

DB2 Regional User Group Tour 2008
More Batch, Less Time

Agenda

- What causes slow down in batch?
- Design considerations for batch applications?
- Tools Needed for Tuning
- What's different about tuning batch?
- HOWTO
 - locate expensive or long-waiting SQL
 - identify and eliminate contention problems between DB2 batch jobs (-911, -913)
 - improve DB2 batch throughput and increase production workflow
 - develop safe and predictable recovery positions for all DB2 production batch jobs

Batch Terminology

- **Batch Job**
A series of programs running in the background without on-line interaction
- **Batch Window**
A series of Batch Jobs that must run at a particular time (daily, weekly, monthly) and must complete in a given time frame. This time frame is usually when on-line access is restricted.

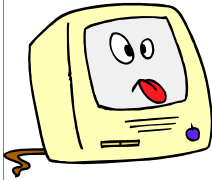
Tuning is a team effort

- **Application Management**
Code Reviews
- **Database Management**
SQL Reviews (Access Tuning)
Index Design
- **DASD Management**
I/O Controllers
Dataset Placement
Cache
- **Systems Management**
CPU
Memory
Dispatching Priority
DB2 System Parameters (ZPARMS)
DB2 Bufferpools



What causes slow down in batch?

- Internal Delays
- External Delays
 - Wait for CPU
 - Wait for Channel I/O
- Contention causes a traffic jam
- Single thread JOBS
- Poor running SQL
- Holding locks



Major Goals when designing Batch

- Reduce Elapse Time
 - Parallel Processing
 - Batch Stream - One Job Step Per Job (DB2 access)
- Reduce CPU usage
 - Filter non-qualifying rows as soon as possible (in DB2 not App)
 - Select only the columns you need
 - Minimize SQL Calls
 - Design for Index Lookaside
- Reduce I/O
 - Design for asynchronous (Sequential Prefetch)
 - Design to access in cluster order
 - Denormalize
- Checkpoint Restart
 - How Long will it take to rollback your work?
- Continuous Availability
 - Design for online access at the same time.

The Best Performing SQL Statement

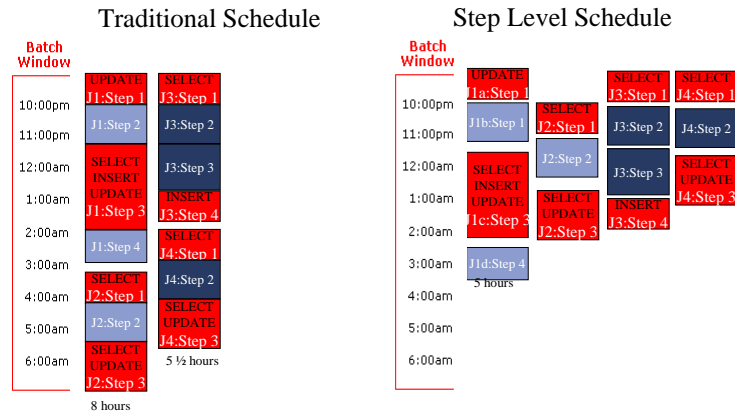
- Is one that does not execute.



Reduce Elapse Time: Parallel Processing

- Process class of data in parallel
 - Unique Key Range
 - Tablespace Partitions
 - Functional Grouping (Region, Country, Department)
- Query Parallelism (CPU, IO, SYSPLEX)
 - Bind Option: DEGREE(ANY)
 - PROS: Avoid program and JCL duplication
 - DB2 can determine number of parallel engines per statement
 - DB2 can enable SYSPLEX parallelism in data sharing
 - CONS: DB2 I/O, CPU, and Sysplex parallelism are for read only SQL
 - DB2 can turn off parallelism at runtime
 - DB2 controls the number of parallel task at runtime

Reduce Elapse: Job Step Level Scheduling



Reduce CPU: Filter non-qualifying rows (incorrect)

```

➤ EXEC SQL
  SELECT * FROM TBL1
  WHERE key_col = :ws-key-col
END-EXEC

IF tbl1.col1 = ws-active THEN
  process active
  . . .
ELSE
  get next row
END-IF.

