

Platform: DB2 for z/OS

Do's and Don'ts for Finding Performance Indexes

Dirk Johann

Senior Consultant/SOFTWARE ENGINEERING

Session: A05

May 24th / 10:00



Agenda

- Motivation
- How to improve performance
- Performance Indexes: what they are
- Procedures: best approach for finding them
- Information Sources: which sources are available
- Index Candidates: how to find them
- Catalog Statistics: why they are so important for index usage
- Summary

Motivation

Why optimize performance in the DB2 area?



- > Improve response times!
- > Save money!!
- > Be a hero!!!



How to improve performance of DB2 applications?

Several performance optimization methods:

- Application design and redesign
- SQL optimization
- DB2 subsystem tuning
- DB2 maintenance optimization
- Database design and redesign
 - Logical design
 - Database, Tablespace, Table, View (re)design
 - Index (re)design

What is a performance index?

A performance index is:

- an additional index with one or more columns on a table that
- improves the performance of one or more SQL accesses through other Optimizer decisions

This presentation deals with a method to manually determine such performance indexes that will improve the performance of existing applications.

Index types

The following index types must be distinguished:

- Indexes based on data models (RI, Unique)
- Partitioning Index
- Clustering Index
- Performance Indexes

Performance indexes offer the opportunity to optimize applications and application systems:

- > without intervention in the application
- > with simultaneous optimization of numerous accesses
(also Dynamic SQL workload)
- > normally without negative consequences

Example:

Table Structure:

```
CREATE TABLE TEST.TEST01
  (CONTRACT_NR      INTEGER  NOT NULL
  ,CUSTOMER_NR     INTEGER  NOT NULL
  ,INVOICE_TYPE    CHAR(08)  NOT NULL
  ,CONTRACT_SUM    SMALLINT NOT NULL
  ,CONTRACT_TYPE   CHAR(32)  NOT NULL
  )
IN TEST100.TEST01;
```

Example:

Index:

```
CREATE TYPE 2 UNIQUE INDEX TEST.TSTX011 ON  
TEST.TEST01  
    (CONTRACT_NR  
     ,CUSTOMER_NR)
```

```
USING STOGROUP SYSDEFLT  
    PRIQTY 480  
    SECQTY 96  
    CLUSTER  
    BUFFERPOOL BP0  
    FREEPAGE 0  
    PCTFREE 20  
    CLOSE YES;
```

Example:

Select:

```
SELECT COUNT (*) FROM TEST.TEST01  
WHERE CUSTOMER_NR BETWEEN 1000 AND 10000;
```

Cost:

```
STMT COST : 51.46  
MIL.SEC. : 26  
SERV.UNITS: 75
```



Non-Matching Index Scan

Example:

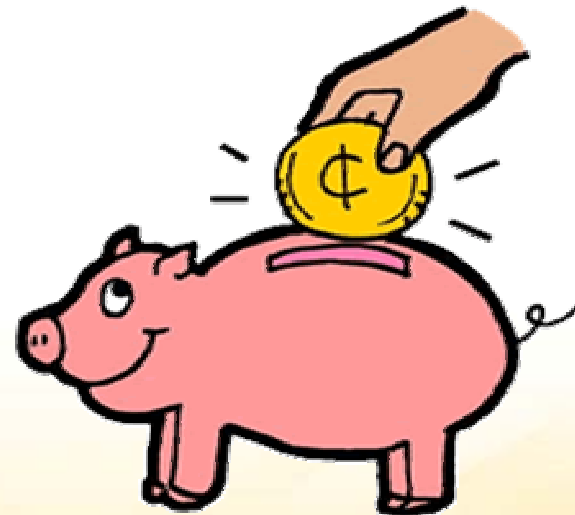
Additional Index:

```
CREATE TYPE 2 INDEX TEST.TSTX012 ON TEST.TEST01  
(CUSTOMER_NR)  
USING ... ;
```

Cost:

```
STMTCOST:    20.39  
MIL.SEC.   :  11  
SERV.UNITS:  30
```

Matching Index Scan



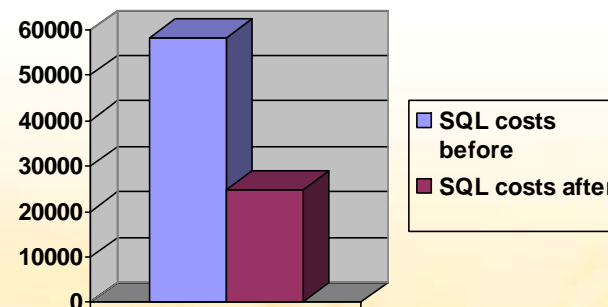
Minimum report to manager:

„For a transaction SQL, executed 200.000 times a day, we saved 15 ms of CPU time.“

Savings: (assume 1 second CPU time costs 0.056 \$)

