DB2 Scripting Hacks

Roland Schock
ARS Computer und Consulting GmbH

Session Code: F13
16 October 2013, 15:45 - 16:45 | Platform: DB2 for LUW

- Get started with DB2 command line, CLPplus and the system commands
- Shells, platform differences, shell variables, proper quoting and calling db2 properly
- Pipes and Redirection
- Exploit GNU text utilities to get things done easily.
- Get along with Windows User Access Control

This session

- Get started with DB2 command line, CLPplus and the system commands
- Shells, platform differences, shell variables, proper quoting and calling db2 properly
- Pipes and Redirection
- Exploit GNU text utilities to get things done easily.
- Get along with Windows User Access Control

Motivation

- Why do we develop scripts
  - To make our life a bit easier.
  - We are lazy! [And that's good in this case!]
  - It is a sanity check for the DBA to incorporate scripts for daily checks and regular tasks.
  - We don't like to write tedious documentation.
- Graphical user interfaces are hard to automate.
- If we write our commands in a script file, we can reuse (automation) and we also have documented our steps.
- Can also be used as training materials for a new team member.

Motivation (cont.)

- Perils of a consultant
  - When you are called to customers to help; you don‘t want to tell them, we need a particular tools suite that costs $$$ to be able to help
  - We have to find ways work with the tools provided by db2
  - Perhaps, you can convince the customer to use some free tools, we can also leave behind without license fee issues
- All roads lead to Rome. ()
  - TIMTOWTDI = "There is more than one way to do it"
    So our examples might not be perfect… ;-)
  - We can review some samples a bit further in the presentation

Motivation (cont.)

- Advantages of Command line tools
  - Keep the results in text files, plus filter unwanted/unneeded parts from the output
    - System commands
    - GNU text utilities
- We have three categories of command line utilities
  - DB2 command line
    - 'db2' as an interactive shell or call-by-call, e.g. db2 list applications
  - DB2 system commands
    - Like db2ilist, db2diag or db2pd
  - DB2 graphical utilities
    - db2top

- Part 1 – DB2 Tools

DB2 Command Line Processor (db2clp)

- DB2 offers two ways to use the command line
  - 'db2' as an interactive shell with a command line prompt
    - Does not need the repetition of 'db2 ' on each line
    - No interpretation of OS shell specific characters, like * or brackets within the interactive shell
    - End the DB2 shell with 'quit' or 'terminate'
  - Using 'db2 ' as a prefix to commands from the operating system shell
    - Starts a separate instance of the db2 process for each command
    - Utilizes a background process to keep the connection to DB2 between invocations
    - Allows I/O redirection of operating system shell with <, >, >>, |

DB2 back-end process (db2bp) and the UNIX shell

`$ . /home/db2inst1/sqllib/db2profile`	No db2bp process has been created yet
`$ db2 list db directory`	db2bp process now exists.
`$ db2 connect to sample`	same db2bp process is re-used
`$ db2 "select count(*) from staff"`	…and re-used again
`$ db2 connect reset`	db2bp still exists after disconnecting.
`$ db2 terminate`	Now db2bp is gone

- Every distinct shell or script that interacts with DB2 is assigned a unique instance of db2bp.
- Multiple shells cannot share the same DB2 back-end process.
- Watch out for shell specific characters!
  The first two lines work, the last line breaks. Do You know why?
  - `$ db2 "select count(*) from staff"`
  - `$ db2 select "count(*)" from staff`
  - `$ db2 select count(*) from staff`

The last line breaks, as – depending on the platform and shell – we have the special characters '(', '*' and ')', which have a differente meaning in the shell. So we have to quote them. Either the whole db2 command or just the critical part.

Another option would be to use escape characters. Typically a backslash '\' is used for escaping. So we could also write to get a working query:

```
db2 select count\(\*\) from staff
```

Useful tricks with db2 command

- Trick: start 'db2 -t' to copy and paste db2 commands interactively from other scripts
  - Hint: Check for other useful parameters via 'db2 list command options'
  - Put your favorites in an environment variable:
    `export DB2OPTIONS='+a -c +ec -o -p'`
- Use ! to spawn any shell command from inside CLP
- Within the DB2 shell, you can start logging with
  `UPDATE COMMAND OPTIONS USING Z ON outfile.txt V ON`

Useful tricks with db2 command (cont.)

