IDUG

SQL and Configuration Tips and Tricks Learned from the SAP/Db2 Project

Joern Klauke, Edgar Maniago, Phil King

IBM and SAP

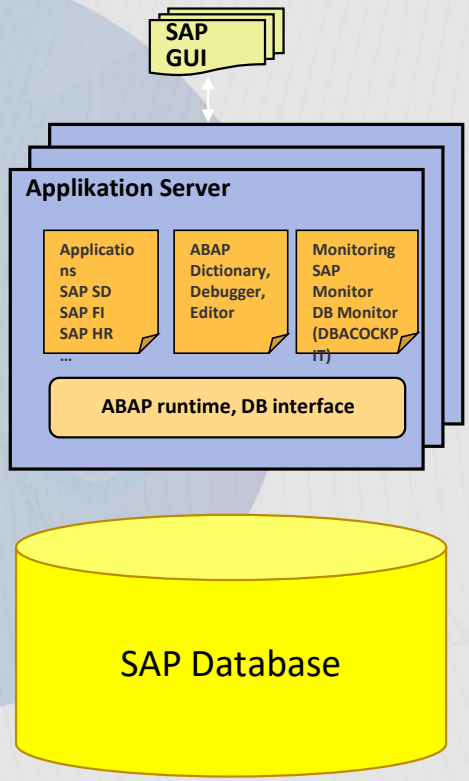
@IDUGDb2
#IDUG_NA24

Session Code: SQL3 | Platform: Working with SQL

Agenda

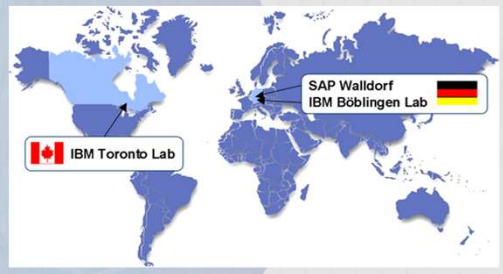
- Overview SAP ABAP stack environment
- Isolation Levels
- Space
- LOBs
- Performance
- Lock Wait Monitoring

Overview SAP ABAP stack environment



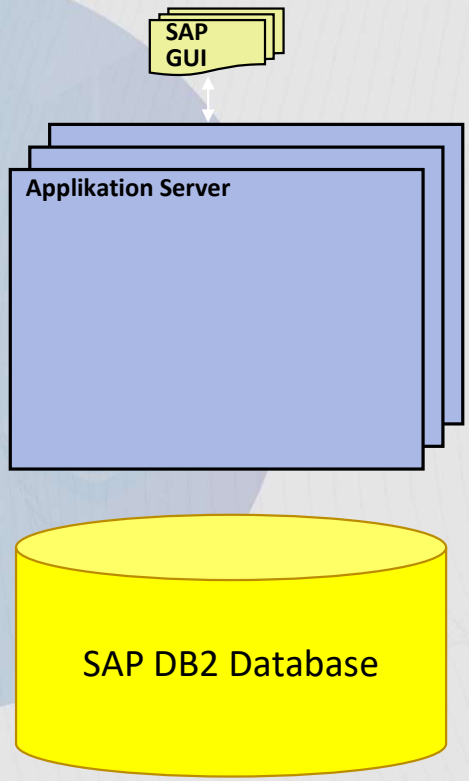
- An SAP ABAP system has one primary database and multiple application servers
- The ABAP system has its own user/pwd management for business users.
- Each application server has multiple work processes that connect to the database with a technical DB user
- An SAP Business Suite system contains more than 100.000 database tables

History SAP ABAP stack environment on Db2 LUW



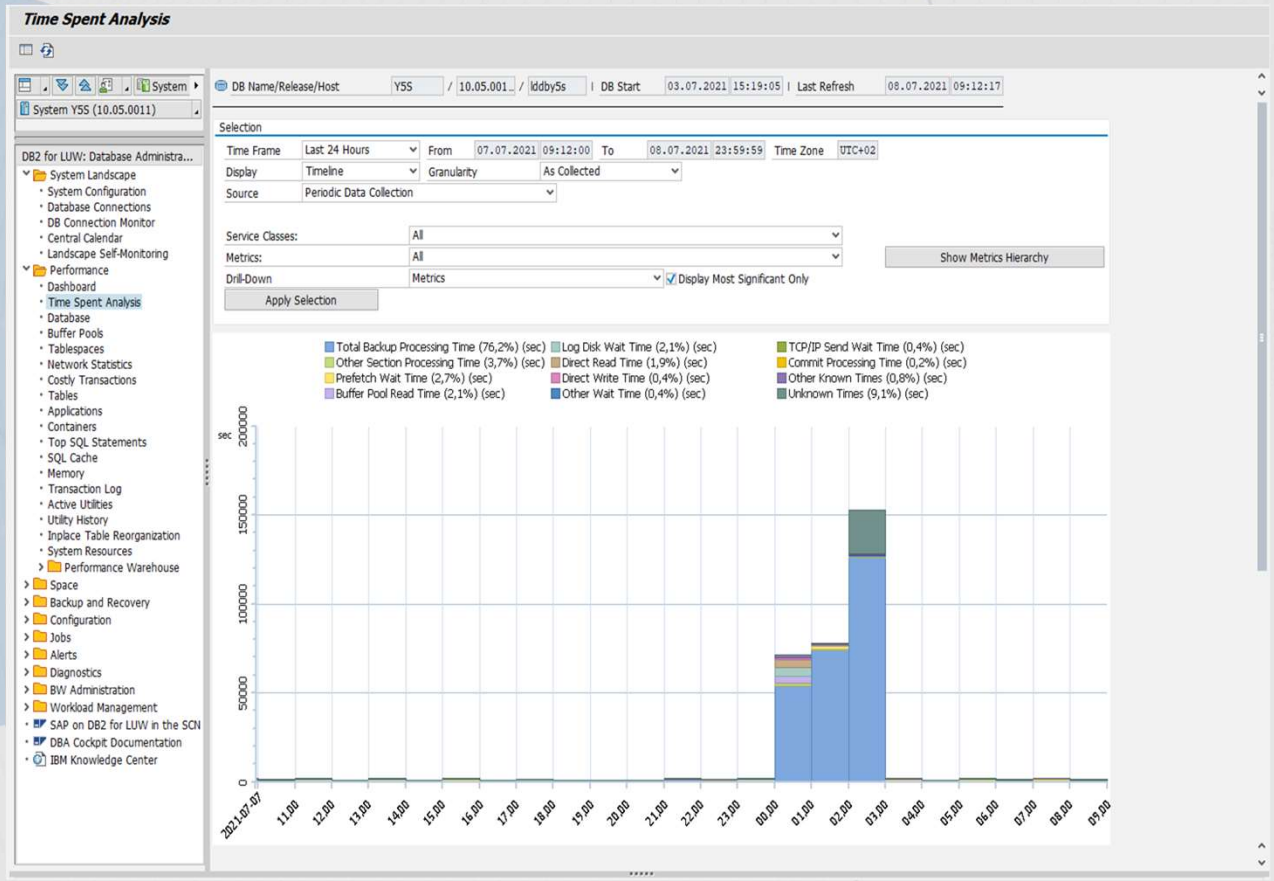
- SAP on Db2 LUW is a joint SAP+IBM project since 1993
  - Joint SAP+IBM development team in Rot (Germany)
  - Joint SAP IBM Integration Center in Toronto
  - Regular meetings and calls with DB2 Development Toronto
- Lots of SQL lessons and other Db2 tricks learned in more than 30 years of project history