200.000 execs/day x 15 ms x 0.056 \$ x 200 workdays

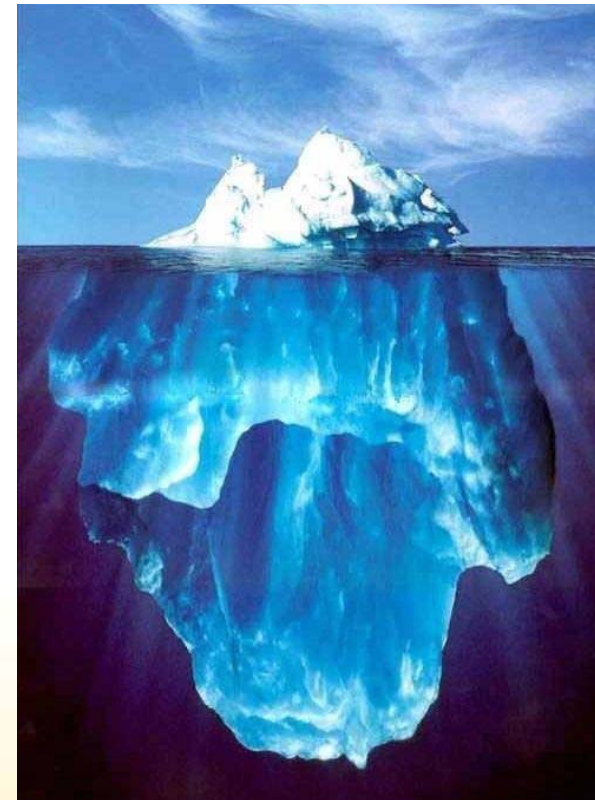
=> 33,600 \$ per year



Procedures (cont.)

Where to tune?

- Look at the heavy hitters
- Take the top consumers
- First tune the tips of the iceberg
- Then the next appearing tips



Procedures (cont.)

Classic program- or SQL-oriented procedures:

- Analyze one program / program group
- Determine the SQL performance (optional)
- Analysis of EXPLAIN data
- Determine meaningful performance indexes

Tools:

- Program monitor / Debugging tool
- EXPLAIN tool
- Paper / pen / brain (preferably the DBA part!!!)

Procedures (cont.)

Advantages / Disadvantages of classic procedures:

- + Easy to find good index candidates
- + Fast approach for optimizing single SQL
- + Information gathering very easy
- Negative effects on other applications possible
- Potentially results in too many indexes (at least one for every application)
- Dynamic workload will not be taken into consideration

Procedures (cont.)

Workload-based approach:

- Trace/monitor the workload to be optimized
- Determine the SQL performance
- Determine the optimization candidates
- Build a SQL / Table and Column matrix
- Determine meaningful/relevant performance indexes
- Verify and analyze with EXPLAIN

Tools:

- DB2 trace facility or DB2 monitor
- EXPLAIN tool
- Paper / pen / brain (in the DBA part!!!)

Procedures (cont.)

Advantages/Disadvantages of workload based approach:

- + Good for overall tuning
- + Consideration of complete workload
- + System wide tuning focus
- + Few indexes for ALL applications
- Information gathering can be tough
- More brain activity necessary

Procedures (cont.)

Why is the workload-based procedure recommended?

- Concentration on the essential problems
- Consideration of dynamic SQL and remote applications, (e.g., DB2 Connect)
- Optimization of the whole system
- Low effort / good results

But:

- Workload must be evaluated
- Determining the relevant information is tough

Information sources

The basic information can be obtained from:

-> DB2 Monitor (whatever available ...)

-> DB2 Performance Trace (-START TRACE (PERFM) ...)

Determined should be:

- Performance information about SQL statements,
e.g., Average Runtime / Total Runtime
- SQL statement text (for dynamic SQL)
- Information about tables that are accessed

DB2 performance trace

Characteristics:

- DB2 standard tool to obtain detailed information
- Available with every DB2 system with NO additional costs!
- Using IFCIDs, need-based information can be determined
- Using a trace also captures statement text from non-static SQL

