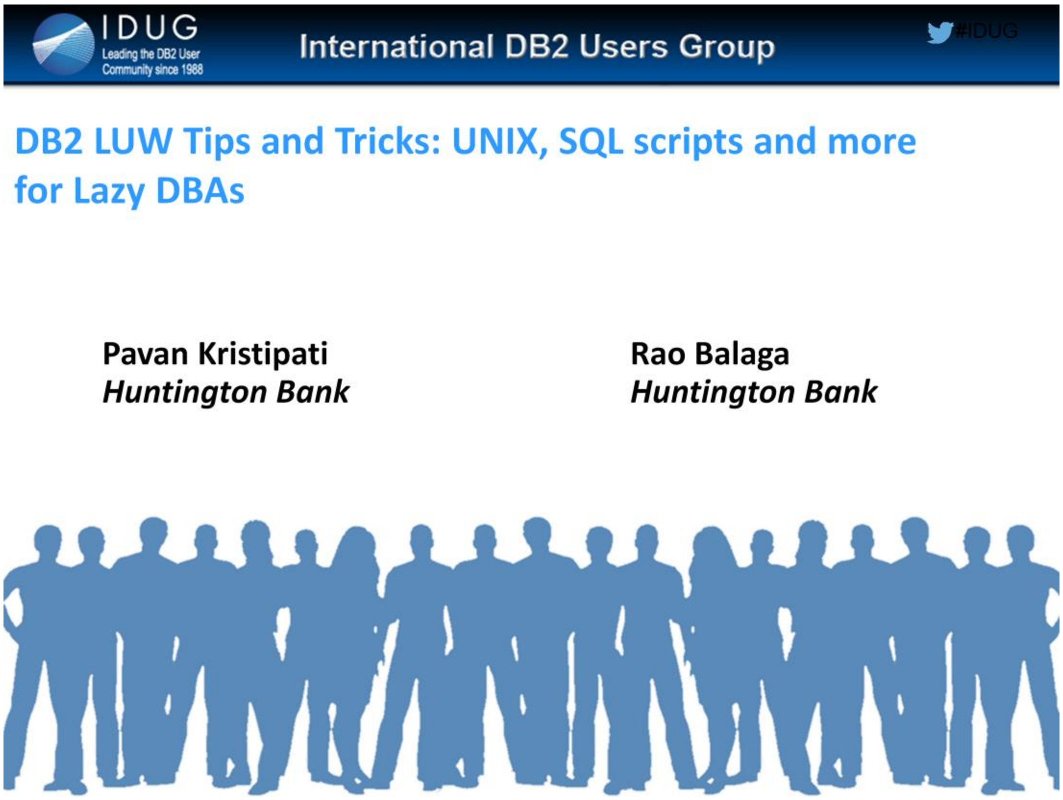

International DB2 Users Group

IDUG Leading the DB2 User Community since 1988

#IDUG

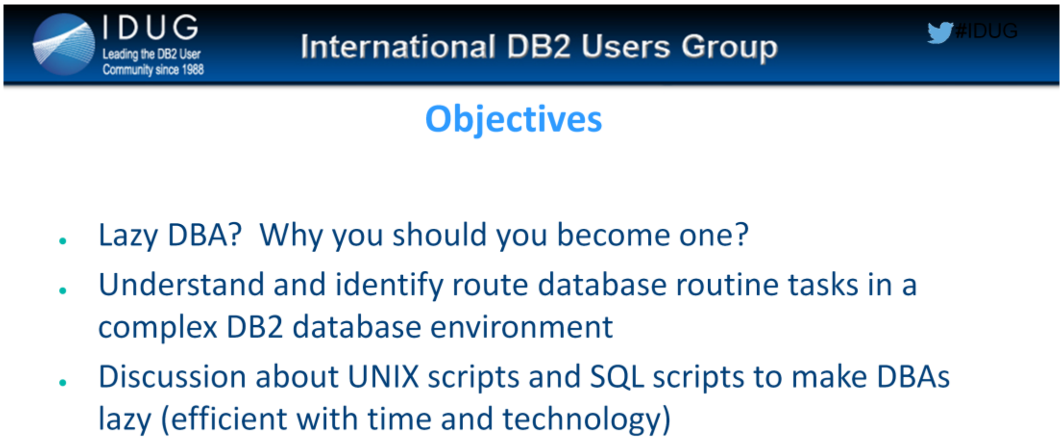
DB2 LUW Tips and Tricks: UNIX, SQL scripts and more for Lazy DBAs

Pavan Kristipati
Huntington Bank

Rao Balaga
Huntington Bank

Welcome to the presentation.
Thank you for taking your time for being here.

I hope that most of you would get something useful out of this presentation.
We will have time at the end for questions.