Overview SAP ABAP stack environment on Db2 LUW



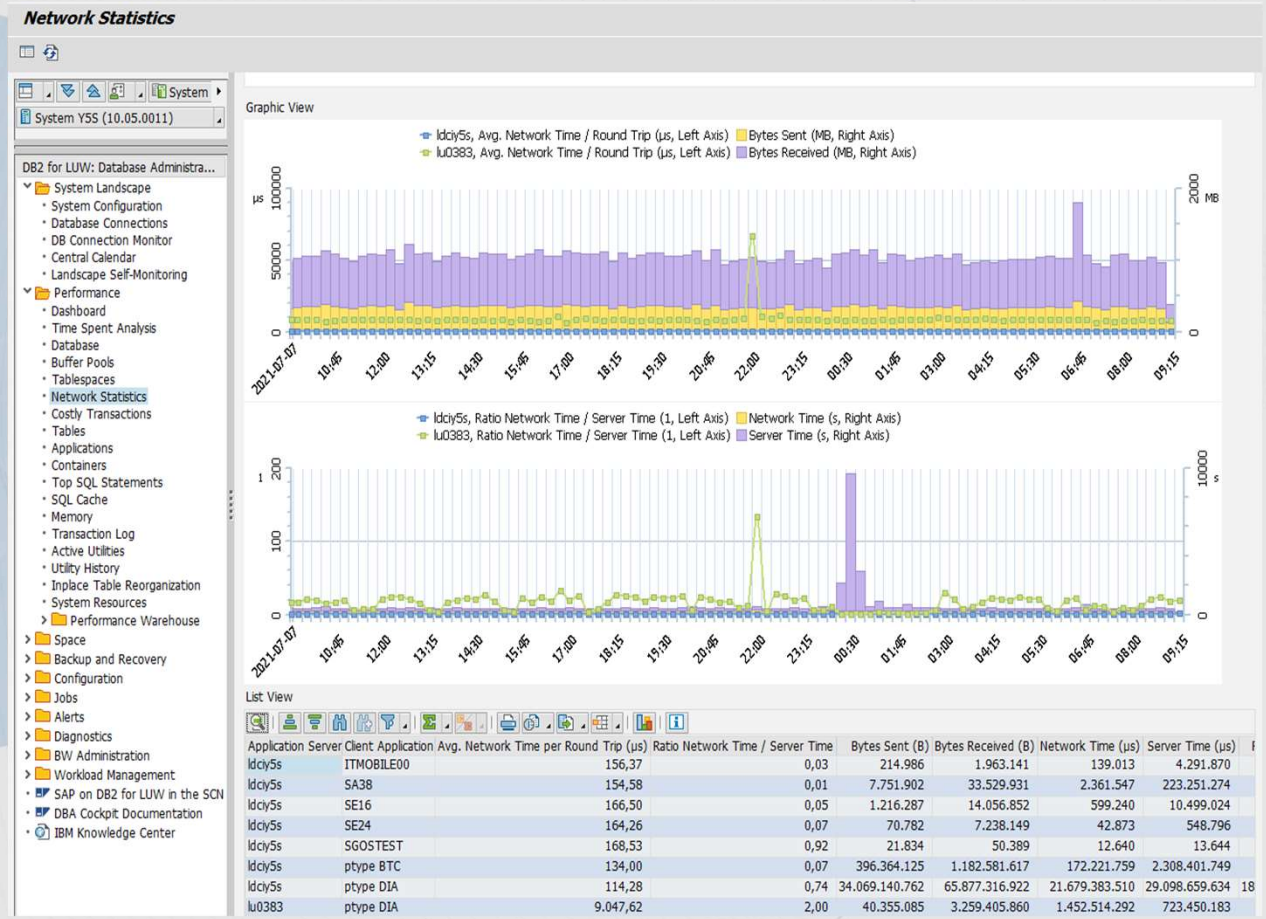
- Only row based tables on ERP systems (OLTP workload)
- On SAP BW systems tables involved in OLAP queries can be BLU
- DPF is supported for SAP BW systems
- PureScale is supported for SAP ERP systems
- Top rated features according to SAP on Db2 LUW customers:
  - Compression (BLU and non-BLU)
  - Automation (Reorg, Runstats, STMM, Auto Storage)
  - Performance
  - Easy online table maintenance using ADMIN_MOVE_TABLE