```

Reduce CPU: Filter non-qualifying rows (correct)

```
➤ EXEC SQL
  SELECT * FROM TBL1
  WHERE key_col = :ws-key-col
     and col1 = :ws-active
END-EXEC.
```

```
process data
...
get next row
```

Reduce CPU: Select only needed columns (incorrect)

```
➤ EXEC SQL
  SELECT col1, col2, col3, col4
  FROM TBL1
  WHERE key_col = :ws-key-col
     and col1 = :ws-active
END-EXEC.
```

Index is
COL1, COL2, COL3

```
process data
...
get next row
```

Reduce CPU: Select only needed columns (correct)

- EXEC SQL
 SELECT col2, col3
 FROM TBL1
 WHERE key_col = :ws-key-col
 and col1 = :ws-active
END-EXEC.

process data
...
get next row
- Index is
COL1, COL2, COL3

Index only ACCESS

Reduce CPU: Minimize SQL Calls

- Counting Rows or calculating
 Push calculation function into DB2 SELECT COUNT(*)
 Each SELECT cost a few thousand instructions.
- Use SELECT not CURSOR for 1 row
 A cursor requires thousands more instructions than a select
 statement.
 DB2 V8: SELECT COL1, COL2
 FROM TBL1
 WHERE condition
 FETCH FIRST 1 ROW ONLY

Reduce CPU: Stage 1 Predicate

➤ DB2 9 – Performance Monitoring and Tuning Guide

<u>Predicate Type</u>	<u>Index</u>	<u>Stage1</u>
COL = <i>value</i>	Y	Y
COL = <i>nocol expr</i>	Y	Y
COL IS NULL	Y	Y
COL op <i>value</i>	Y	Y
COL BETWEEN <i>value</i> and <i>value</i>	Y	Y
.....		
COL IS NOT NULL	Y	Y
XMLEXISTS	Y	N
.....		
T1.COL1 = T1.COL2	N	N
COL = (<i>cor subq</i>)	N	N

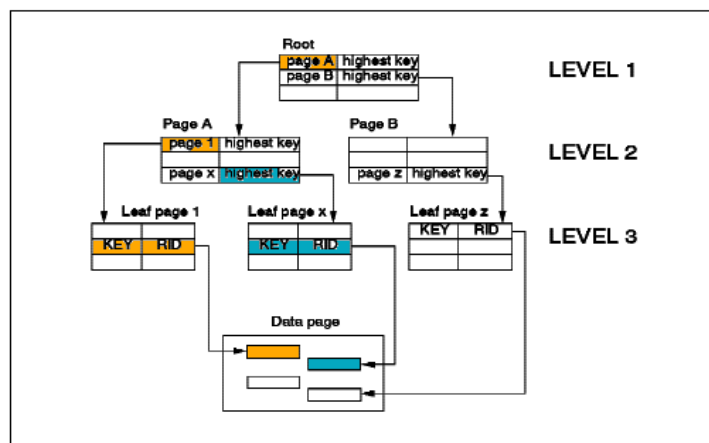
Reduce CPU: Static SQL /REOPT(ALWAYS)

- Use **STATIC SQL**
- If you must use dynamic to get a better access path at run time then use **REOPT(ALWAYS)** synonym **REOPT(VARS)**.
- **REOPT(ALWAYS)** is 10% cheaper than **PREPARE** for dynamic because authorization checks are already done.
- Isolate **REOPT** statements into a called I/O module when using **(ALWAYS)**

Reduce CPU: Index Look-aside

- Index Look-aside reduces the number of index/data pages accessed by remembering the current data, leaf, and immediate parent non-leaf page position across repeated executions of a certain DB2 process.
- You get maximum CPU savings by accessing the data in the table in sequence of the clustering index
- Sort your input data in cluster order

Reduce CPU: Index Lookaside (path)



Reduce I/O: Denormalized Table (SUB-TYPE)

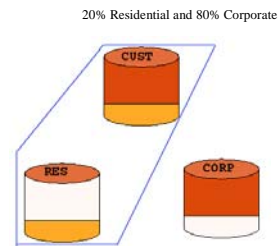
➤ Normalized

```
SELECT P.name, P.address, P.phone
      , R.call_time
FROM CUST P
      , RES R
WHERE P.CUST_ID = R.CUST_ID
```

➤ De-Normalized Flag

Save sub-type indicator on parent table.

```
SELECT P.name, P.address, P.phone
      , R.call_time
FROM CUST P
      , RES R
WHERE P.CUST_ID = R.CUST_ID
      AND P.CUST_TYPE = 'R'
```



Reduce I/O: Denormalized Table(Repeating)