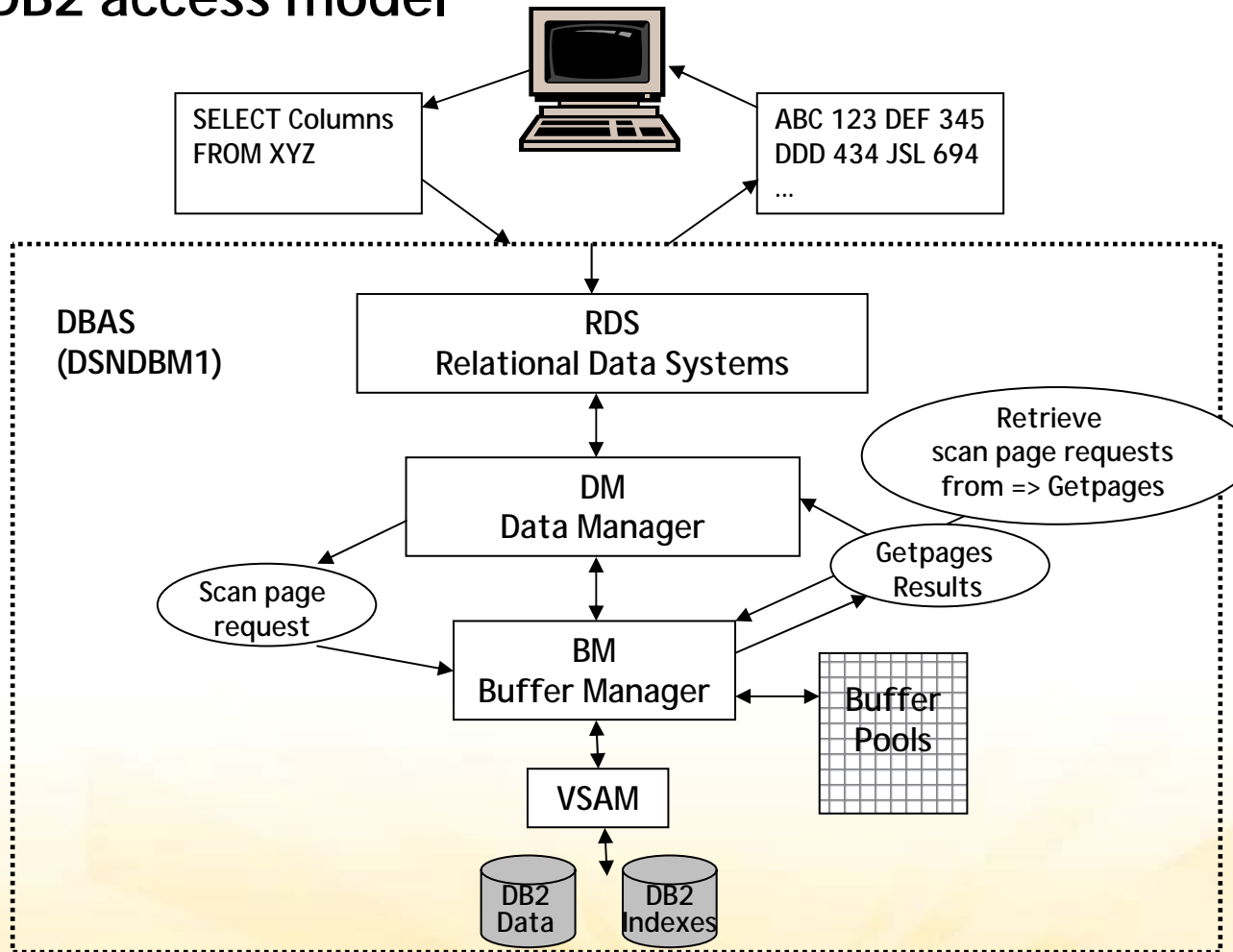
Advantages:

- General availability
- Also for dynamic SQL and DB2 Connect applications
- Enables consideration of machine independent data
(scan and getpage jobs versus CPU time and elapsed time)

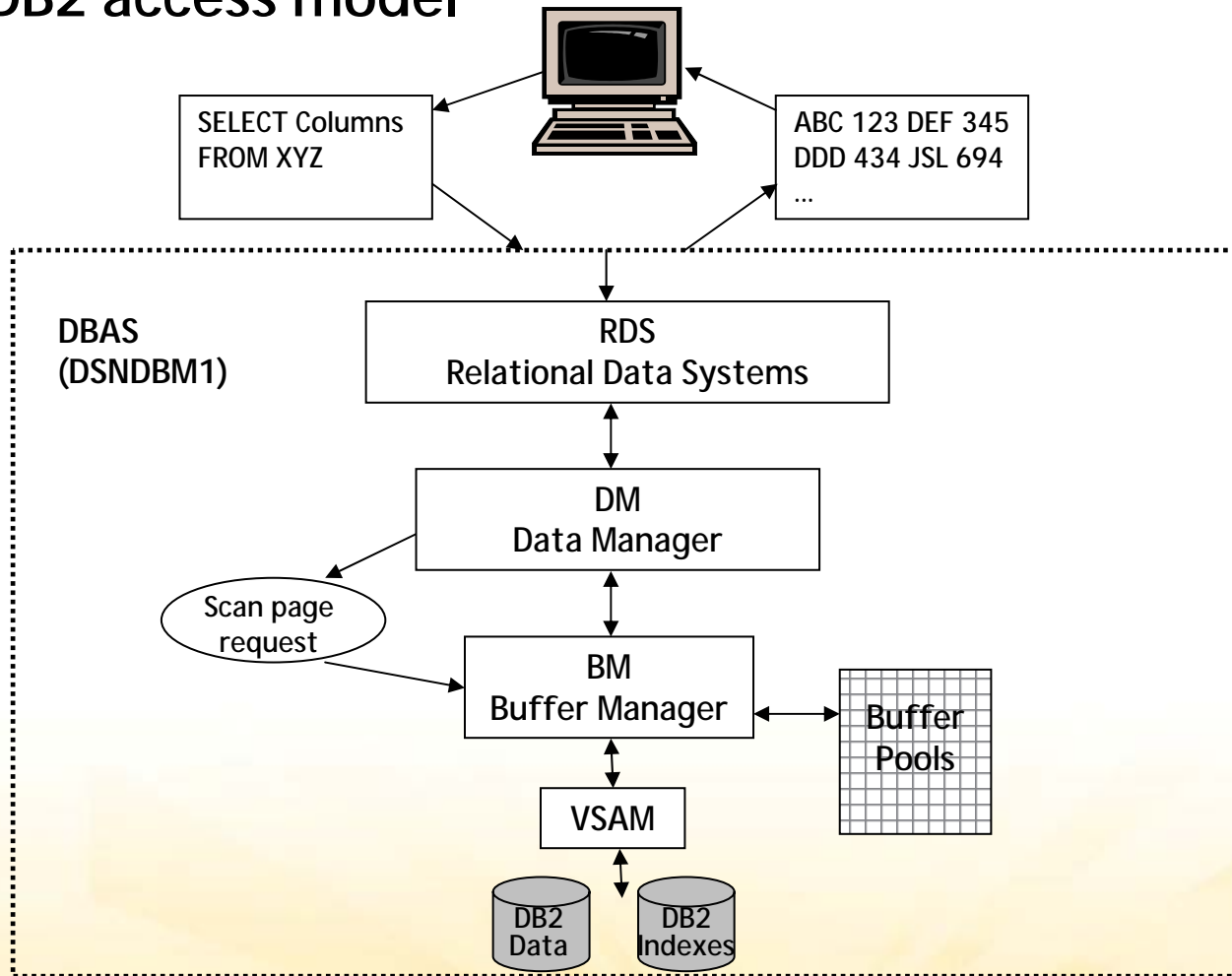
Disadvantages:

- Based on IFCIDs, large amounts of data can result
- Evaluation of the information is not simple

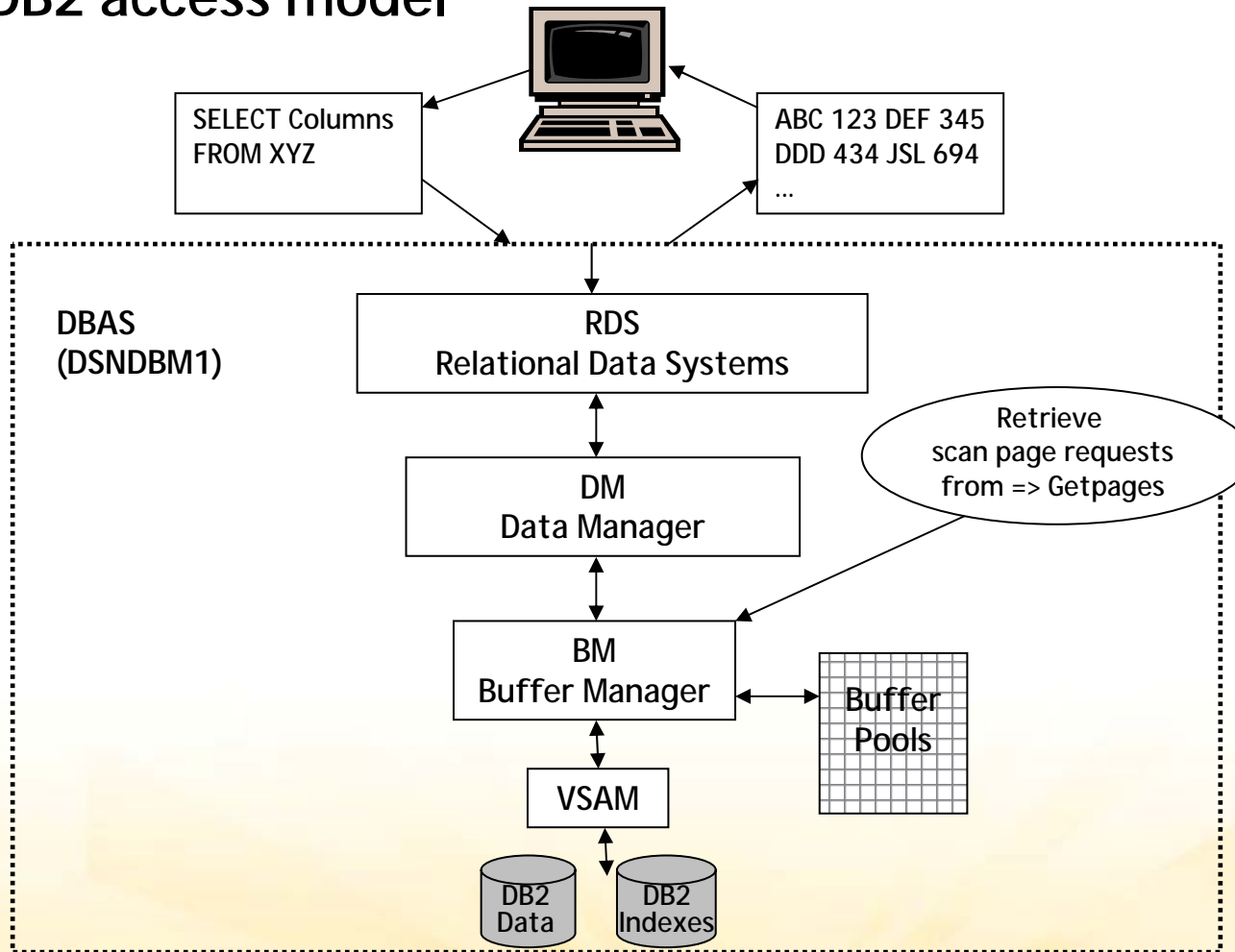
DB2 access model



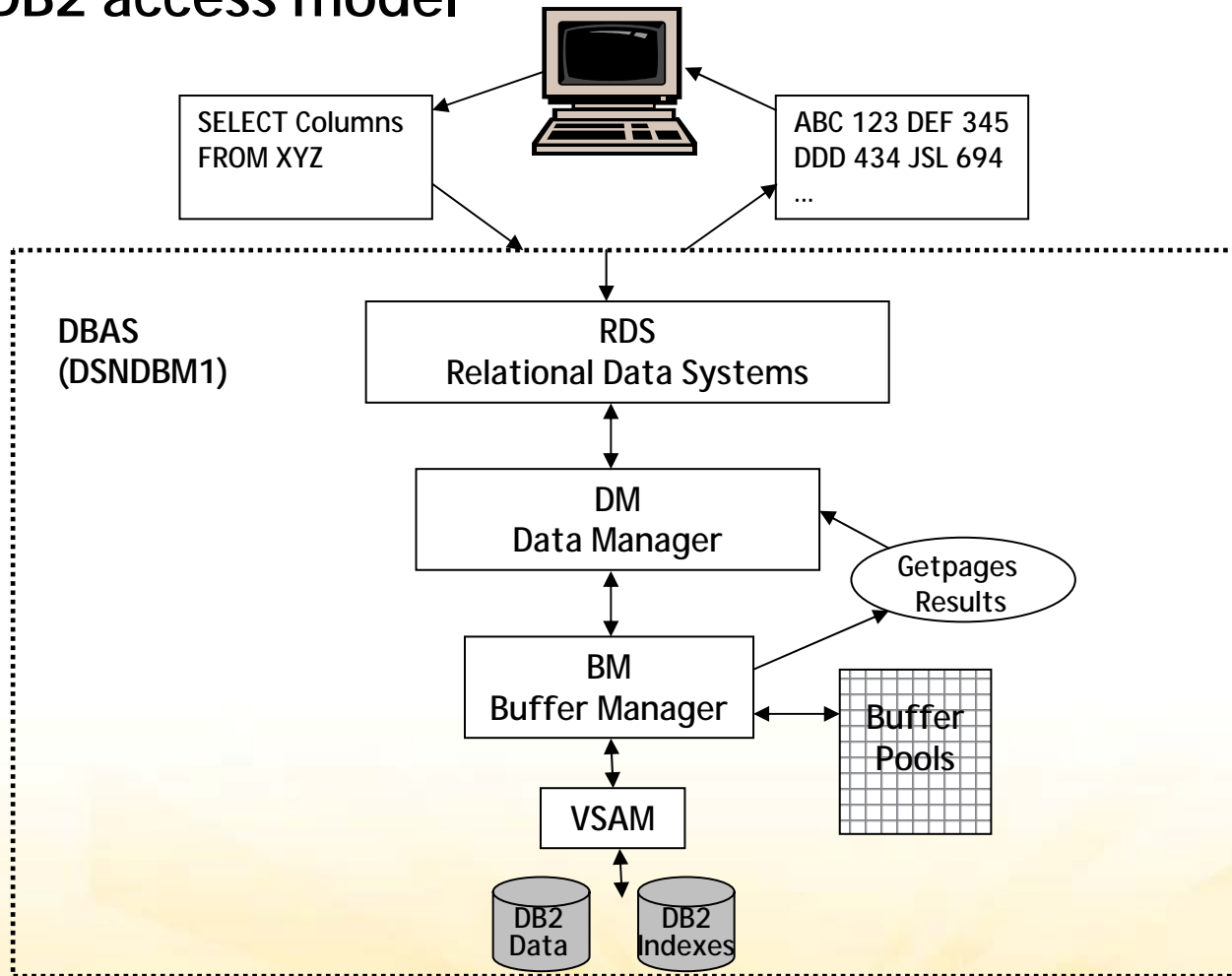
DB2 access model



DB2 access model



DB2 access model



DB2 performance trace

Starting a performance trace:

- GTF Trace Facility must be started

Name of the GTF procedure must be obtained from the system programmer and correct parameters must be in place

```
S GTF .xxxx
```

```
P xxxx
```

- Start the performance trace with:

```
-START TRACE (PERFM) CLASS(31) PLAN(*_____)  
AUTHID(*_____) IFCID(ifcids) DEST(GTF)  
TDATA(COR,CPU)
```

DB2 performance trace

Relevant IFCIDs:

- 15, 16, 17, 18 Scan Pages
- 58, 59, 60, 61, 62, 63, 64, 65, 66 SQL
- 198 Getpages
- 325 Trigger
- V8: 350 (instead of 63) Statement Text (2 MB)

Detailed information in the Appendix of DB2 Administration Guide

DB2 performance trace output

```

000000 001A0000 0001FFFF 94B6A6E9 BD6636FA 5C021000 00010000 0000
      A          B          C          D          E      F
000000 011C0000 FF00A6E9 C33E28F7 DD03EFB9 00F91400 E2E2D6D7 D4E2E3D9 01000100
      G          H          I          J      K          L          M      N          O
000020 E2E2D6D7 00000001 000000A0 00980001 00000038 00680001 0060005E 4DE2E3C1
000040 D9E340E3 D9C1C3C5 404DE2E3 C1E3405D C3D3C1E2 E2404D5C 405DD9D4 C9C4404D
000060 5C405DC4 C5E2E340 4DC7E3C6 405DD7D3 C1D5404D 5C405DC1 E4E3C8C9 C4404D5C
000080 405DC9C6 C3C9C440 4D5C405D C2E4C6E2 C9E9C540 4D5C405D FFFFFFFF 00040101
      P          Q      R      S
0000A0 004C0110 000402xx 00B3ADB8 E2E2D6D7 A6E9C33E 28EF4403 00000006 00000001
      T
0000C0 00000001 E2C1D5E3 C16DE3C5 D9C5E2C1 6DD3C1C2 C4C2F2D5 C5E34040 D3E4D5C4
0000E0 F0404040 A6E9C33E 271F0001 004C0200 E2E8E2D6 D7D94040 F0F2F34B C7C3E2C3
000100 D5F6F0F2 E2E2D6D7 40404040 40404040 40404040 E2E8E2D6 D7D94040
      U
000000 00440000 FF00A6E9 C33E2901 1303EFB9 00F91400 E2E2D6D7 D4E2E3D9 00280200
000020 E2E2D6D7 00000001 00000000 00000000 00000000 00000000 00000000 00000000
000040 00000000 V
      W
000000 011C0000 FF00A6E9 C33E2948 E203EFB9 00F91400 E2E2D6D7 D4E2E3D9 01000100
000020 E2E2D6D7 00000002 000006D8 004C0001 00000090 001C0004 00000100 001C000E
000040 00000288 0018000E 00000590 00400001 000005D0 00740001 00000480 00440001
000060 000003D8 00800001 00000458 00280001 00000644 00480001 000004E4 00AC0001
000080 0000068C 004C0001 000004C4 00200001 D4E2E3D9 00000001 762236F2 00000000
0000A0 59F48900 001E001E 00F91400 C4C2D4F1 00000001 1A789573 00000000 95826100
0000C0 001F001F 00F90E00 C4C9E2E3 00000000 3413C60E 00000000 1C4D0A00 00220022
0000E0 00F90480 C9D9D3D4 00000000 0629E2BC 00000000 145CE000 001D001D 00F91600
000100 E2D4C640 00000046 00000046 00000000 00000000 00000000 00000000
    
```

DB2 performance trace description

IFCID	Mapped by Macro
0001	DSNDQWST, subtype=0
0002	DSNDQWST, subtype=1
0003	DSNDQWAS
0004--0057	DSNDQW00
0058--0139 (except 0106)	DSNDQW01
0106	DSNDQWPZ
0140--196, 198, 199	DSNDQW02
0201--0249 (except 0202, 230 and 239)	DSNDQW03
0202	DSNDQWS2, subtype=2
0230	DSNDQWST, subtype=3
0239	DSNDQWAS and DSNDQWA1
0250--0330	DSNDQW04

DB2 performance trace description (cont.)

Additional information can be found in
SDSNSAMP(DSNWMSG) (V7)

IFCIDs

65	Open	15-18	Scanpages
66	Close	198	Getpages
59	Fetch		
58	End SQL		all except 63/350/15-18/198
61	Delete/Ins/Upd		containing information about
60	Single Select		program/package/statement no.
63	Text (s. 350)		
64	Prepare		