- To diagnose code page issues:
  - `db2 -a connect to dbname`
  - The SQLCA contains useful information for code page problems (See speaker notes for details)
- Use the return code of db2 for error handling in your scripts

Code	Description
0	DB2® command or SQL statement executed successfully
1	SELECT or FETCH statement returned no rows
2	DB2 command or SQL statement warning
4	DB2 command or SQL statement error
8	Command line processor system error

After a successful connect to the database, the user/application gets some information returned in the SQLCA data area:

- The second token in the SQLERRMC field (tokens are separated by X'FF') indicated the code page of the database. The ninth token indicates the code page of the application. If they are different, we will experience code page conversion.

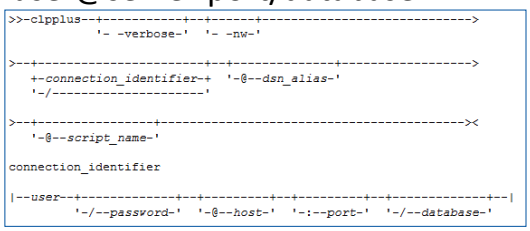
- The first and second entries in the SQLERRD array: SQLERRD(1) contains an integer value equal to the maximum expected expansion or contraction factor for the length of mixed character data, when converted from the applications code page to the database code page. SQLERRD(2) contains this value for conversions from database code page to application code page. A value of 0 or 1 indicated no expansion. A value greater 1 indicates a possible expansion in length; a negative value a possible contraction.

CLPPlus

- New since DB2 9.7
- Optional installation component of IBM Data Server Client
- It is not included in
  - DB2 Runtime Client
  - Data Server Driver Package
  - DB2 Express-C
- Requires at least Java JRE 1.5.x
- Can connect to DB2 LUW, DB2 z/OS and Informix via JDBC

CLPPlus Start

- From the Startmenu: Start → IBM DB2 → Command line tools → Command line Processor Plus
- On the command line: clpplus user@server:port/database



- Missing arguments will be requested for connect
- Database does not need to be cataloged
- Continuous improvements in upcoming DB2 fixpacks and versions

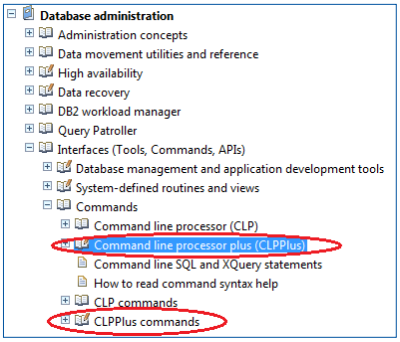
CLPPlus Befehle

- Editor-Buffer can be used for most recently used commands
- Commands for formating the data and pagination
- Simple configuration commands
- …

```
SQL> ? index

INDEX
-------

Type 'HELP [topic]' for command line help.

@                ACCEPT           APPEND           BREAK
BTITLE           CHANGE           CLEAR            CLPPLUS
COLUMN           COMPUTE          CONNECT          COPY
CURRENT_SCHEMA   DEFINE           DEL              DESCRIBE
DISCONNECT       EDIT             EXECUTE          EXIT
EXPORT           GET              GET_DBM_CFG      GET_DB_CFG
HELP             HOST             IMPORT           INDEX
INPUT            LIST             LOAD             PASSWORD
PAUSE            PRINT            PROMPT           QUIT
REMARK           REPFOOTER        REPHEADER        RESET_DBM_CFG
RESET_DB_CFG     SAVE             SERVEROUTPUT     SET
SHOW             SPOOL            START            TTITLE
UNDEFINE         UPDATE_DBM_CFG   UPDATE_DB_CFG    VARIABLE
WHENEVER
```