➤ Normalized

```
SELECT P.CUST_ID
      , SUM(P.bill_amt) as YTD_AMT
FROM CUST P
WHERE P.CUST_ID = :ws-cust-id
GROUP BY P.CUST_ID
```

➤ De-Normalized Flag

```
SELECT P.CUST_ID
      ,( P.jan_amt + P.feb_amt + mar_amt +
        P.apr_amt + P.may_amt + jun_amt +
        P.jul_amt + P.aug_amt + P.sep_amt +
        P.oct_amt + P.nov_amt +
        P.dec_amt) as YTD_AMT
FROM CUST P
WHERE P.cust_id = :ws-cust-id
```

Cust_id	Month	Amt
100126	1	25.21
100126	2	29.94
100126	3	27.21
100126	4	30.16
100126	5	29.43
100126	6	30.21
100126	7	33.76
100126	8	26.58
100126	9	33.21
100126	10	34.21
100126	11	35.21
100126	12	55.62

Cust_id	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
100126	25.21	29.94	27.21	30.16	29.43	30.21	33.76	26.58	33.21	34.21	35.21	55.62

Reduce I/O: Clustering Order

- Your PRIMARY KEY does not have to be the clustering order. Make sure the clustering index will support most of your join processing and batch sequential processing.
- File Processing –sorted first
 - Sequential Prefetch – detection at runtime
 - Pre-Read up to 32 pages into memory
 - Reduced I/O by not re-reading a page
- OPEN CURSOR
 - ORDER BY – CLUSTER ORDER (Avoid Sort)
 - When monitoring this statement should be fast

Reduce CPU/IO: PREFETCH

- Preferred Method: Sequential or Dynamic Prefetch
32 pages read in 1 Asynchronous I/O
- Sequential Prefetch
Decided by optimization at bind time.
Each page read cost 1.6 msec
- Sequential Detection (Dynamic Prefetch)
Triggered at execution time.
Each page read cost 1.6 msec
Group of pages (32) read (may contain pages not needed)
- List Prefetch (non cluster access)
Read only pages needed
Requires a sort
Cost range from 1.6 to 20 msec per page

Reduce CPU/IO: Build Reference Tables

- Load small lookup tables into memory
 - Save on thousands of selects

- Load Medium Lookup tables
 - When Small percentage of data is needed
 - Search lookup table if not found then read DB2 and add the value to the lookup table.

Continuous Availability: Lock Size

- ANY (default) -DB2 chooses for you
 - Pro: Works most of the time. Good for ERP packages
 - Con: Can escalate a page lock to tablespace Lock
- PAGE – Is a good design default.
 - Pro: Will use system parameters to control application lock usage
 - Con: Application can fail if they don't commit.
- ROW – Is best option to improve concurrency
 - Pro: Provides most granular locking
 - Con: Potentially high CPU cost
- TABLE – Multiple tables in a segmented tablespace
- TABLESPACE – Single Table, Partitioned Tablespace

Continuous Availability: Checkpoint Restart

- **Commit for concurrency**
Variable commit frequency to release locks sooner
5 – 15 seconds
- **Commit for restart ability**
Commit to reduce rollback time.
5 – 15 minutes
- **Design to remove Batch Window (run with onlines)**
Design program to run with onlines
Commit every 1 to 5 seconds
- **Design for Checkpoint – z/OS, IMS, CICS**

Current Batch Remedies

- **Get more CPU to get the jobs done faster**
- **Low Return on Investment**
- **Software Cost are Higher than Hardware Cost**
Hardware (CPU) \$ 250,000.00
Software \$1,000,000.00

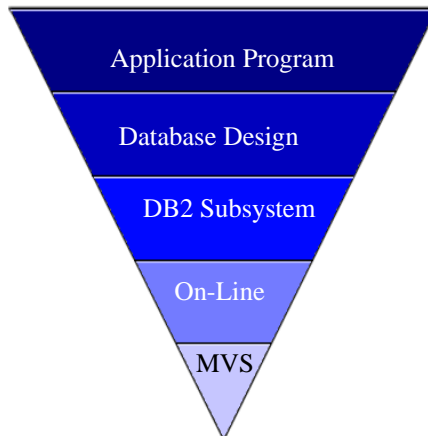
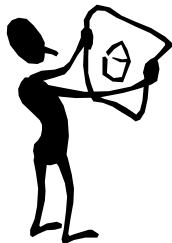