IDUG Leading the DB2 User Community since 1988

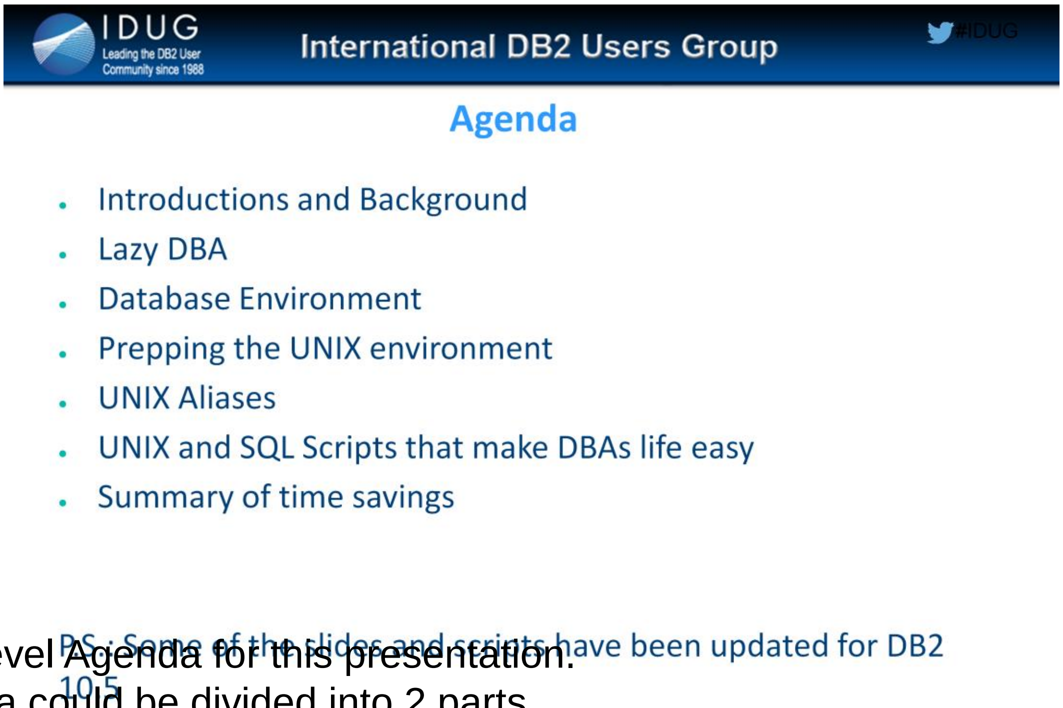
International DB2 Users Group

#IDUG

Objectives

- Lazy DBA? Why you should you become one?
- Understand and identify route database routine tasks in a complex DB2 database environment
- Discussion about UNIX scripts and SQL scripts to make DBAs lazy (efficient with time and technology)

Objectives of the presentation that were included when abstract was submitted to IDUG for consideration.


IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Agenda

- Introductions and Background
- Lazy DBA
- Database Environment
- Prepping the UNIX environment
- UNIX Aliases
- UNIX and SQL Scripts that make DBAs life easy
- Summary of time savings

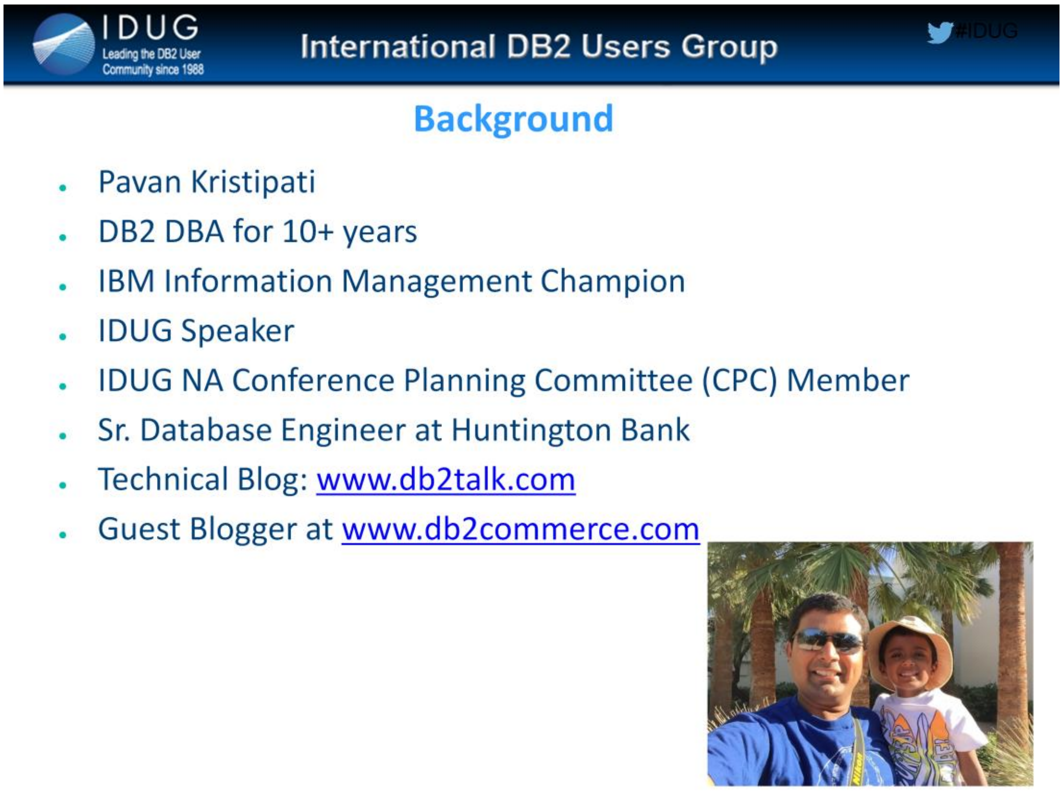
PS : Some of the slides and scripts have been updated for DB2 10.5