CLPPlus Example

```
SQL> CREATE PROCEDURE getEmployeeData( ID INT, OUT NAME char(10),
          OUT DOB Date, OUT SAL DECIMAL(7,2))
          LET NAME='dummy';
          LET DOB='10/10/2010';
          LET SAL=0;
     SELECT empname, empdob, salary INTO name, dob, sal FROM emp   WHERE empid = ID;
     END PROCEDURE;
/
DB250000I: The command completed successfully.

SQL> define var_id=1001         /* usage of substitution variable */
SQL> Variable name varchar(10)
DB250000I: The command completed successfully.
SQL> Variable dob date
DB250000I: The command completed successfully.
SQL> Variable salary double
DB250000I: The command completed successfully.

Call getEmployeeData(&var_id,  :name, :dob, :salary)
DB250000I: The command completed successfully.
SQL> Print name
'JOHN'
SQL> Print dob
'26/04/1982'
SQL> Print salary
10000.50
```



DB2 System Commands

- DB2 system commands to query the system
  - Did you have a look in the manuals lately? ;-)
  - Sometimes there's no other way to RTFM and it is usually interesting/enlightening.
  - I think it is like a cross word. Just look over it from time to time and discover new features.
- Have you heard about?
  - db2_local_ps, db2caem, db2ckbkp
  - db2flsn, db2logsforrfwd, db2osconf
  - db2tdbmgr, db2xprt, db2relocatedb

db2_local_ps

Outputs all of the DB2 processes running under an instance

db2caem automates the process for creating an activity event monitor,

Enabling capture for the statements of interest

Invoking the statements (each statement is rolled back after being executed to prevent side effects in the database)

Formatting the output information (including exporting activity information for the statements of interest and generation of formatted explain output from the captured section and section actuals).

db2ckbkp can be used to test the integrity of a backup image and to determine whether or not the image can be restored. It can also be used to display the metadata stored in the backup header.

db2flsn returns the name of the file in a log stream that contains the log record identified by a specified log sequence number (LSN) or log record identifier (LRI).

db2logsForRfwd list log files required for rollforward recovery command . It parses the DB2TSCHG.HIS file and allows a user to find out which log files are required for a table space rollforward operation.

DB2 System Commands: db2pd

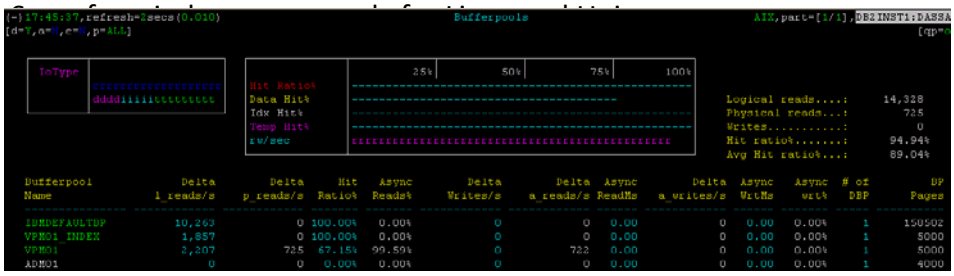
- db2pd
  - Direct access to DB2 system memory (i.e. ~~dirty~~ uncommitted read ;-)
  - Low level details, but very helpful for analysis
- Samples:
  - db2pd –db SAMPLE –locks // check for locks
  - db2pd –help | more
  - db2pd –db SAMPLE –tcbstats // lists table control block statistics
  - db2pd –edus interval=10 top=5// top five CPU consumers of last 10 secs
  - db2pd -alldbs -hadr | grep -e onnected -e artition -e "andby " -e "rimary " // multiple dbs

DB2 System Commands: db2pd and db2diag

- db2diag
  - db2diag –H // displays last 30 minutes of the db2diag.log
  - db2diag –A // archives the db2diag.log and creates a new one
  - db2diag –g db=TEST // shows the entries only for the Database TEST
  - db2diag -l severe -H 2d // last 2 days of severe errors
  - db2diag -gvi msg:=DB2_MAX_INACT_STMTS | more
  - db2diag -gi data:="password validation" -fmt "%{tsmonth}-%{tsday}-%{tshour}.%{tsmin}.%{tssec}\t%{database}\t%{data}" | sed "s/DATA.*bytes//;N;s/\n//"

```
2013-07-22-15.51.33   SAMPLE   Password validation for user janedoo   failed with rc = -2146500507
2013-07-22-15.51.59   SAMPLE   Password validation for user james007 failed with rc = -2146500502
```