Overview SAP ABAP stack environment on Db2 LUW



- ABAP systems contain an integrated DB monitoring transaction (DBACOCKPIT)
- Data is periodically collected using Db2s MON_GET* table functions
- Other integrated features in DBACOCKPIT include EXPLAIN, Index Advisor, …

Overview SAP ABAP stack environment on Db2 LUW



Application Server	Client Application	Avg. Network Time per Round Trip (µs)	Ratio Network Time / Server Time	Bytes Sent (B)	Bytes Received (B)	Network Time (µs)	Server Time (µs)
ldciy5s	ITMOBILE00	156,37	0,03	214.986	1.963.141	139.013	4.291.870
ldciy5s	SA38	154,58	0,01	7.751.902	33.529.931	2.361.547	223.251.274
ldciy5s	SE16	166,50	0,05	1.216.287	14.056.852	599.240	10.499.024
ldciy5s	SE24	164,26	0,07	70.782	7.238.149	42.873	548.796
ldciy5s	SGOSTEST	168,53	0,92	21.834	50.389	12.640	13.644
ldciy5s	ptype BTC	134,00	0,07	396.364.125	1.182.581.617	172.221.759	2.308.401.749
ldciy5s	ptype DIA	114,28	0,74	34.069.140.762	65.877.316.922	21.679.383.510	29.098.659.634
lu0383	ptype DIA	9.047,62	2,00	40.355.085	3.259.405.860	1.452.514.292	723.450.183

Analysis of network related performance problems

- Network statistics data is collected by Db2 CLI
- Persisted network data can be visualized in DBACOCKPIT
- Often slow average database response times are wrongly attributed to the Db2 engine

```
db6x161001:db2fmh 59> db2set -all | grep DB2_WORKLOAD
[i] DB2_DYNAMIC_SSL_LABEL=ON [DB2_WORKLOAD]
[i] DB2_SECTION_SCRATCH_BUFFER_SIZE=512K [DB2_WORKLOAD]
[i] DB2_CDE_INX_EXCL_BEHAVIOUR=OFF [DB2_WORKLOAD]
[i] DB2_REDUCE_FLUSHING_DURING_BACKUP=ON [DB2_WORKLOAD]
[i] DB2_SYNC_RELEASE_LOCK_ATTRIBUTES=YES [DB2_WORKLOAD]
[i] DB2_ONLINERECOVERY_WITH_UR_ACCESS=FALSE [DB2_WORKLOAD]
[i] DB2_AVOID_LOCK_ESCALATION=TRUE [DB2_WORKLOAD]
[i] DB2_PARALLEL_ACS=YES [DB2_WORKLOAD]
[i] DB2_TRANSCHEMA_EXCLUDE_STATS=TRUE [DB2_WORKLOAD]
[i] DB2_USE_FAST_LOG_PREALLOCATION=TRUE [DB2_WORKLOAD]
[i] DB2_CDE_STMTCACHING=YES [DB2_WORKLOAD]
[i] DB2_INDEX_PCTFREE_DEFAULT=0 [DB2_WORKLOAD]
[i] DB2_SKIP_VIEWRECREATE_SAP=TRUE [DB2_WORKLOAD]
[i] DB2_RESTORE_GRANT_ADMIN_AUTHORITIES=YES [DB2_WORKLOAD]
[i] DB2_BLOCKING_WITHHOLD_LOBLOCATOR=NO [DB2_WORKLOAD]
[i] DB2_AGENT_CACHING_FMP=OFF [DB2_WORKLOAD]
[i] DB2_TRUST_MDC_BLOCK_FULL_HINT=YES [DB2_WORKLOAD]
[i] DB2_CREATE_INDEX_COLLECT_STATS=YES [DB2_WORKLOAD]
[i] DB2_ATS_ENABLE=YES [DB2_WORKLOAD]
[i] DB2_RESTRICT_DDF=YES [DB2_WORKLOAD]
[i] DB2_DUMP_SECTION_ENV=YES [DB2_WORKLOAD]
[i] DB2_OPTSTATS_LOG=ON,NUM=8,SIZE=256 [DB2_WORKLOAD]
[i] DB2_OPT_MAX_TEMP_SIZE=10240 [DB2_WORKLOAD]
[i] DB2_WORKLOAD=SAP
[i] DB2_TRUNCATE_REUSESTORAGE=IMPORT,LOAD,TRUNCATE [DB2_WORKLOAD]
[i] DB2_MDC_ROLLOUT=DEFER [DB2_WORKLOAD]
[i] DB2_ATM_CMD_LINE_ARGS=-include-manual-tables -low-priority-update-systables [DB2_WORKLOAD]
[i] DB2_SKIPINSERTED=YES [DB2_WORKLOAD]
[i] DB2_VIEW_REOPT_VALUES=YES [DB2_WORKLOAD]
[i] DB2_OBJECT_TABLE_ENTRIES=65532 [DB2_WORKLOAD]
[i] DB2_IMPLICIT_UNICODE=YES [DB2_WORKLOAD]
[i] DB2_BCKP_PAGE_VERIFICATION=TRUE [DB2_WORKLOAD]
[i] DB2_BCKP_INCLUDE_LOGS_WARNING=YES [DB2_WORKLOAD]
[i] DB2_RUNTIME_DEBUG_FLAGS=TOLERANT_FLOAT,DISABLE_BLANK_TOLERANCE,SECTION_LEVEL_LOB [DB2_WORKLOAD]
[i] DB2STMM=APPLY_HEURISTICS:YES,GLOBAL_BENEFIT_SEGMENT_UNIQUE:YES [DB2_WORKLOAD]
[i] DB2_INLIST_TO_NLJN=YES [DB2_WORKLOAD]
[i] DB2_MINIMIZE_LISTPREFETCH=YES [DB2_WORKLOAD]
[i] DB2_REDUCED_OPTIMIZATION=4,INDEX,JOIN,NO_TQ_FACT ON,NO_HSJN_BUILD_FACT ON,STARJN_CARD_SKEW ON,NO_SORT_MGJOIN,REDUCE_LO
CKING,CART OFF,CAP OFF,BLDCNFTST OFF,TQCNFTST OFF [DB2_WORKLOAD]
[i] DB2NOTIFYVERBOSE=YES [DB2_WORKLOAD]
[i] DB2_INTERESTING_KEYS=YES [DB2_WORKLOAD]
[i] DB2_EVALUNCOMMITTED=YES [DB2_WORKLOAD]
[i] DB2_EXTENDED_OPTIMIZATION=GY_DELAY_EXPAND_ROW_SJOK 1000,NO_NLJN_SPLIT_BUFFER,NO_HVCHECK_ALL,NOBLDCRDTST ON,COL_PARALLE
L_IUD_THR 50000,CDE_RTABLE_INS_THR 190000,SAPMT_OUTER ON [DB2_WORKLOAD]
[i] DB2_SELECTIVITY=ALL [DB2_WORKLOAD]
[i] DB2_ANTIJOIN=EXTEND [DB2_WORKLOAD]
[i] DB2COMPOPT=VOLATILETSF,WORKLOADSAP,BLU_SAP_MEMTBL,BREAK_VIEWCSE,REPL_UNLIMITED,SAPMTPD,OJSJ_REORDER,LOTO1 [DB2_WORKLOA
D]
```