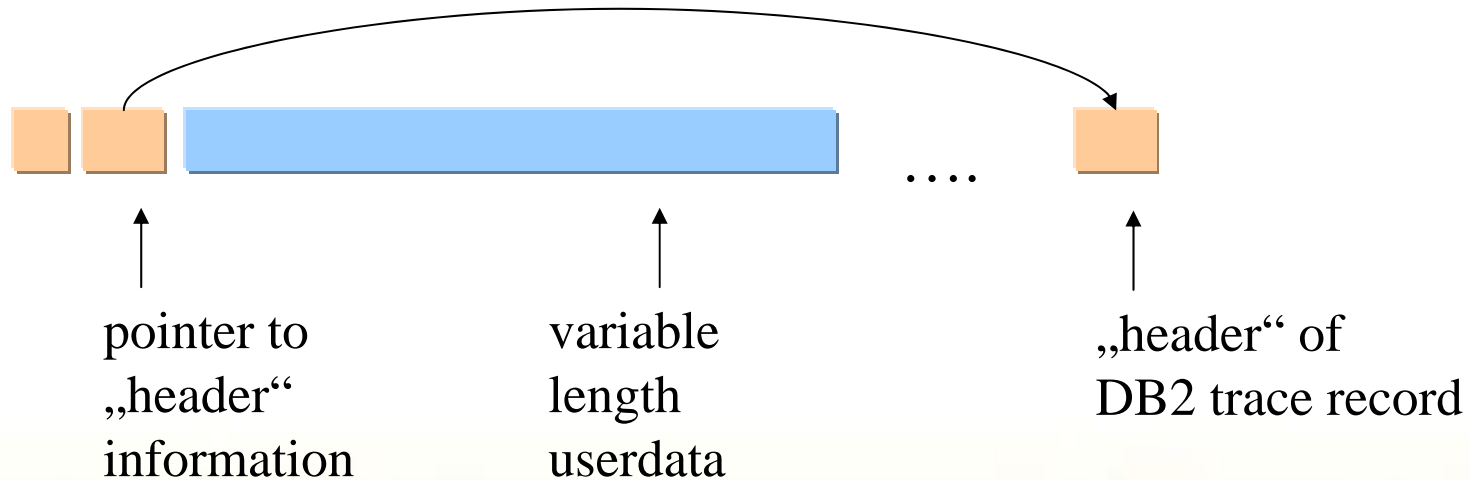
DB2 performance trace analysis

Procedure:

- If needed, merge traces (using IPCS with option COPYTRC)
- Separate control records from DB2 trace records (FF for userdata in GTF header)
- Join DB2 trace record parts together
- In DB2 trace record: standard header (DSNDQWHS) with IFCID no. / timestamp / ... at end of record
- Condense similar SQL, if possible

DB2 performance trace analysis (cont.)

Trace record analysis



DB2 performance trace analysis (cont.)

Typical IFCID sequence (static / dynamic):

Static

65	Open cursor
58	End-Of-Statement
59	Fetch
15-18	Scanpages
198	Getpages
58	EOS
59	...
...	
66	Close
58	EOS

Dynamic

64	Prepare
63	„SELECT ...“
58	EOS
65	Open cursor
...	

DB2 V8: Consider EXPLAIN STMTCACHE ALL

With DB2 V8 post GA PTF PQ88073:

Possible to gather runtime information from Dynamic Statement Cache

-> EXPLAIN STMTCACHE ALL returns information like IFCIDs 0316
and 0317 into DSN_STATEMENT_CACHE_TABLE (s. APAR desc.)

-> only for Dynamic Statement Cache

- ZPARM for using Dynamic Statement Cache switched on
- Create database/tablespace/LOB tablespace/table
- Useful information for runtime performance of Dynamic SQL

Determination of index candidates

- Measure a typical application workload (Trace / Monitor)
- Determine the top ten DB2 tables with the highest overhead
- Determine the SQL accessing these tables
- Build an SQL/table matrix
- Determine the top five SQL out of this matrix
- Build a column / predicate matrix
- Decide on index candidates
- Test the indexes (e.g., EXPLAIN, introduce into test environment)

Determination of index candidates

The tables are determined with the maximum workload from the trace or monitored data.

These tables are combined in a matrix with the SQL having the maximum workload:

	SQL-1	SQL-2	SQL-3	SQL-4	SQL-5
Table-1	10.000	500	200		200
Table-2		15.000	500	12.000	
Table-3	800	10		13.000	500
Table-4		30.000		20.000	25.000

Determination of index candidates

After selecting „sense making“ tables and SQL combinations, a table column /SQL predicate matrix is created:

	SQL2_ Pred1	SQL2_ Pred2	SQL4_ Pred1	SQL5_ Pred1	SQL5_ Pred2
Table4_Col1	X		X	X	
Table4_Col2			X		X
Table4_Col3	X	X	X	X	X
Table4_Col4			X		

Choosing predicates

Using Administration Guide V7 5.8.3.2: Which predicates are indexable?

Predicate Type	Index-able?	Stage 1?	Notes
COL = value	Y	Y	13
COL = noncol expr	Y	Y	9,11,12
COL IS NULL	Y	Y	
COL op value	Y	Y	
COL op noncol expr	Y	Y	9, 11
COL BETWEEN value1 AND value2	Y	Y	
COL BETWEEN noncol expr1 AND noncol expr2	Y	Y	9, 11
value BETWEEN COL1 AND COL2	N	N	
COL BETWEEN COL1 AND COL2	N	N	10

V8 news for indexes

Some important changes for DB2 for z/OS V8 and indexes:

- VARCHAR true index-only access
- More predicates indexable when type mismatch
- col BETWEEN expr1 AND expr2 indexable
- col IS NOT NULL indexable
- Backward index scan possible (see redundant indexes!)