DB2 System Commands: db2top

- Can be run interactive or in batch mode
  - db2top [—d dbname] [—n nodename] [—u username] [—p password] [—V schema] [—i interval] [—P <partition>] [—b option] [—a] [—B] [—k] [—R] [—x] [—f file </HH:MM:SS> <+offset>] [—D delimiter] [—C <option>] [—m duration] [—o outfile]
  - —b option, db2top will display information in CSV format.
- 

- Part 2 – The Shell

Different operating systems – different shells

- Platform differences
  - Windows CMD, PowerShell, (Posix-Shell with R2)
  - Linux: bash, csh, sh, pdksh, …
  - AIX: ksh, bash, csh, …
  - Solaris, HP-UX and other Unix dialects… ;-)
  - Different shells, mostly similar
  - See also http://en.wikipedia.org/wiki/Comparison_of_command_shells

Environment variables in Windows

- set DB2INSTANCE=db2
- echo %DB2Instance%
- Variables are not case sensitive

- In batch files use %% instead of %
  because single % is for variables on console
- Example:
  - for %i in (1 2 3) do echo %i

dbdo.cmd to execute *.db2 files with command interpreter

```
@echo off
if "%~x1" == ".db2" goto good
echo Please use this batch file with db2 commands in a .db2 file as parameter (termination char is ';').
goto leave

:good
for /f "tokens=2" %%i in ('date /t') do set today=%%i
for /f "tokens=1,2,3 delims=/" %%i in ("%today%") do set today=%%k%%j%%i
for /f "tokens=1" %%i in ('time /t') do set now=%%i
for /f "tokens=1,2 delims=:" %%i in ("%now%") do set now=%%i%%j

db2 -tvf %1 -z %~n1_%today%_%now%.out

:leave
```

Environment variables in Unix

- export DB2INSTANCE=db2
- echo $DB2INSTANCE
- Shell is case sensitive about variable names

- Works in shell scripts and on command line
- Might need curly brackets to delimit the variable name:
  - echo ${LOGDIR}/mylogfile.txt
- Example:
  - for i in 1 2 3 ; do echo ${i} ; done

dbdo.cmd to execute *.db2 files with command interpreter

```
@echo off
if "%~x1" == ".db2" goto good
echo Please use this batch file with db2 commands in a .db2 file as parameter (termination char is ';').
goto leave

:good
for /f "tokens=2" %%i in ('date /t') do set today=%%i
for /f "tokens=1,2,3 delims=/" %%i in ("%today%") do set today=%%k%%j%%i
for /f "tokens=1" %%i in ('time /t') do set now=%%i
for /f "tokens=1,2 delims=:" %%i in ("%now%") do set now=%%i%%j

db2 -tvf %1 -z %~n1_%today%_%now%.out

:leave
```

Shell command line expansion

- Be careful with your shell command line expansion!
  If you do not quote/escape properly you will get strange results.
  - db2 "SELECT * FROM employee"
  - db2 SELECT * FROM employee
- An * in Unix gets replaced by all filenames in the current directory if it isn't quoted or escaped.
- A tilde '~' will be replaced by your user home directory
- Alias is just plain text replacement without parameters
  - alias ll='ls –alF'

Quoting in the command line

- Some characters are interpreted in the shell and may need quotes
  - Watch out with *, ?, %, &, |, <, >, \, ", !, ^, =, ~, $, , , {, }, (,), ', `
- DB2 uses single quotes for Text ' '
  - db2 UPDATE employee SET lastname='Smith' where ID='00210'
- Shell uses double quotes " ", but can also use single quotes
  - db2 'select count(*) from employee'
- Brackets have different meaning in a Unix shell:
  - tar cf - * | (mkdir /tmp/demo ; cd /tmp/demo ; tar xvf -)