Aggregate Db2 Registry Variable DB2_WORKLOAD=SAP

- All SAP on Db2 LUW systems run with aggregate registry variable DB2_WORKLOAD=SAP
- DB2_WORKLOAD=SAP covers lots of settings that benefit SAP on Db2 LUW
- Settings change with new Db2 versions and fixpacks
- Settings influence: Optimizer, Isolation, Backup, Log Management and others

db2sap Library

```
db6xen030:db2fmh 55> db2 "CALL SYSINSTALLOBJECTS('DB2SAP','C','','')"

  Return Status = 0
db6xen030:db2fmh 56> db2 "select count(*) from syscat.functions where funcschema ='SAPTOOLS' "

1
-----------
         45

  1 record(s) selected.
```

The db2sap library is shipped with the Db2 software.

Invoking the library creates a number of objects in the SAPTOOLS schema that are used by the SAP ABAP stack.

SAP default isolation level

- For SQL queries standard SAP applications require a default DB isolation level where “readers never block writers”
- Those applications should use logical SAP enqueue locks to prevent themselves from processing dirty data.
- Since Db2 V9.7 the SAP system can choose one of the following default isolation levels (SAP note 1514016)
  - Uncommitted read (UR)
  - Cursor Stability (CS) with CUR_COMMIT semantics

ABAP “SELECT SINGLE … FOR UPDATE” statements

An ABAP “SELECT SINGLE … FOR UPDATE” statement makes sure that the caller can execute an UPDATE in the same UOW.

Translating those statement to DB2 “SELECT … FOR UPDATE” statements would be too weak since those statements set only (U)pdate locks on row level and two (U)pdate locks are compatible.

As an alternative we use the following:

SELECT … WITH RS USE AND KEEP EXCLUSIVE LOCKS

X locks ensure that other statements requiring the same locks wait.

ABAP “MODIFY” statements

An ABAP “MODIFY” statement wants to
UPDATE a row it exists and
INSERT if it does not exist.

We use DB2 MERGE statements for this purpose.

```
MERGE INTO "DYNPLOAD" AS t
  USING TABLE( VALUES( ?, ?, ?, ? ) ) AS p( "DATA", "PROGNAME", "DYNPNUMBER", "TYPE" )
  ON t."PROGNAME"=p."PROGNAME" AND t."DYNPNUMBER"=p."DYNPNUMBER" AND t."TYPE"=p."TYPE"
WHEN MATCHED THEN
  UPDATE SET ( t."DATA" ) = ( p."DATA" )
WHEN NOT MATCHED THEN
  INSERT ( "DATA", "PROGNAME", "DYNPNUMBER", "TYPE" )
  VALUES ( p."DATA", p."PROGNAME", p."DYNPNUMBER", p."TYPE" )
WITH CS WAIT FOR OUTCOME
```

Using “WITH CS WAIT FOR OUTCOME” reduces the probability that MERGE returns a SQL0803N .

ABAP “INSERT … ACCEPTING DUPLICATE KEYS” statements

An ABAP “INSERT … ACCEPTING DUPLICATE KEYS” statement INSERTs rows that do not exist.
SQL0803 errors due to existing rows are ignored. We use the IGNORE DUPLICATE clause for this purpose.

INSERT … IGNORE DUPLICATES

Currently the IGNORE DUPLICATES clause is only available under DB2_WORKLOAD=SAP.

Such INSERT statements can also be non-atomic INSERT … SELECT statements.

Space – Virtual Tables

An SAP ERP system can contain more than 100.000 tables. Around 70% of those tables are initially empty and may never be used. Each empty table consumes space in data, index and long table spaces.

1 EMP extent (EMP = Extent Map Pages)
1 data extent
2 extents for the index object
1 page per index
2 extents for the LONG object if exists
4 extents for the LOB Object if exists

5 - 11 Extents for a single empty table

Space – Virtual Tables

The waste of space caused by empty or small tables increases with larger extent sizes.