High level Agenda for this presentation.
Agenda could be divided into 2 parts.

In Part 1, we will introduce who a lazy DBA is and if you should be one? Later, we cover prepping the database environment to implement methodologies/scripts that are covered in step 2.
In Part 2, we will share UNIX and SQL tips and tricks that increase productivity of DBAs and help save time (be efficient)

** in the slide indicates the following:
1) We did not include major error handling in UNIX and SQL scripts
2) Chances are there might be no standards that were followed when scripts were written as they were written purely to make DBAs life easy
3) Scripts could be re-written to make them better

Our goal is to share what we have done and encourage other DBAs to implement similar thought process in their daily work life.



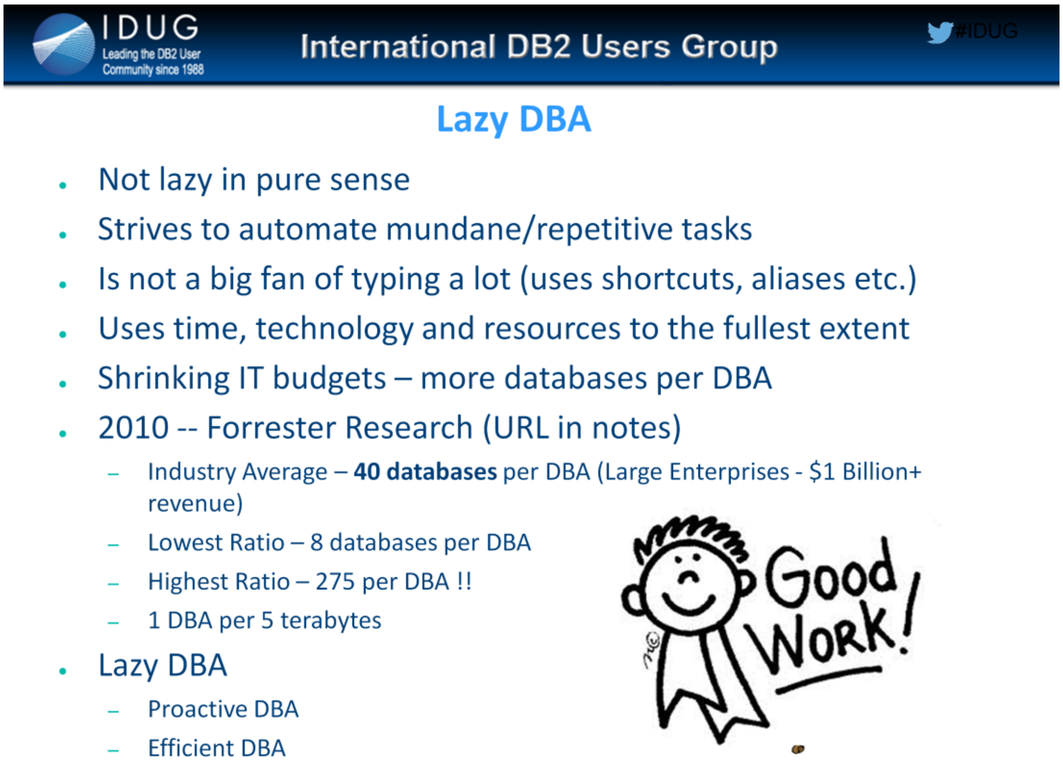
International DB2 Users Group

#IDUG

Background

- Pavan Kristipati
- DB2 DBA for 10+ years
- IBM Information Management Champion
- IDUG Speaker
- IDUG NA Conference Planning Committee (CPC) Member
- Sr. Database Engineer at Huntington Bank
- Technical Blog: www.db2talk.com
- Guest Blogger at www.db2commerce.com

Objectives of the presentation that were included when abstract was submitted to IDUG for consideration.