Quoting in the command line

- Nesting can be tricky! Here we add quotes around our reorg command:
  - Windows:
    db2 "select 'db2 ""reorg table '||rtrim(tabschema)||'.'||rtrim(tabname)||' "" ' from syscat.tables where stats_time < current timestamp - 5 days"
  - Unix:
    db2 "select 'db2 " ' " ' "reorg table '||rtrim(tabschema)||'.'||rtrim(tabname)||' " ' " ' " ' from syscat.tables where stats_time < current timestamp - 5 days"
  - Or if you know the ASCII codes… ;-)
    db2 "select 'db2 '||chr(34)||'reorg table '||rtrim(tabschema)||'.'||rtrim(tabname)||chr(34) from syscat.tables where stats_time < current timestamp - 5 days"
- All of the above commands can be directly used in your shell or in a script

Calling batch files

- Starting a batch file from a command line will pass an environment to the batch
  - Windows
    - All environment variables are passed on
    - The batch process can re-use the database connection
    - Changes to variables remain. Otherwise use setlocal/endlocal
  - Unix
    - Exported variables are passed on. Variables defined with 'set' are left out.
    - The batch file is started as a sub-process (fork) and needs it's own connection to DB2
    - Remedy: Use sourcing and prepend the command with '. ' (dot and space) to let the batch file run in the same shell instead of forking a new shell interpreter for the batch.

Calling batch files (cont.)

- Batch files can get input parameters from the command line
  - Windows use them as %1, %2, etc.
  - Windows Shell Extensions can use e.g. %~x1 to split filenames in parameters
  - Unix gets $1, $2, etc.; $* stands for all parameters
    $? Is the return code of the previous command

Pipes and I/O redirections

- Concatenation of commands
- Sending output from one command to some other tool via pipes
  `db2 list applications | more`
- Multiple pipes
  - All commands in the pipe can run at the same time
  - `cat LOAD*.msg | grep "^DB2" | sort | uniq -c // check for DB2 msgs in loads`
- Conditional execution
  - `db2start && db2 connect to SAMPLE`
- Using return codes for error handling

Standard streams

- Standard streams stdin, stdout und stderr
- Well known by C/C++-Programmers ;-)
- The shell and command tools takes input from stdin (stream 0) and sends output to stdout (stream 1) and error messages to stderr (stream 2) .
  - Combining stderr in stdout
    - To redirect both stdout and stderr to the same file, use 2>&1
    - E.g. useful for cronjobs, which send the output as mail
  - stderr exists for clever programmers, who want to show error messages even if the output is sent to a file

- Part 3 – GNU Utils

GNU Core Utilities – General Introduction

- What they are?
  - Gnu's Not Unix, is the name for a complete Unix-compatible software system, which is free
  - Richard Stallman is the Guru/Father of the GNU Project.
- Where did they come from?
  - By the 1980s, almost all software was proprietary. This made the GNU Project necessary.
  - By 1990 they either found or written all the major components except one—the kernel. Then Linux, a Unix-like kernel, was developed by Linus Torvalds in 1991 and made free software in 1992.
- GNU Core Utilities, available via http://www.gnu.org/software/coreutils/

In 1971, when Richard Stallman started his career at MIT, he worked in a group which used free software exclusively. Even computer companies often distributed free software. Programmers were free to cooperate with each other, and often did.

GNU Text Utilities

- Overview of GNU Core Utilities
  - Utilities that are valuable when trying to match, replace and advance line processing (grep, sed, awk, …)
- Which commands are useful
  - cat, head, tail, more|less, cut
  - wc, uniq, sort
  - fgrep, grep
  - diff, patch
  - sed, awk, xargs
  - Less used tools
    - split, csplit, reverse

GNU Text Utilities

- cat: Output a text file to console
- head, tail: Output the first or last n lines from a file
- more|less: paginate a file on screen (needs user input)
- cut: Returns parts of a text line (by char or field)
- wc: Count charachters, words or lines
- uniq: Remove duplicate lines (may need a sort first ;-)
- sort: Sort text files line by line
- grep, fgrep: Search for (regular) expressions in text

GNU Text Utilities

- diff: Find differences in two text files
- patch: Insert changes in text files generated by diff
- sed: Stream-Editor, change/replace text in streams
- awk: Programming language for pattern matching and execution, now superseeded by perl
- xargs: builds and executes commands from standard input