Tools Needed for Tuning

- Application Monitor
- DB2 Database Monitor
- MVS System Monitor
- DASD Monitor



Performance Triangle



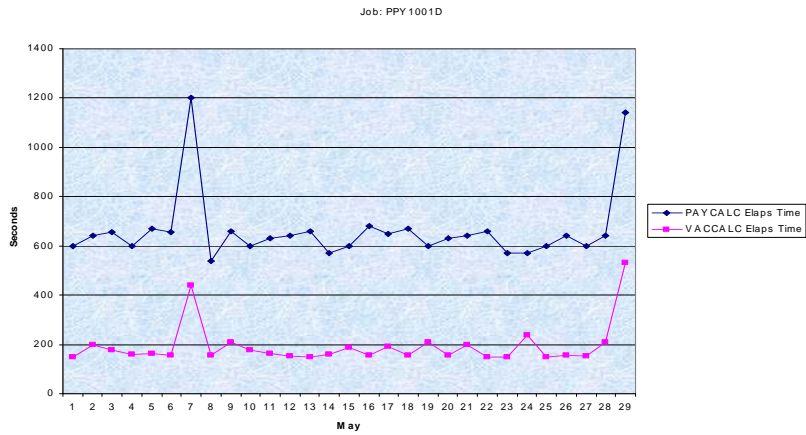
What's different about tuning batch?

- Perception
 - Batch is expected to be slow so why tune?
- Difficult to Monitor
 - Monitors do not report at the JOB STEP level
 - Most issues at night when skilled personnel are sleeping
 - Error messages usually do not return the needed information back from DB2.
- No Squeaky Wheel (On-Line User)
- Tuning Resources
 - What's the cost of tuning?
 - What are the savings in tuning?
 - Charge Back (CPU Cost)

What job is running long

- First pass any jobstep taking more than 15 minutes
- How do you get this information?
 - Look at SYSOUT and record in a DB2 Performance Warehouse:
 - **JOB Name, Step Name, Program Name, Plan Name**
 - **Begin Timestamp, End Timestamp, Elapse Time**
- Trend Report
 - Graph the Elapse time Per Job
 - Graph the Elapse time Per Jobstep
- Contention Report (Scheduling)
 - What jobs are running at the same time?
 - What job steps are running at the same time using same tables?

Job Step Trend Analysis



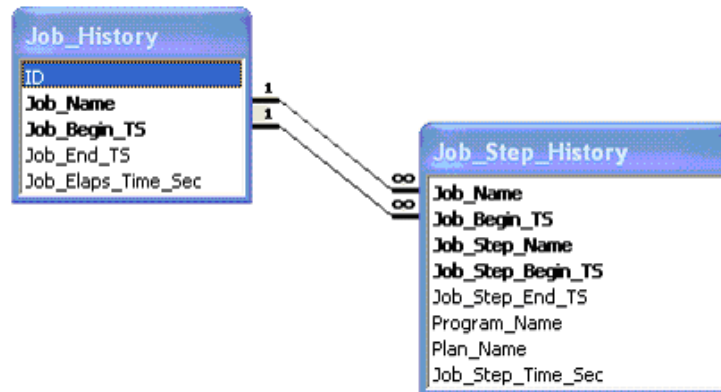
Contention Report

Job Contention Report
05/07/2004

Job: PHR2002W Begin 1:19:55 End: 1:28:16

Job Step	Table Name	Contending Job	Job Step	Program
VACCALC	PROD.EMPLOYEE	PPY1001D	PAYCALC	PAYCALC10
		PHR3100D	HR310010	HRBEN100
	EMP_PROJ_ACTIVITY	PPY1001D	PAYCALC	PAYCALC10
PROJECT		PPY1001D	PAYCALC	PAYCALC10
		PHR3100D	HR310010	HRBEN100

Performance Warehouse Model



Performance Warehouse: Job History (COBOL DCLGEN)

```
EXEC SQL DECLARE JOB_HIST TABLE
( JOB_NAME           CHAR(8) NOT NULL,
  JOB_BEGIN_TS       TIMESTAMP NOT NULL,
  JOB_END_TS         TIMESTAMP NOT NULL,
  JOB_ELAPSE_TM_SEC  DECIMAL(15, 2) NOT NULL
) END-EXEC.
01 DCLJOB-HIST.
   10 JOB-NAME           PIC X(8).
   10 JOB-BEGIN-TS       PIC X(26).
   10 JOB-END-TS         PIC X(26).
   10 JOB-ELAPSE-TM-SEC  PIC S9(13)V9(2) USAGE
                           COMP-3.
```