=> extentsize 2 is used as SAP default (pagesize 16K)

Empty tables…

- also consume memory in other areas like DBHEAP
- contribute to the number of objects in a table space
  (DB2 limit at about 51000 objects per table space)
- cause increased workload at DBA operations
  (Automatic Runstats, Automatic Reorg, monitoring jobs , ...)

Space – Virtual Tables

Tables are created on demand by the SAP database interface when the first write occurs.

```
CREATE TABLE “POSTAB"
    ( "NUM" SAPDB6FIXCHAR(3) DEFAULT '000' NOT NULL ,
     "POS" INTEGER DEFAULT 0 )
     IN LZY#DDICD INDEX IN LZY#DDICI LONG IN LZY#DDICD

CREATE UNIQUE INDEX "POSTAB~0" ON "POSTAB“ ( "NUM“ )
    ALLOW REVERSE SCANS

ALTER TABLE "POSTAB“  ADD CONSTRAINT “POSTAB~0" PRIMARY KEY
    ( "NUM")

CREATE INDEX "POSTAB~A" ON "POSTAB“ ( “POS“ )  ALLOW REVERSE
    SCANS
```

Space – Virtual Tables

Table exist as virtual tables (views without dependent objects) as long as no writes occur.

```
CREATE VIEW "POSTAB" AS SELECT * FROM
  ( VALUES( CAST ( NULL AS VARCHAR(9) ), CAST ( NULL AS INTEGER) )
   AS "POSTAB" ( "NUM", "POS" )  WHERE 1 = 2
   --DDL_TABLE CREATE TABLE "POSTAB"
      ( "NUM" SAPDB6FIXCHAR(3) DEFAULT '000' NOT NULL ,
        "POS" INTEGER DEFAULT 0 )
         IN LZY#DDICD INDEX IN LZY#DDICI LONG IN LZY#DDIC
   --DDL_INDEX CREATE UNIQUE INDEX "POSTAB~0" ON
       "POSTAB" ( "NUM" ) ALLOW REVERSE SCAN
   --DDL_PRKEY ALTER TABLE "POSTAB" ADD CONSTRAINT "POSTAB~0"
      PRIMARY KEY ( "NUM", "POS" )
   --DDL_INDEX CREATE INDEX "POSTAB~A" ON "POSTAB" ( "POS" )
      ALLOW REVERSE SCANS
```

Reading applications do not notice the difference.

```
> db2 " select * from postab "

NUM       POS
--------- -----------

  0 record(s) selected.

> db2 " describe table postab "


Column name  schema    type          Len Scale
------------ --------- ---------- --- --
NUM          SYSIBM    VARCHAR     9    0
POS          SYSIBM    INTEGER     4    0

  2 record(s) selected.
```

Space – Tablespace Pools

To circumvent the maximum number of objects limit for a single tablespace and to optimize database backup times tables are distributed over a pool of tablespaces at create time

Tablespaces (DATA, INDEX, LONG/LOB):

```
SID#DATA@01D   SID#DATA@01D  SID#DATA@01D
SID#DATA@02D   SID#DATA@02D  SID#DATA@02D
SID#DATA@03D   SID#DATA@03D  SID#DATA@03D
SID#DATA@04D   SID#DATA@04D  SID#DATA@04D
SID#DATA@05D   SID#DATA@05D  SID#DATA@05D
SID#DATA@06D   SID#DATA@06D  SID#DATA@06D
…
```

Space – DB2_OBJECT_TABLE_ENTRIES

Operations like TRUNCATE TABLE … REUSE STORAGE, REORG may have to wait for an online backup to backup all object table pages before they can proceed.

db2set DB2_OBJECT_TABLE_ENTRIES=65532

ensures that all object table pages are created at the beginning of each tablespace. This minimizes such lock wait times.

Space – SAP RUNSTATs and REORG strategy

```
Automatic maintenance                    (AUTO_MAINT) = ON
  Automatic database backup           (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance        (AUTO_TBL_MAINT) = ON
    Automatic runstats                (AUTO_RUNSTATS) = ON
      Real-time statistics          (AUTO_STMT_STATS) = ON
      Statistical views            (AUTO_STATS_VIEWS) = OFF
      Automatic sampling              (AUTO_SAMPLING) = ON
    Automatic reorganization             (AUTO_REORG) = ON
```

- Automatic runstats and real time stats are recommended
- Automatic reorg is recommended for index cleanup and extent reclaim
- ADMIN_MOVE_TABLE is recommended as online method for other REORG like purposes

Space – Insert Time Clustered tables (ITC)

The SAP system contains tables where older data is periodically archived and deleted.

Customers that what to easily reclaim the space freed up by archive deletes can use ITC for such tables.

Similar to MDC tables empty extents in an ITC table can be easily reclaimed without data movement.

LOB Performance – Separate LOB Tablespaces

```
CREATE TABLE "XXX"
( ... )
IN "SID#DATA@01D"
INDEX IN "SID#DATA@01I"
LONG IN "SID#DATA@01L"
COMPRESS YES ADAPTIVE;

ALTER TABLESPACE "SID#DATA@01L" FILE SYSTEM CACHING;
```

File system caching helps for tablespaces that contain LOB data only.

All other tablespaces are created without FILE SYSTEM CACHING.

LOB Performance – LOB Inlining

ABAP system databases contain many LOB columns that contain mostly small LOB values

LOB Inlining allows to store small LOB values in data pages

Inline LOB values:

- do not need additional LOB segments
- can be compressed
- are buffered in the DB2 bufferpool

LOB Performance – Prefetcher Configuration

Only small LOB values can be directly retrieved by db2agents

Larger LOB values are fetched by the Db2 prefetchers

For good LOB performance it is important to have enough prefetchers available and to have a maximum number of Prefetch queues.

NUM_IO_SERVERS = AUTOMATIC(104)

LOB space consumption

DB2 Runstats does not collect statistics about space used by LOB values.