- Less used tools
  - split, csplit, reverse, …

- Part 4 – Samples

Utilities and db2 commands

- Samples of all things you can play around and customize to your needs
  - Quick way find out what DDL was executed on your DB
    - db2 list history all for db MYPRODDB | grep DDL
  - Check if your instance is running
    - ps -ef | grep $DB2INSTANCE | grep db2sysc | grep -v grep | wc –l
    - db2pd -agents | grep -e Partition -e Unable
  - Quick and dirty number of connections to all databases in instance:
    - db2 list applications | cut -c 99-106 | sort | uniq –c

Utilities and db2 commands (cont.)

- Find the busiest tables via command line or ksh
  - db2pd -db dbname -tcbstats| awk '/TCB Table Stats/ { x =1} x==1 { print}' | awk '/^0x/ { print $9, $2}' | sort –rn | head -15
- Get reorg recommendations and change them to inplace reorg instead of offline reorg
  - db2 get recommendations for health indicator db.tb_reorg_req for database on d3t | grep "^REORG" |awk ' /REORG TABLE /{gsub(/;/, " inplace ;")}; 1'|awk '{gsub(/\"RUNSTATS/,";\nRUNSTATS"); print}' | tee -a reorg.sql

Utilities and db2 commands (cont.)

- Generate Skript to re-catalog your databases
  - db2 list db directory | \
    egrep "Database alias|Node name" | \
    sed -e 's/Database alias//g;s/Node name//g;s/ //g;s/=//g' | \
    sed 'N;s/\n/ : /' | \
    awk '{print "db2 catalog db "$1" at node "$3;}'

Use SQL to generate stuff

- Extract statements from package cache as input for db2advis
  - db2 -x "select '--#SET FREQUENCY '||ltrim(char(num_executions))||'@@'||rtrim(STMT_TEXT)||';' from table(snapshot_dyn_sql('PK2I11C1', -1)) as SNAP_STMT where stmt_text is not null and num_executions > 100 order by num_executions desc" | fgrep -iv "SYSIBM" | sed „s/ *$//“ >MostCommonQueries.txt
  - Now replace @@ by CRLF and feed into db2advis or db2batch
- Checking integrity after foreign keys if needed
  - db2 -xtd! "select 'SET INTEGRITY FOR '||chr(34)||rtrim(tabschema)||chr(34)||'.'||chr(34)||rtrim(tabname)||chr(34)||' IMMEDIATE CHECKED FORCE GENERATED' from syscat.tables where tabschema not in ('SYSIBM','SYSCAT','SYSSTAT') and STATUS='C' order by parents,children!" >checkref.sql
    db2 -v -f checkref.sql
    echo Checking integrity for constraints done.

KEY-BASED AUTHENTICATION USING SSH

- So why would you want to set up key based authentication using ssh?
- Key based authentication is situation of authentication takes place based on the public/private information, rather than with the more usual method of prompting for a password. This is very convenient if a non-interactive process is trying to authenticate with a remote machine
- SSH is the preferred way of setting up such connections now as it uses public and private key pairs to establish a secure connection.
- It allows you securely to start commands on another machine without providing passwords!

- The keys and encryption methods make it extremely difficult for IP spoofers to generate a connection and since it is secure, you do not need to worry about data being sniffed.

Key-based authentication for SSH

- Arranging key-based authentication for SSH
  - Machine "A" is the machine you want to connect from
  - Machine "B" is the machine you want to connect to
- Setup – Part 1:
  - On Machine "A", generate your key pair using ssh-keygen -t dsa
  - This will create two files:
    - ~/.ssh/id_dsa
    - ~/.ssh/id_dsa.pub

Key-based authentication for SSH (cont.)

- Setup – Part 2:
  - Log onto Machine "B"
    - Secure copy the id_dsa.pub key across to the local machine
    - Now look at the file /~.ssh/id_dsa.pub, if it starts with "ssh-dss", ie
      - ssh-dss AAAAB3NzaC1kc3MAAAC+CLO2M9OfcIjEaFBJ+cNAubJeCw8dtlH
    - Then append it to the file ./ssh/authorized_keys2 and ./ssh2/authorized_keys2
      - `cat id_dsa.pub >> ~/.ssh/authorized_keys2`
      - `cat id_dsa.pub >> ~/.ssh2/authorized_keys2`
      - change permissions on files to user read/write only
  - Test – now you will be able to ssh from Machine A to Machine B (no password req.)