Performance Warehouse Job Step History (COBOL DCLGEN)

```
EXEC SQL DECLARE JOB_STEP_HIST TABLE
( JOB_NAME          CHAR(8) NOT NULL,
  JOB_BEGIN_TS      TIMESTAMP NOT NULL,
  JOB_STEP_NAME     CHAR(8) NOT NULL,
  JOB_STEP_BEGIN_TS TIMESTAMP NOT NULL,
  JOB_STEP_END_TS   TIMESTAMP NOT NULL,
  PROGAM_NAME       CHAR(8) NOT NULL,
  PLAN_NAME         CHAR(8) NOT NULL,
  JOB_STEP_TIME_SEC DECIMAL(15, 2) NOT NULL
) END-EXEC.
01 DCLJOB-STEP-HIST.
   10 JOB-NAME          PIC X(8).
   10 JOB-BEGIN-TS     PIC X(26).
   10 JOB-STEP-NAME    PIC X(8).
   10 JOB-STEP_BEGIN-TS PIC X(26).
   10 JOB-STEP_END-TS  PIC X(26).
   10 PROGRAM-NAME     PIC X(8).
   10 PLAN-NAME        PIC X(8).
   10 JOB-STEP-TIME-SEC PIC S9(13)V9(2) USAGE COMP-3.
```

Calculate JOB Elapse Time in Seconds

```
SELECT
  ( DECIMAL(DAYS(JOB_END_TS)) * 86400
    + HOUR  (JOB_END_TS) * 3600
    + MINUTE (JOB_END_TS) * 60
    + SECOND (JOB_END_TS) )
-
  ( DECIMAL(DAYS(JOB_BEGIN_TS)) * 86400
    + HOUR  (JOB_BEGIN_TS) * 3600
    + MINUTE (JOB_BEGIN_TS) * 60
    + SECOND (JOB_BEGIN_TS) )
FROM JOB_HIST
WITH UR;
```

Locate Expensive or Long-Waiting SQL

- What jobs and job steps are running long?
- Is the job step using DB2?
- Is the SQL statement getting CPU time?
- Is the SQL having contention?
- Are other jobs waiting?

Identify Long Running SQL

- Using Monitor
- Build your own: SQL Counts and Elapse Times
- Detail or Summary

Identify Long Running SQL: Monitor

- **Turn on DB2 Accounting Trace Class 1, 2, 3, 7 and 8**
 - Class 1: Standard Accounting Data (Total Thread Time)
 - Class 2: Separate out Application and DB2 time (DB2 Time)
 - Class 3: Elapsed Wait Time in DB2 (Wait Time)
 - Class 7: Separate out Application and DB2 time for PACKAGE
 - Class 8: Elapsed Wait Time in DB2 for PACKAGE

- **Using DB2 PM report on long running SQL**
 - Accounting Short Report

- **Some shops will not turn these traces on!**
 - Class 1 < 1%
 - Class 2 add 2.5% to CICS, 10% to batch
 - Class 3 < 1%
 - Class 7 – minimal if class 2 is on
 - Class 8 – minimal if class 3 is on

DB2PM: Accounting Short Report

- **Class 1, Class 2, Class 3 times**
 - 1: Total Thread Time
 - 2: In DB2 Time
 - 3: In DB2 Time Waiting on some resource

- **DB2 Response Time (Elapse Time)**

- **Resources Used (CPU, IO)**

- **Lock Suspension**

- **Package Level Class 7, 8**
 - (CPU, IO, Wait, and Lock Wait)

Monitor for Developer: Build Your Own

- **Build Common Performance Routine to gather data**
 - Job, Job Step, Begin Step Timestamp, End Step Timestamp
 - Statement Number, Execution Count,
 - Average Elapse Time, Max Elapse Time
 - DB2 Table Creator, DB2 Table Name

- **SYSOUT - Job / Step (Elapse Time & CPU)**
- **SQL Statement Level (Elapse Time & CPU)**
 - Modify Program to Call Performance Data Collector
 - Purchase product that does not require code changes

- **Report on Performance by Job and Job Step**

Job Performance Warehouse: Summary or Detail

- **Summary should be used for day to day operations**
 - Identify Long Running Jobs, Job Step, Program and Statements
 - Trend Elapse Time