After deleting data there is no straightforward way to find out if a LOB object is partly empty or fragmented.

The following queries can be used to estimate the space a single LOB column should need in a LOB object of a table.

This can be compared with the current LOB object size:

```
SELECT LOB_OBJECT_P_SIZE
FROM TABLE(
  ADMIN_GET_TAB_INFO( CURRENT SCHEMA, 'LOBTAB'))
```

LOB space consumption- COMPACT LOBs

Estimated Space used in the LOB object by a single COMPACT LOB column:

```
SELECT
  SUM((LENGTHB(DATA) + 1023)/1024) AS SIZE_IN_KB
FROM LOBTAB
WHERE ADMIN_IS_INLINED(LOBCOL) <> 1
```

(LOB values are stored in segments with minimum size 1K)

LOB space consumption- NON COMPACT LOBs

Estimated Space used in the LOB object by a single NON COMPACT LOB column:

```
WITH T (BYTELEN) AS (
SELECT LENGTHB(DATA) FROM LOBTAB
WHERE ADMIN_IS_INLINED(LOBCOL) <> 1 )
SELECT
SUM(POWER(2.0, CEIL(log10(DOUBLE(BYTELEN))/log10(2.0) -10.0))))
AS SIZE_IN_KB FROM T
```

(LOB values are stored in segments of size 1K times 2^N)

LOB space consumption- NON COMPACT LOBs

```
WITH T (BYTELEN) AS ( SELECT LENGTHB(DATA) FROM LOBTEST1 WHERE ADMIN_IS_INLINED(DATA) <> 1 )
SELECT
SUM( case
   when BYTELEN <= 1024      THEN 1
   when BYTELEN <= 2048      THEN 2
   when BYTELEN <= 4096      THEN 4
   when BYTELEN <= 8192      THEN 8
   …
   when BYTELEN <= 536870912  THEN 524288
   when BYTELEN <= 1073741824 THEN 1048576
   ELSE                      2097152
   END )
   AS SIZE_IN_KB
FROM T
```

Looks more clumsy but is faster …

Performance- Use of generic table functions

ABAP SQL statements can contain both DB tables and ABAP internal tables.

To execute such statements the rows from the ABAP internal tables need to be passed to the DB2 server efficiently.

Since ABAP internal tables can have any structure we use a generic table function to pass the internal table rows encoded as BLOB input parameter

```
SELECT * FROM
TABLE( SAPTOOLS.MEMORY_TABLE( CAST( ? AS BLOB( 2G )) ) )
AS "t_00"
( "C_0" VARCHAR(3) , "C_1" INTEGER , "C_2" DOUBLE , ... )
```

SAPTOOLS.MEMORY_TABLE is a fenced C procedure.

Performance- Joining DB table(s) and ABAP table

ABAP SQL

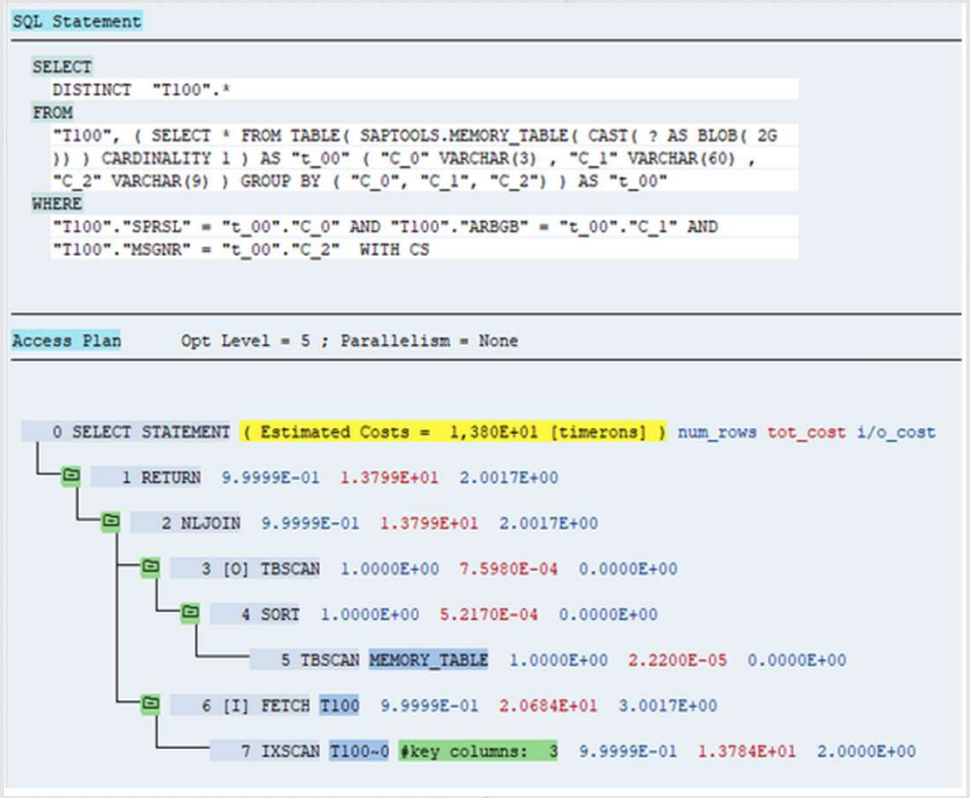
```
SELECT * INTO TABLE db6out FROM t100
BYPASSING BUFFER
FOR ALL ENTRIES IN db6
WHERE sprsl = db6-sprsl
AND arbgb = db6-arbgb
AND msgnr = db6-msgnr
```

An ABAP statement with FOR ALL ENTRIES clause joins a single ABAP internal table with one or more DB tables.

```
SELECT DISTINCT "T100".*
FROM
"T100",
TABLE( SAPTOOLS.MEMORY_TABLE( CAST( ? AS BLOB( 2G )) ) CARDINALITY 1 )
AS "t_00"
( "C_0" VARCHAR(3) , "C_1" VARCHAR(60) , "C_2" VARCHAR(9) )
WHERE
"T100"."SPRSL" = "t_00"."C_0" AND
"T100"."ARBGB" = "t_00"."C_1" AND
"T100"."MSGNR" = "t_00"."C_2"
WITH UR
```

- This is translated to a JOIN of the DB table with the MEMORY_TABLE table function
- „CARDINALITY 1“ clause is used to enforce a save NLJOIN access path.