Backup/Restore via pipe

- Pipes are special file objects under Unix operating systems
  - Create with mkfifo
  - Can be used for a quick backup/restore from one machine to another or between instances
- Machine 1:

```
mkfifo /tmp/mydest
```

Machine 2:

```
mkfifo /tmp/mysource

db2 restore db test_1 as test_new
from /tmp/mysource replace existing
without prompting
```

```
rsh host2 'cat > /tmp/mysource' < /tmp/mydest &

db2 backup db test_1 to /tmp/mydest
```

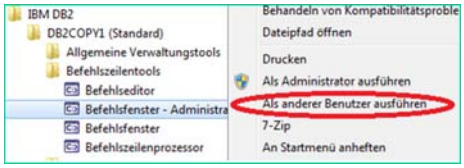
Windows 7 User Access Control

- With User Access Control in Windows 7 or Windows Server 2008, you have to confirm the execution of administrative commands.

- To start the instance via db2start you need admin rights or you have to be instance owner. If the instance owner is your domain user, it is getting tricky, if you are on the road with your laptop. Runas doesn't help here!

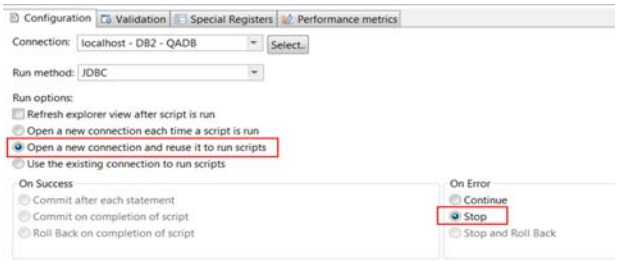
Windows 7 User Access Control (cont.)

- If you use your domain account for DB2 installation and administration, you get admin rights only as long as your laptop is in the domain.
  - You should better use a local user for install and administration
  - db2set DB2_GRP_LOOKUP and SYSADM_GROUP could help
- Much easier is following trick:
  - Start → IBM DB2 → Commandline tools → Command window → Shift+Right-Click on the needed entry and login with MachineName\InstanceOwner (if the instance owner is a local userid)



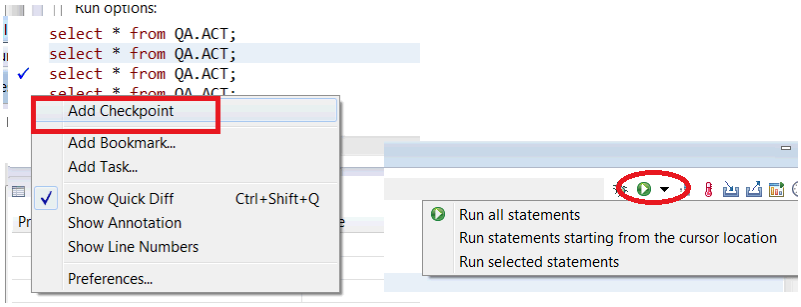
Scripts on Data Studio?

- Scripts on Data Studio just for DB2 scripts via JDBC
- Data Studio has some cool new enhancements, checkpoint support in the script editor.
  - It allows the user to evaluate the script execution to that point and make decisions on how to proceed.



Checkpoints on scripts

- To set a checkpoint, right click at the space right before the desired line and select "Add Checkpoint".



- When the script is executing, it will stop before executing any statement with a checkpoint and keep the transaction open.
- You can alter the script, run one or more statements or restart the script from any point.
- Remember, transactions stays open until committed or the script editor is closed.
- This can result in locks being held for a long time.
- Also, remember to commit your work before closing the editor.
- Any uncommitted statements will be rolled back when the window is closed or Data Studio is shut down.

Roland Schock
ARS Computer und Consulting GmbH
schock@ars.de

Session Code: F13
DB2 Scripting Hacks

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988