Choosing predicates

Depending on the DB2 version and the Optimizer in use (PTF dependent), adhere to the following rules:

- No further index usage on Column OP or VARCHAR
- Do not use columns with low cardinality
- Do not consider columns with extreme length
- Suitability of VARCHAR columns is limited (before V8)
- If possible, do not use columns in indexes that are regularly changed.
- Do consider the possibility of index-only access (e.g., to complement existing indexes)

Index contra indicators

Contra indicators for index determination:

- Many Inserts / Updates / Deletes for table in question
- Row order of the columns is very different for different SQL statements
- The SQL access already strongly uses existing indexes
- Columns unsuitable for usage in index (e.g., length, cardinality)

Finding out unused indexes

- Use EXPLAIN(YES) for ALL static SQL production packages
- Check PLAN_TABLE entries, cross-check with catalog
- Use Dynamic Statement Cache for dynamic SQL (capture several snapshots)
- EXPLAIN STMTCACHE ALL, then EXPLAIN of statements
- Check PLAN_TABLE entries, cross-check with catalog
- Recommend dropping index to a colleague, take vacation ;-)

Index verification

- RUNSTATS for current statistics
- Check index usage for selected SQL statements using EXPLAIN tools
- Test the performance of the access in the preproduction environment
- Monitor after introduction into the production environment
- Document the savings (no heroes without witnesses!)

Catalog statistics

Why correct statistics are so important

- the optimizer is god
- help god make the right decisions
- incorrect statistics may cause:
 - less efficient join sequence
 - less efficient method of accessing individual tables (sync I/O instead of prefetch)
 - wrong or no index may be used
- Terry Purcell, IDUG Prague 2004:
„50 % of all bad access paths are caused by bad statistics“

Providing correct statistics

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

Providing correct statistics (cont.)

Consider gathering frequency / distribution for columns

- RUNSTATS with KEYCARD / FREQVAL on skewed columns
- Can enforce index usage
- Take care of SYSCOLDIST maintenance
- New options also for non-indexed columns in V8
- Further information in DB2 Administration Guide

Summary of recommended procedures

- Workload-based
- Find the top candidates
- First optimize the tip of the iceberg
- Create matrix to determine the index candidates
- Iterate procedures
- Document and present the savings results (to become a hero!)
- Important: Don't forget to use the DBA parts of your brain!

Summary of top indicators

- High workload on the table space
- Workload caused by few SQL statements
- Predicates are indexable, common columns are in similar order
- Few updates / inserts / deletes on the table space
- High column cardinality
- Till now, few indexes on the table space

In closing:

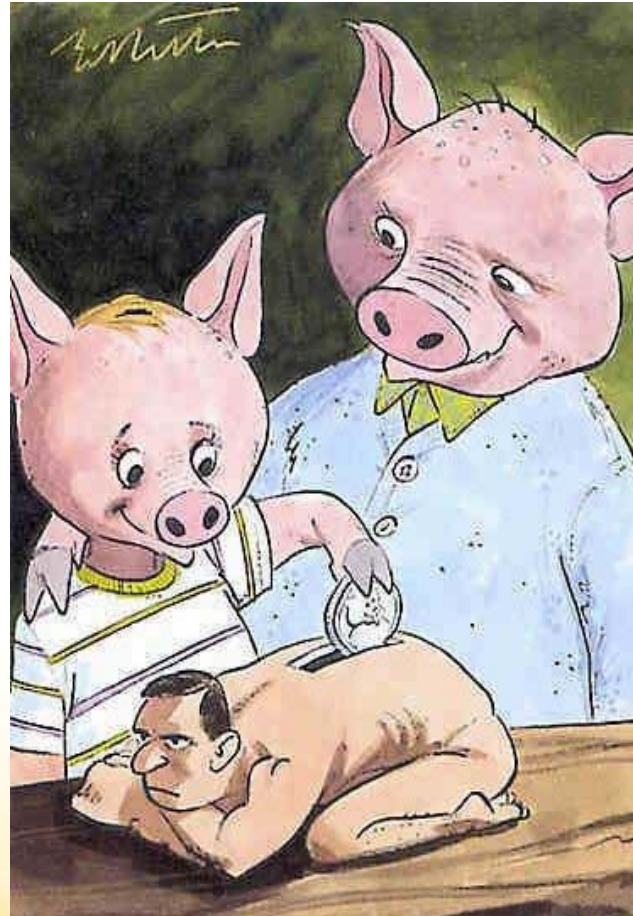
- Don't forget: Performance optimization is an ongoing task
- ... with which we always have something to do ...
- ... and when we are almost finished, IBM changes something with the Optimizer => then we start over
- **But:**
never, never think that one is an index expert!
Stay modest!

About the Presenter

Dirk Johann

- 16 years of hands-on experience in the MVS OS/390 z/OS world,
- having strong skills in performance optimization among others
- is a senior consultant for **SOFTWARE ENGINEERING**

Questions?



Do's and Don'ts for Finding Performance Indexes
Session: A05

Dirk Johann
SOFTWARE ENGINEERING
D.Johann@seg.de

A downloadable version of this presentation is available on our partner's website:

www.neonesoft.com