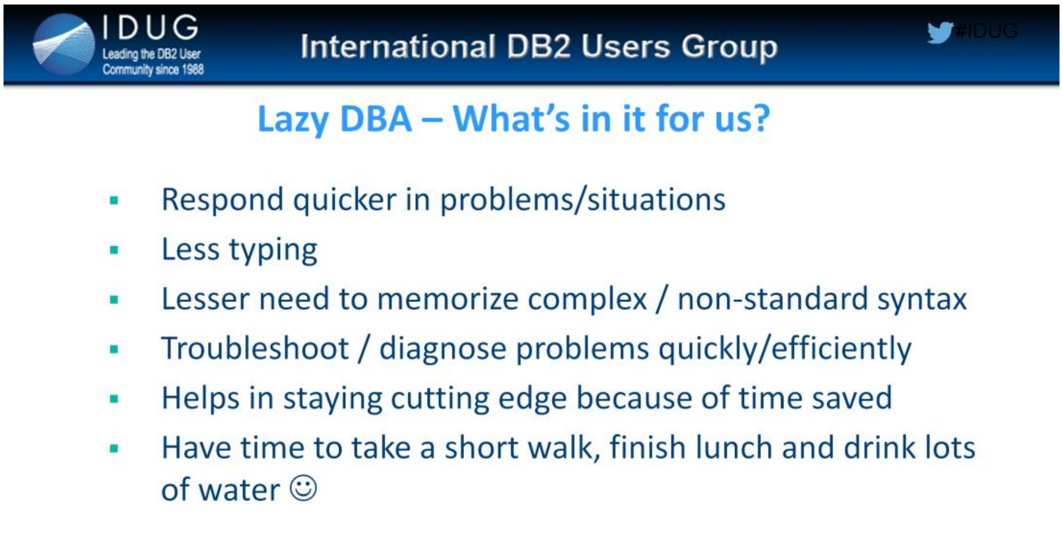
IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Lazy DBA

- Not lazy in pure sense
- Strives to automate mundane/repetitive tasks
- Is not a big fan of typing a lot (uses shortcuts, aliases etc.)
- Uses time, technology and resources to the fullest extent
- Shrinking IT budgets – more databases per DBA
- 2010 -- Forrester Research (URL in notes)
  - Industry Average – **40 databases** per DBA (Large Enterprises - $1 Billion+ revenue)
  - Lowest Ratio – 8 databases per DBA
  - Highest Ratio – 275 per DBA !!
  - 1 DBA per 5 terabytes
- Lazy DBA
  - Proactive DBA
  - Efficient DBA

A Lazy DBA is not lazy in pure sense.
He/she is one who strives to automate most of the mundane tasks to be able to work on more exciting things.
He/she strives to save time by finding simpler ways of doing things that we need on a daily basis.

URL for Forrester Research:
http://blogs.forrester.com/noel_yuhanna/10-09-30-how_many_dbas_do_you_need_support_databases