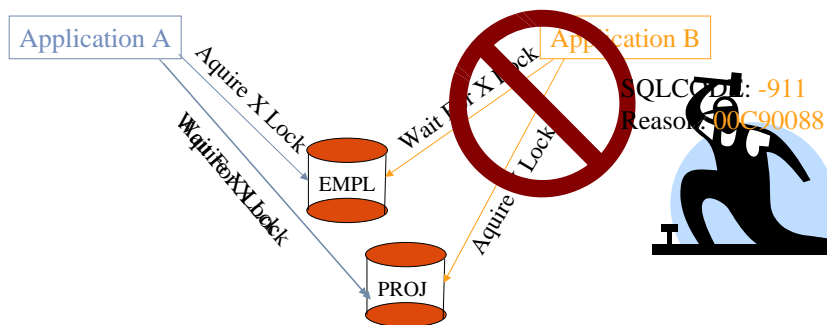
- **Once a problem is identified work with DBA team to run a detail trace and produce an Accounting Trace – Long Report**

- **Store Performance Data in GDG or DB2?**
 - If no DBA support then store in GDG otherwise DB2 for ease of reporting

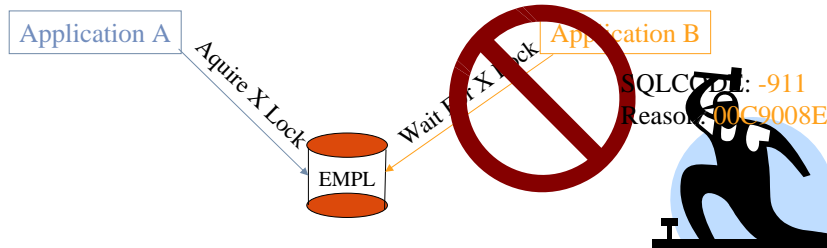
Identify and eliminate contention problems

- **Performance Warehouse**
Identify programs running at the same time using the same tables
- **SQLCODE -911,-913**
Report the entire SQLCA not just SQLCODE
- **Eliminate Contention:**
 - Change the Schedule
 - Add Commit Processing
 - Tune Locking: DB2 System Params "zparms"
 - Change Bind Parameters
 - **ISOLATION (CS)**
 - **CURRENTDATA(NO)**

What is a Deadlock?



What is a Timeout?



SQLCA – SQL Communication Area

-911 THE CURRENT UNIT OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR TIMEOUT. REASON reason-code, TYPE OF RESOURCE resource-type, AND RESOURCE NAME resource-name

- SQLERRD(3) also contains the reason-code which indicates whether a deadlock or timeout occurred. The most common reason codes are:
 - 00C90088 - deadlock
 - 00C9008E - timeout
- How do I find what job caused the deadlock/timeout?
 - DB2 Master Started Task or z/OS System Log

Common Resource Types : DSNT500I

- See DSNT500I for an explanation of 'resource-type'

TYPE Code	Type of Resource
00000100	Database
00000200	Table space
00000201	Index space
00000210	Partition
00000D00	DBID/OBID
00000D01	DBID/OBID

How to Find Resource Name

```
DSNT408I SQLCODE = -911, ERROR: THE CURRENT UNIT  
OF WORK HAS BEEN ROLLED BACK DUE TO DEADLOCK OR  
TIMEOUT. REASON 00C9008E, TYPE OF RESOURCE  
00000D01, AND RESOURCE NAME 00000403.00000043
```

- For Type 'D00' and 'D01', find name using:

```
SELECT CREATOR, NAME  
FROM SYSIBM.SYSTABLES  
WHERE DBID = 403 AND OBID = 43;
```

Who has the Resource?(DB2 Master Log)

```
20.19.43 STC09299 DSNT376I - PLAN=DSNESPCS WITH
CORRELATION-ID=CSBP
CONNECTION-ID=TSO
LUW-ID=P390.DSN1LU.BB974B7A1C85=387
THREAD-INFO=CSBP:***
IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=DSNESPCS
WITH
CORRELATION-ID=CSBN
CONNECTION-ID=TSO
LUW-ID=P390.DSN1LU.BB974B2FA85E=386
THREAD-INFO=CSBN:***
ON MEMBER DSN1
20.19.43 STC09299 DSNT501I - DSNILMCL RESOURCE UNAVAILABLE
CORRELATION-ID=CSBP
CONNECTION-ID=TSO
LUW-ID=*
REASON 00C9008E
TYPE 00000D01
NAME 00000403.00000043
```

