourself**
  - ALWAYS bind with EXPLAIN (YES)
    - Though EXPLAIN PACKAGE provides some mitigation after DB2 9
  - ALWAYS use Plan Management
    - Provides quick fallback for any post-rebind issues
  - PLEASE consider a Performance Database
    - Without one, you're just guessing

“Reactive Tuning”

- But this is reactive tuning
- Responding to a KNOWN problem
- In response to an external event (a complaint)

- Wouldn’t it be better to prevent problems?
- Or make things even BETTER than they are?

Problems you DON'T know about

- What if people are not complaining
- But things are worse than they could be
- Do you care?

- "It Depends" on how you are paying for those resources
- And if there is value in "being better"

Hidden performance problems

- You are using resources unnecessarily
- And perhaps paying more than you should
  - Either for measured cpu consumption
  - Or as part of your 4-hour peak
- If you use less, you could be paying less
- OR do MORE processing for the same money

“Proactive Tuning”

- **Proactive tuning has many benefits**
- **Traditional tuning tends to *assume* that things are “OK”**
- **You need a sophisticated tool to do analysis here**
  - BMC Apptune
  - CA Detector
  - IBM Query Monitor
- **All of which look for costly SQL executions**
  - Both individually
  - Or aggregated

Example

- ```
  SELECT COUNT(*)
  FROM SYSIBM.SYSDUMMY1;
  ```
- How costly can THIS be?
- Well, how often do you execute it?
- Is this any better?

```
SELECT COUNT(*)
FROM SYSIBM.SYSDUMMY1
WHERE 0 = 1;
```
```

Example

- A “traditional” DB2 monitor would never have uncovered this
- A VERY cheap statement
- Executed MILLIONS of times
- Wasting resources unnecessarily

Monetary tuning

- Looking for the big cpu consumers is OK sometimes
- It depends on what you are trying to achieve
- If you pay for ALL cpu, then great
- Otherwise there may be a better use of your time

Monetary tuning

- **Why not look for the highest cpu consumer**
  - In your Four Hour Peak period
- **Saving cpu here will speed transactions**
- **AND save money**
  - And continue to save EVERY month
- **And may even enable higher transaction throughput**

Monetary tuning

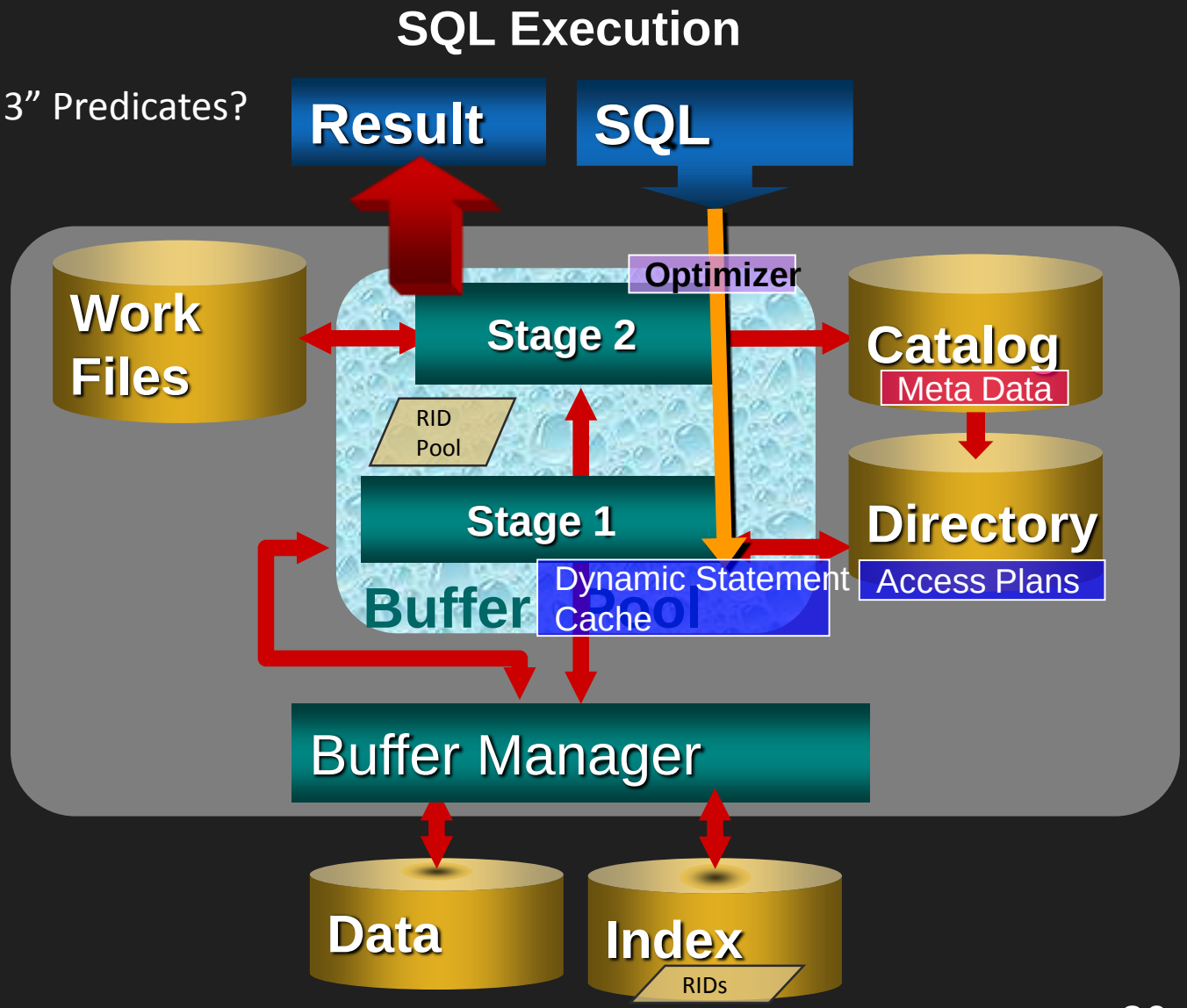
- **So you need to find statements ordered by cpu consumption**
- **At specific times of the month**
  - Assuming your peaks are consistent across months

Pre-emptive tuning

- How about looking for problems in the way SQL is coded?
- Preventing a problem is MUCH cheaper than fixing one
- This is easy with Static SQL
  - Less so with Dynamic
- And built around best practices
- That are CONSTANTLY reviewed

Pre-emptive Tuning

"Stage 3" Predicates?



Pre-emptive tuning

- **Look for stage 2 predicates**
  - Make them stage 1
- **Teach developers about stage 3**
  - “Do it in SQL if you can”
- **Look for indexability**
  - But don’t FORCE index access where it is not appropriate
- **Look for sub-optimal access paths**
  - But understand how the data is accessed

Use Math to Change the Optimizer's Mind

+0, CONCAT ' ' also –0, *1, /1

- Place no op next to predicate
- Use as many as needed
- Discourages index access, however, preserves Stage 1
- Can Alter table join sequence
- Can fine tune a given access path
- Can request a table scan
- Works at the predicate level

Math Example - Scan

SALES_ID.MNGR.REGION Index | MNGR Index | REGION Index