IDUG Leading the DB2 User Community since 1988

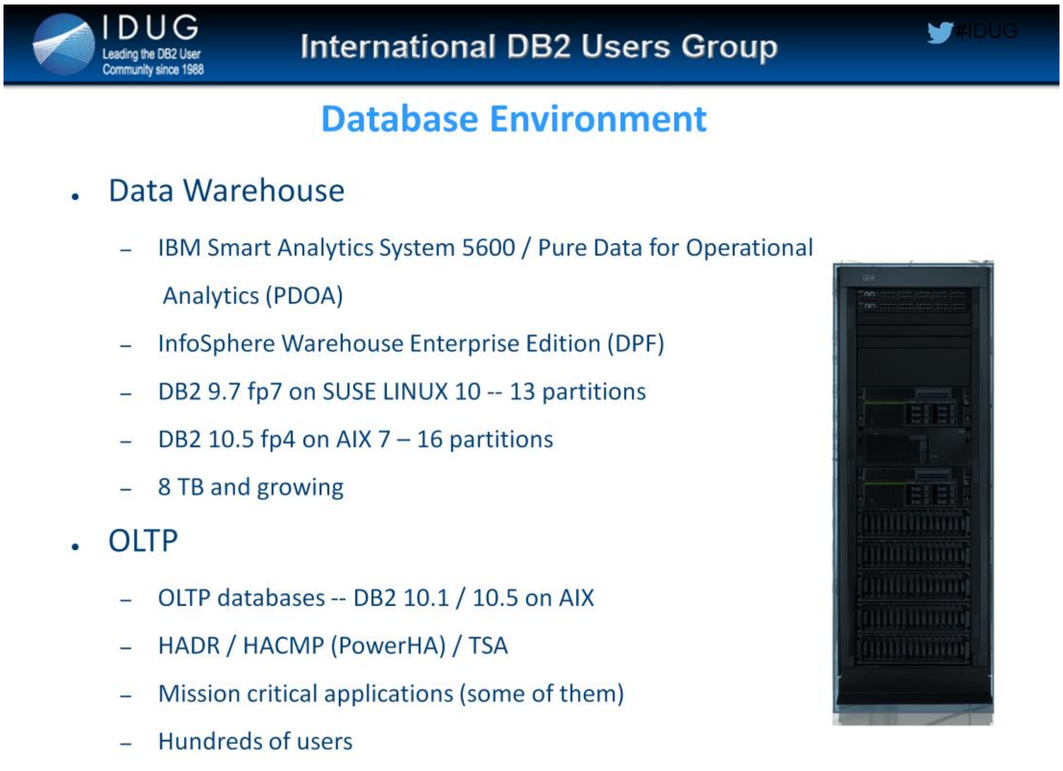
International DB2 Users Group

#IDUG

Lazy DBA – What's in it for us?

- Respond quicker in problems/situations
- Less typing
- Lesser need to memorize complex / non-standard syntax
- Troubleshoot / diagnose problems quickly/efficiently
- Helps in staying cutting edge because of time saved
- Have time to take a short walk, finish lunch and drink lots of water ☺

Becoming a lazy DBA helps in staying cutting edge because of the time that is saved by automating tasks.



IDUG Leading the DB2 User Community since 1988

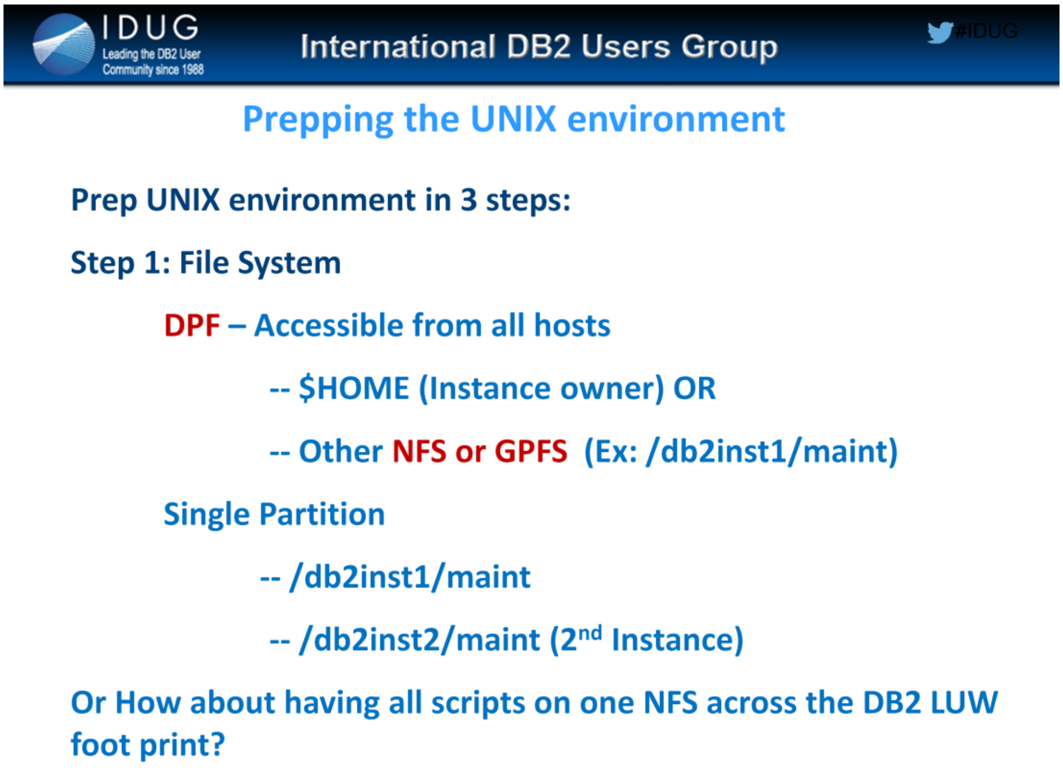
International DB2 Users Group

#IDUG

Database Environment

- Data Warehouse
  - IBM Smart Analytics System 5600 / Pure Data for Operational Analytics (PDOA)
  - InfoSphere Warehouse Enterprise Edition (DPF)
  - DB2 9.7 fp7 on SUSE LINUX 10 -- 13 partitions
  - DB2 10.5 fp4 on AIX 7 – 16 partitions
  - 8 TB and growing
- OLTP
  - OLTP databases -- DB2 10.1 / 10.5 on AIX
  - HADR / HACMP (PowerHA) / TSA
  - Mission critical applications (some of them)
  - Hundreds of users

Overview of DB2 LUW databases at Huntington Bank.