Performance- Joining DB table and generic table function



- In most cases MEMORY_TABLE should be used on the outer side of a nested loop join with the database tables.
- This allows to efficiently use the database indexes

Performance – Inserting from ABAP table into DB table

ABAP SQL

```
INSERT <DBTAB> FROM TABLE <ITAB> [ ACCEPTING DUPLICATE KEYS ].
```

This ABAP SQL statement inserts all rows from an ABAP table into a DB table. Duplicate Key errors are ignored.

The statement used to be translated into a non-atomic CLI array INSERT where SQL0803N errors were ignored.

Using MEMORY_TABLE and IGNORE DUPLICATES this can now be executed as atomic INSERT statement.

Performance – Inserting from ABAP table into DB table

```
INSERT INTO "DBTAB"
( SELECT * FROM TABLE( SAPTOOLS.MEMORY_TABLE( CAST( ? AS BLOB( 2G )) )
CARDINALITY 200000 ) AS DB6_MEMORY_TABLE ( <column definition> ) )
IGNORE DUPLICATES
```

Atomic INSERT statements can benefit from parallel INSERT on BLU target tables. The CARDINALITY clause can be used to influence the degree of INSERT parallelism.

ABAP Mass DELETE and Mass UPDATE statements can be translated to Atomic statements in a similar way. Mass UPDATE is a little more tricky. We still need some optimizer enhancements for good performance.

Performance – Using Optimizer Guidelines in statement text

ABAP SQL

```
SELECT VERSION FROM SVERS
  %_HINTS DB6 '<IXSCAN TABLE=''SVERS'' SAP_INDEX=''0'' />'.
```

ABAP SQL statements can contain hints that are translated to Db2 Optguidelines.

We prefer Optguidelines over Db2 Optimizer Profiles because Optguidelines

- are easier visible in SQL Cache monitoring
- can be contained in the ABAP code and can be added at prepare time

```
SELECT VERSION FROM SVERS
/* <OPTGUIDELINES><IXSCAN TABLE='SVERS' INDEX='SVERS~0' /></OPTGUIDELINES> */
```

Performance – REOPT Guidelines

ABAP SQL uses parameter markers for input parameters in the statement text. REOPT Guidelines can be used when this takes away too much information from the Db2 optimizer.

```
SELECT * FROM "SVERS" WHERE VERSION = ? WITH UR
/* <OPTGUIDELINES><REOPT='ONCE' /></OPTGUIDELINES> */
/* with ? = '%' or ? = 'NOENTRY' */
```

REOPT ONCE generates the final access path when the first set of input parameters is available

REOPT ALWAYS generates the final access path at every execution
This guideline should be used in exceptional cases only for statements with high execution time that are not too often executed.

Lock Wait Monitoring – tool db6util

```
> db6util -sl -lock2key

DEADLOCKS:
---------

Deadlock 1 :
                2919                2939                2919
  <--    (PID:15235)  <--    (PID:15326)  <--    (PID:15235)  <--
               db2bp               db2bp               db2bp


LOCK WAITS:
----------

DETAILED INFORMATION ABOUT LOCKED PROCESSES:
-------------------------------------------
```

SAP tool „db6util“
can be used
for ad hoc monitoring
of lock wait
or deadlock situations.

The tool shows the application handles
and more detailed information
about the lock wait participants.

Lock Wait Monitoring – tool db6util details

```
DETAILED INFORMATION ABOUT LOCKED PROCESSES:
----------------------------------------

      ID       PID        APPL-NAME      HOSTNAME(MEMB)   MODE RMODE OBJECT TABLE

    2919     15235            db2bp        DB-Server(0)      X     U    ROW SAPFMH.FMH_LOCK

Status      : DEADLOCKED (507 seconds)
Lock Name   : 060055130500000000000000052
Row Lock    : SCHEMA       = "SAPFMH"
              TABNAME      = "FMH_LOCK"
              DBPARTITION  = 0
              RID          = 5
              Key data:
               I           = '2'
Last SQL    : update FMH_LOCK set j = 4711 where J = 2 with CS wait for
              outcome

    2939     15326            db2bp        DB-Server(0)      X     U    ROW SAPFMH.FMH_LOCK

Status      : DEADLOCKED (2 seconds)
Lock Name   : 060055130600000000000000052
Row Lock    : SCHEMA       = "SAPFMH"
              TABNAME      = "FMH_LOCK"
              DBPARTITION  = 0
              RID          = 6
              Key data:
               I           = '3'
Last SQL    : update FMH_LOCK set j = 4711 where J = 3 with CS wait for
              outcome
```

The details section contains

- the last SQL statement
- the held and required lock types
- the locked object
- in case of row locks: the locked key values
- the lock name
- …