```
SELECT  S.QTY_SOLD
          , S.ITEM_NO
          , S.ITEM_NAME
FROM      SALE S
WHERE   S.SALES_ID > :hv-id +0
    AND   S.MNGR = :hv-mngr CONCAT ' '
    AND   S.REGION BETWEEN
           :hvlo AND :hvhi CONCAT ' '
ORDER BY S.REGION
FOR FETCH ONLY
WITH UR
```

- If you know the predicates do very little filtering, force a table scan
- Use a No Op on ***every*** predicate
- This forces a table scan
- FOR FETCH ONLY encourages parallelism
- WITH UR for read only tables to reduce CPU

Should this be Documented?

DISTINCT Table Expressions Example

- SELECT Columns
  FROM TABX, TABY,
  (SELECT DISTINCT COL1, COL2 …..
  FROM BIG_TABLE Z
  WHERE local conditions) AS BIGZ
  WHERE join conditions

- Optimizer is forced to analyze the table expression prior to joining TABX & TABY

Workload tuning

- Perhaps the Holy Grail of tuning?
- The chance to make EVERYTHING better
- Find hidden problems in an entire workload of SQL

Reminders

- **PLEASE bind with EXPLAIN (YES)**
  - EXPLAIN PACKAGE only helps after DB2 9
- **PLEASE use Plan Management**
  - Without it you have no fallback for any post-rebind issues
- **PLEASE consider a Performance Database**
  - Eliminate guesswork about the past
  - Or the future

Digital Transformation

Questions?

- ?

Sheryl Larsen
Sr. DB2 Product Specialist
President, Chicago DB2 Users Group
IBM Certified Application Developer - DB2 11 for z/OS
(224) 343-5427
sheryl_larsen@bmc.com

THANK YOU


bmc
Bring IT to Life