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Prepping the UNIX environment

Prep UNIX environment in 3 steps:

Step 1: File System

DPF – Accessible from all hosts

-- $HOME (Instance owner) OR

-- Other NFS or GPFS (Ex: /db2inst1/maint)

Single Partition

-- /db2inst1/maint

-- /db2inst2/maint (2nd Instance)

Or How about having all scripts on one NFS across the DB2 LUW foot print?

From a high level, we have ‘custom’ and ‘maint’ sub-directories on either a dedicated file system or file system that has Instance Owner’s home directory in DPF database.

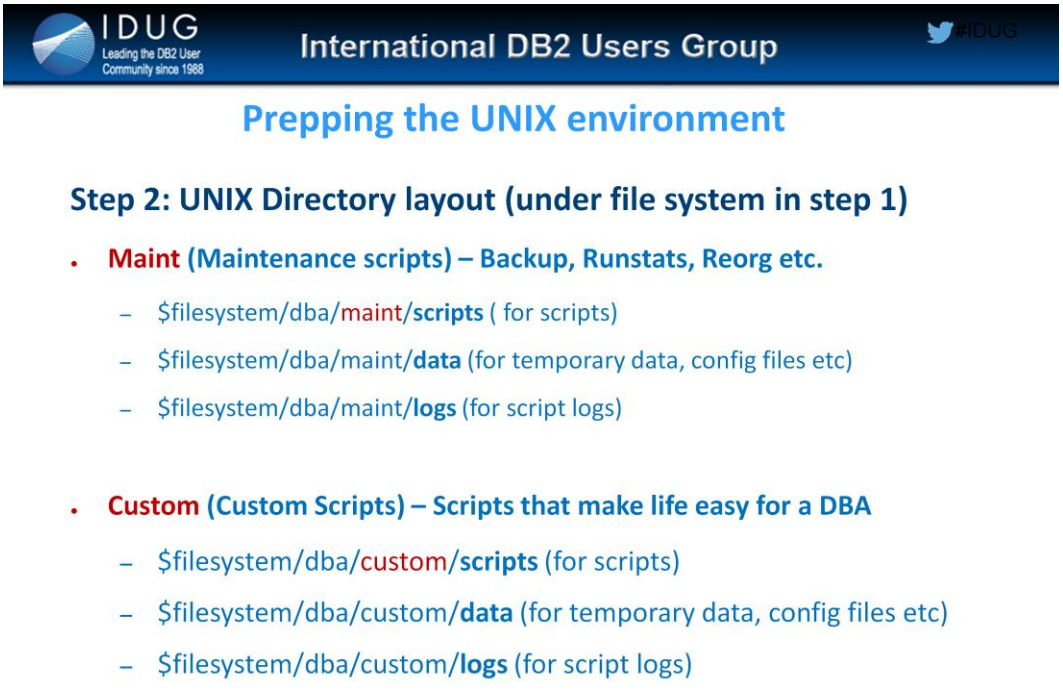
Each of these directories in-turn have:

- scripts
- data
- logs

as sub-directories.

/custom/scripts directory has in it all scripts that make DBAs life easy. Some of them are shared in next slides.

/maint/scripts directory has in it all scripts that are for DB2 maintenance purposes.. Example: runstats, reorg, backup etc.


IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Prepping the UNIX environment

Step 2: UNIX Directory layout (under file system in step 1)

- Maint (Maintenance scripts) – Backup, Runstats, Reorg etc.
  - $filesystem/dba/maint/**scripts** (for scripts)
  - $filesystem/dba/maint/**data** (for temporary data, config files etc)
  - $filesystem/dba/maint/**logs** (for script logs)

- Custom (Custom Scripts) – Scripts that make life easy for a DBA
  - $filesystem/dba/custom/**scripts** (for scripts)
  - $filesystem/dba/custom/**data** (for temporary data, config files etc)
  - $filesystem/dba/custom/**logs** (for script logs)

From a high level, we have ‘custom’ and ‘maint’ sub-directories on either a dedicated file system or file system that has Instance Owner’s home directory in DPF database.

Each of these directories in-turn have:

- scripts
- data
- logs

as sub-directories.

/custom/scripts directory has in it all scripts that make DBAs life easy. Some of them are shared in next slides.

/maint/scripts directory has in it all scripts that are for DB2 maintenance purposes.. Example: runstats, reorg, backup etc.