DB2 V8: GET DIAGNOSTICS

- **Statement Information**
Provide information about the last statement executed
- **Condition Information**
Provide information about each error condition within the execution of the last statement
- **Combined Information**
Provides the TEXT representation of all the information gathered about the execution of the SQL statement in one continuous string separated by semicolons

GET DIAGNOSTICS – Statement Info

:host-variable = statement-information-item-name

GET DIAGNOSTICS :hv-number = NUMBER

DB2_LAST_ROW
DB2_NUMBER_PARAMETER_MARKERS
DB2_NUMBER_RESULT_SETS
DB2_RETURN_STATUS
DB2_SQL_ATTR_CURSOR_HOLD
DB2_SQL_ATTR_CURSOR_ROWSET
DB2_SQL_ATTR_CURSOR_SCROLLABLE
DB2_SQL_ATTR_CURSOR_SENSITIVITY
DB2_SQL_ATTR_CURSOR_TYPE
MORE
NUMBER
ROW_COUNT

GET DIAGNOSTICS – Condition Info

**GET DIAGNOSTICS CONDITION :I
:hvDB2message = MESSAGE_TEXT**

CATALOG_NAME	DB2_RETURNED_SQLCODE	DB2_AUTHENTICATION_TYPE
CONDITION_NUMBER	DB2_ROW_NUMBER	DB2_AUTHORIZATION_ID
CURSOR_NAME	DB2_SQLERRD_SET	DB2_CONNECTION_ID
DB2_ERROR_CODE1	DB2_SQLERRD1	DB2_CONNECTION_STATE
DB2_ERROR_CODE2	DB2_SQLERRD2	DB2_CONNECTION_STATUS
DB2_ERROR_CODE3	DB2_SQLERRD3	DB2_ENCRYPTION_TYPE
DB2_ERROR_CODE4	DB2_SQLERRD4	DB2_SERVER_CLASS_NAME
DB2_INTERNAL_ERROR_POINTER	DB2_SQLERRD5	DB2_PRODUCT_ID
DB2_MESSAGE_ID	DB2_SQLERRD6	
DB2_MODULE_DETECTING_ERROR	DB2_TOKEN_COUNT	
DB2_ORDINAL_TOKEN_n	MESSAGE_TEXT	
DB2_REASON_CODE	RETURNED_SQLSTATE	
	SERVER_NAME	

GET DIAGNOSTICS – Combined Info

```
GET DIAGNOSTICS :hv_varchar = ALL STATEMENT
```

➤ **:HV_VARCHAR** is a string that will contain all statement information delimited by a semicolon for all conditions from the last statement.

Sample:

```
CONDITION_NUMBER=1;RETURNED_SQLSTATE=02000;  
RETURNED_SQLCODE=100;CONDITION_NUMBER=2;RET  
URNED_SQLSTATE=01004
```

Tune Locking

- **DB2 System Params: ZPARMS**
Locks Per User (10,000) – Panel DSNTIPJ
SQLCODE -904 “resource unavailable”
- **Lock Escalation**
LOCKMAX (1000) – Panel DSNTIPJ
- **Create/Alter Tablespace**
LOCKMAX (n, SYSTEM, 0)
LOCKSIZE (tablespace, table, page, row, any, lob)
MAXROWS n – limit number of rows on a page (1 – 255)
- **LOCK TABLE**
LOCK TABLE table-name PART int IN SHARE | EXCLUSIVE MODE

Lock Escalation Message

```
DSNI031I  - DSNILKES - LOCK ESCALATION HAS OCCURRED
FOR
  RESOURCE NAME = CSBP.EMPL
  LOCK STATE = X
  PLAN NAME : PACKAGE NAME = DSNTEP71 : DSNTEP2
  STATEMENT NUMBER = 001091
  CORRELATION-ID = CSBPT2
  CONNECTION-ID = BATCH
  LUW-ID = P390.DSN1LU.BB998E1FE90F
  THREAD-INFO = CSBP      : *                : *
```

How to improve throughput

- Increase MIPS
- Reduce I/O: Cache DASD, Compression
- Lock Avoidance
 - CURRENT DATA NO
 - With "UR"
- Reduce DB2 Sort
 - DISTINCT, ORDER BY, GROUP BY
- Fewer Steps in Batch Job
 - Schedule at the STEP LEVEL not JOB LEVEL