Step 1: Lock Wait Monitoring – MON_GET_APPL_LOCKWAIT

```
WITH LW AS (
  SELECT
    W.REQ_APPLICATION_HANDLE REQ_APPL_HDL, W.HLD_APPLICATION_HANDLE HLD_APPL_HDL,
    W.REQ_MEMBER REQ_MEMBER, W.HLD_MEMBER HLD_MEMBER, W.LOCK_NAME LOCK_NAME,
    TIMESTAMPDIFF( 2, CHAR( CURRENT TIMESTAMP - W.LOCK_WAIT_START_TIME )) WAIT_TIME,
    W.LOCK_MODE LOCK_MODE, W.LOCK_MODE_REQUESTED REQ_LOCK_MODE, W.REQ_EXECUTABLE_ID EXECUTABLE_ID,
    CASE LOCK_OBJECT_TYPE WHEN ... END AS LOCK_OBJECT_TYPE,
    T.TABSCHEMA TABSCHEMA, T.TABNAME TABNAME, T.TBSPACE TBSPACE
  FROM TABLE( MON_GET_APPL_LOCKWAIT(NULL,-2) ) AS W
  LEFT OUTER JOIN SYSCAT.TABLES AS T ON W.TBSP_ID = T.TBSPACEID AND W.TAB_FILE_ID = T.TABLEID )
SELECT
  REQ_APPL_HDL, HLD_APPL_HDL, REQ_MEMBER, HLD_MEMBER, WAIT_TIME, LOCK_NAME,
  LOCK_MODE, REQ_LOCK_MODE, EXECUTABLE_ID, LOCK_OBJECT_TYPE, TABSCHEMA, TABNAME, TBSPACE, 'LOCKWAIT'
FROM LW
UNION
SELECT
  HLD_APPL_HDL,          -1, HLD_MEMBER,         -1,        -1,      NULL,
  'NON',         'NON',          NULL,               '',        '',      '',      '', 'UNKNOWN'
FROM LW
WHERE HLD_APPL_HDL IS NOT NULL AND HLD_APPL_HDL NOT IN ( SELECT REQ_APPL_HDL FROM LW )
ORDER BY 1 WITH UR;
```

MON_GET_APPL_LOCKWAIT is used to retrieve all participants in a lock wait chain.

Step 2: Lock Wait Monitoring – using MON_GET*

```
SELECT
  CO.CLIENT_PID, CO.COORD_MEMBER, CO.APPLICATION_ID, CO.APPLICATION_NAME,
  CO.CLIENT_USERID, CO.CLIENT_WRKSTNNAME, CO.CLIENT_APPLNAME, CO.CLIENT_ACCTNG,
  DET.WORKLOAD_OCCURRENCE_STATE, DET.LAST_EXECUTABLE_ID
FROM TABLE( MON_GET_CONNECTION( ?, -2 ) ) AS CO
LEFT OUTER JOIN TABLE( MON_GET_UNIT_OF_WORK_DETAILS( CO.APPLICATION_HANDLE, CO.COORD_MEMBER) ) AS MET
  ON CO.APPLICATION_HANDLE = MET.APPLICATION_HANDLE ,
XMLTABLE ( XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'), '$det/db2_unit_of_work'
  PASSING XMLPARSE(DOCUMENT MET.DETAILS) as \"det\"
  COLUMNS \"WORKLOAD_OCCURRENCE_STATE\" VARCHAR(20) PATH 'workload_occurrence_state',
          \"LAST_EXECUTABLE_ID\" CHARACTER(64)      PATH 'last_executable_id'
) AS DET
WHERE CO.MEMBER = CO.COORD_MEMBER ;
```

MON_GET_CONNECTION and MON_GET_UNIT_OF_WORK_DETAILS are used to retrieve status and last executable id for lock participants.
After this MON_GET_PKG_CACHE_STMT is used to retrieve the last SQL statement.

```
SELECT STMT_TEXT FROM TABLE ( MON_GET_PKG_CACHE_STMT( NULL, ? , NULL ,? ) )
```

Step 3: Lock Wait Monitoring – resolving row lock names

```
SELECT
  MAX( CASE WHEN NAME = 'LOCK_OBJECT_TYPE'  THEN RTRIM(VALUE) END ),
  MAX( CASE WHEN NAME = 'TABSCHEMA'         THEN RTRIM(VALUE) END ),
  MAX( CASE WHEN NAME = 'TABNAME'           THEN RTRIM(VALUE) END ),
  MAX( CASE WHEN NAME = 'DATA_PARTITION_ID' THEN VALUE END ),
  MAX( CASE WHEN NAME = 'PAGEID'            THEN VALUE END ),
  MAX( CASE WHEN NAME = 'ROWID'             THEN VALUE END ),
  MAX( CASE WHEN NAME = 'TSNID'             THEN VALUE END )
  FROM TABLE( MON_FORMAT_LOCK_NAME( ? ) ) WITH UR;
```

For row locks MON_FORMAT_LOCK_NAME can be used to retrieve more information about the locked row.

Step 4: Lock Wait Monitoring – retrieve the locked row

For row based tables an old blog from Serge Rielau described how to calculate the RID .

https://community.ibm.com/community/user/hybriddatamanagement/viewdocument/have-lock-seek-row-a-handy-functi?CommunityKey=ea909850-39ea-4ac4-9512-8e2eb37ea09a&tab=librarydocuments

```
rid = datapartitionid )<<48 + pageid<<16 + slotid;
```

After this the locked row can be retrieved via SQL .

```
SELECT <key_columns> FROM <table> AS T WHERE RID(T) = ? AND DBPARTITIONNUM( T.<colname> ) = ?
```

Both UR and CS isolation may have to be tried depending on the pending operation.

Step 4b: Lock Wait Monitoring – retrieve the locked row (BLU)

For BLU table the TSNID can be used directly as input for the RID function to retrieve the locked row.

```
SELECT <key_columns> FROM <table> AS T WHERE RID(T) = ?
```

Prerequisite:

Db2 V11.5.4.0 or higher

IDUG

SQL and Configuration Tips and Tricks Learned from the SAP/Db2 Project

Joern Klauke, Edgar Maniago, Phil King

jklauke@de.ibm.com
edgar.maniago@sap.com
phil.leslie.king@ibm.com

SQL3

Please fill out your session evaluation!

@IDUGDb2
#IDUG_NA24