Prepping the UNIX environment

Step 3: Add directories (created in step 2) to $PATH

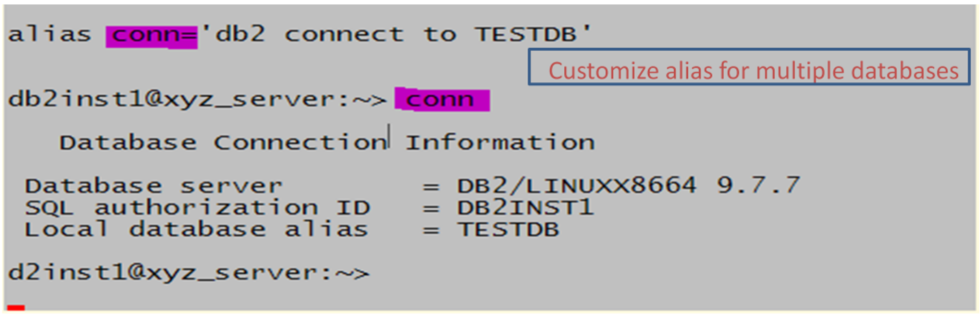
In $HOME/.profile of instance owner, add 2 lines:

- PATH=$PATH\:$HOME/dba/custom/scripts ; export PATH
- PATH=$PATH\:$HOME/dba/maint/scripts ; export PATH

Step 3 is done to be able to find / execute scripts from any path

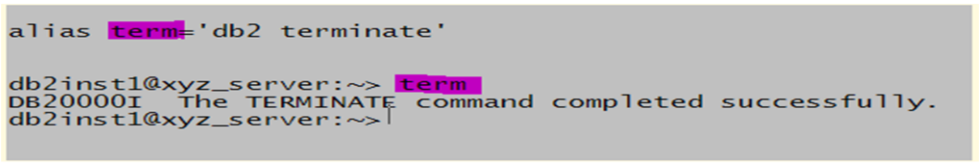
UNIX ALIASES

UNIX Aliases -- Save time and reduce typing - 1



```
alias conn='db2 connect to TESTDB'

db2inst1@xyz_server:~> conn

   Database Connection Information

 Database server        = DB2/LINUXX8664 9.7.7
 SQL authorization ID   = DB2INST1
 Local database alias   = TESTDB

d2inst1@xyz_server:~>
```

Customize alias for multiple databases

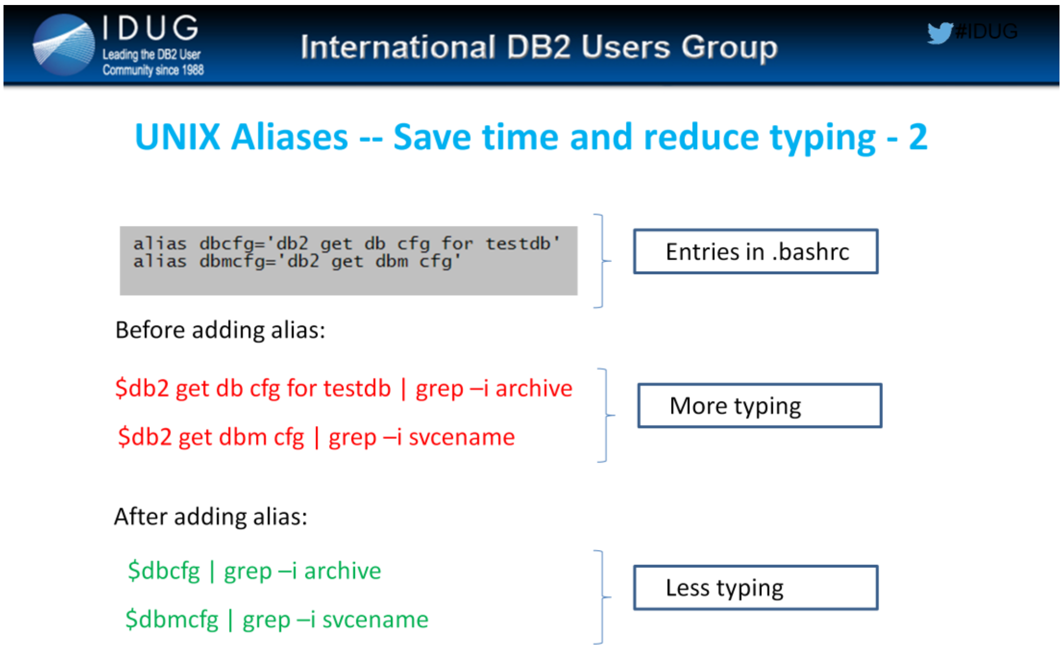


```
alias term='db2 terminate'

db2inst1@xyz_server:~> term
DB20000I  The TERMINATE command completed successfully.
db2inst1@xyz_server:~>
```

Adding an alias is a one-time task. Using multiple aliases instead of typing long commands saves time.