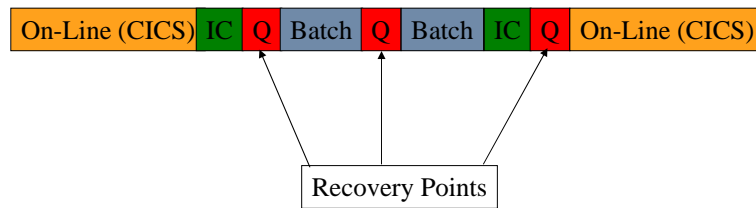
How to develop safe and predictable recovery positions for DB2 Batch Jobs

- Backup & Recovery
Imagecopies
- Point-In-Time Recovery
Quiesce
- Checkpoint Restart
Application Failure
- Unload/Load Files
Application Table backup
- Log Analyzer
Application Transaction Backout

What tablespaces are at risk?

```
SELECT TS.DBNAME, TS.NAME
FROM SYSIBM.SYSTABLESPACE TS
WHERE NOT EXISTS (SELECT 1
FROM SYSIBM.SYSCOPY C
WHERE C.DBNAME = TS.DBNAME
AND C.TSNAME = TS.NAME
AND C.ICTYPE = 'F'
)
ORDER BY 1,2
WITH UR;
```

Point in Time Recovery for DB2



Do you have a Recovery Disaster?

➤ Long Running UR CHECK FREQ

- ZPARM: DSN6SYSP URCHKTH

Console Message: DSNR035I

DSNR035I csect-name WARNING - UNCOMMITTED UR AFTER nn CHECKPOINTS -

CORRELATION NAME = xxxxxxxxxxxx CONNECTION ID = yyyyyyy LUWID = logical-unit-of-work-id=token PLAN NAME = xxxxxxxx AUTHID = xxxxxxxx END USER ID = xxxxxxxx TRANSACTION NAME = xxxxxxxx WORKSTATION NAME = xxxxxxxx

Trace Record : IFCID 0313

➤ UR LOG WRITE CHECK

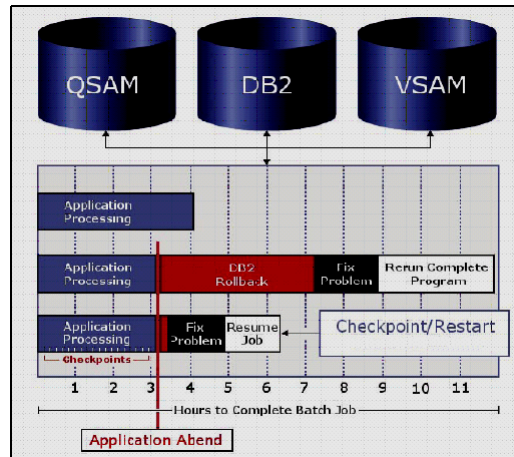
Console Message: DSNB260I

DSNB260I csect-name WARNING - A READER HAS BEEN RUNNING FOR nnnn MINUTES -

CORRELATION NAME = xxxxxxxxxxxx, CONNECTION ID = yyyyyyy, LUWID = LOGICAL-UNIT-OF-WORK-ID=token, PLAN NAME = xxxxxxxx, AUTHID = xxxxxxxx, END USER ID = xxxxxxxx, TRANSACTION NAME = xxxxxxxx, WORKSTATION NAME = xxxxxxxx

Trace Record : IFCID 0313

Checkpoint Restart



Resources for Batch Design

- **Manuals: IBM Information Center and Libraries**
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp>
- **Redbook: DB2 for OS/390 Application Design Guidelines for High Performance**
<http://www.redbooks.ibm.com/redbooks/pdfs/sg242233.pdf>
- **Software**
Softbase Batch Analyzer
<http://www.softbase.com/sbprod2.php>

DB2 Blogs and Social Networking

- Troy Coleman – DB2utor
<http://ibmsystemsmag.blogs.com/db2utor/>
- Willie Favero – Getting the most out of DB2 for z/OS
<http://blogs.ittoolbox.com/database/db2zos>
- Craig Mullins – Several blogs
<http://www.craigsmullins.com/blogs.htm>
- Planet DB2 – Blog Aggregator
<http://planetdb2.com/>
- Facebook – DB2 Group, MWDUG Group
<http://planetdb2.com/>
- Twitter – Social Micro Blog
Follow Willie Favero, Craig Mullins, Troy Coleman
<http://twitter.com/>

Q & A



Speaker Contact Information

Troy Coleman
Softbase
1-800-669-7076 X334
Direct: 1-847-776-0618
Email: troy.coleman@softbase.com