Next set of slides have aliases for mundane and repetitious commands that DBAs use on a regular basis.

After adding alias, you have to either sign out and sign in again or run $HOME/.bashrc (.kshrc in case of korn shell) in bash shell.


International DB2 Users Group

UNIX Aliases -- Save time and reduce typing - 2

```
alias dbcfg='db2 get db cfg for testdb'
alias dbmcfg='db2 get dbm cfg'
```

Entries in .bashrc

Before adding alias:

$db2 get db cfg for testdb | grep –i archive

$db2 get dbm cfg | grep –i svcename

More typing

After adding alias:

$dbcfg | grep –i archive

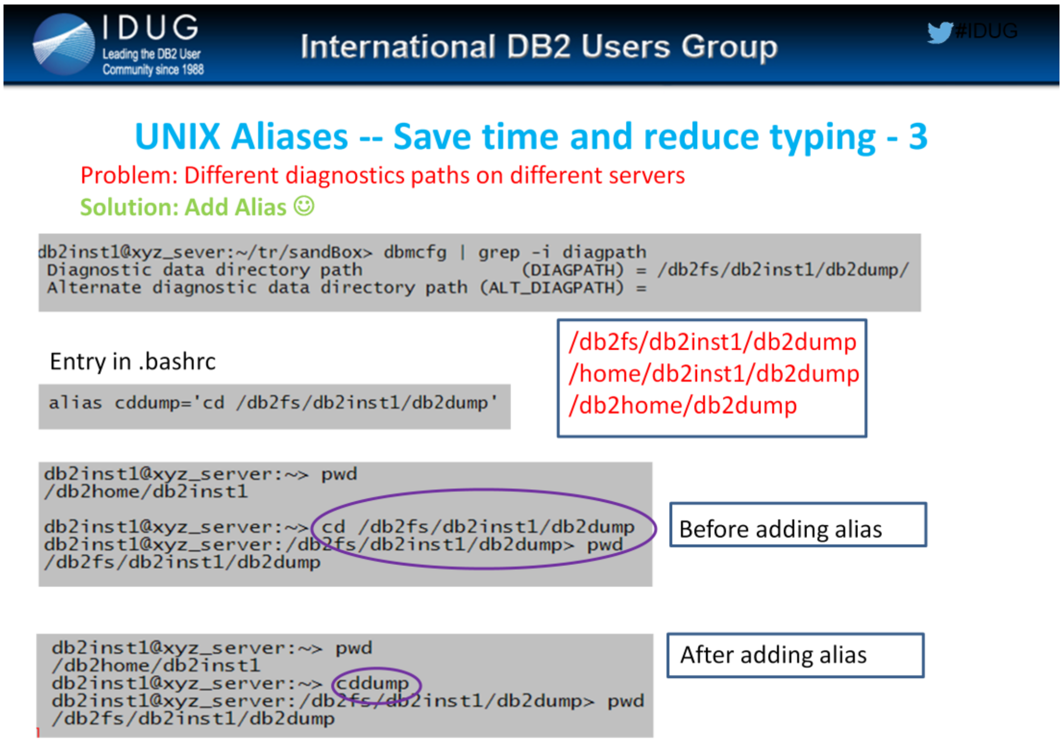
$dbmcfg | grep –i svcename

Less typing

Here is an example of how more aliases could be set up for:

1) Database Configuration file (db cfg)
2) Database Manager Configuration file (dbm cfg)

Instead of typing “db2 get db cfg for TESTDB” one could get away with typing “dbcfg”. Of course, if you have multiple databases, one needs to create a unique alias for each of them if needed.


IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

UNIX Aliases -- Save time and reduce typing - 3

Problem: Different diagnostics paths on different servers

Solution: Add Alias ☺

```
db2inst1@xyz_sever:~/tr/sandBox> dbmcfg | grep -i diagpath
 Diagnostic data directory path               (DIAGPATH) = /db2fs/db2inst1/db2dump/
 Alternate diagnostic data directory path (ALT_DIAGPATH) =
```

/db2fs/db2inst1/db2dump
/home/db2inst1/db2dump
/db2home/db2dump

Entry in .bashrc

```
alias cddump='cd /db2fs/db2inst1/db2dump'
```

Before adding alias

```
db2inst1@xyz_server:~> pwd
/db2home/db2inst1

db2inst1@xyz_server:~> cd /db2fs/db2inst1/db2dump
db2inst1@xyz_server:/db2fs/db2inst1/db2dump> pwd
/db2fs/db2inst1/db2dump
```

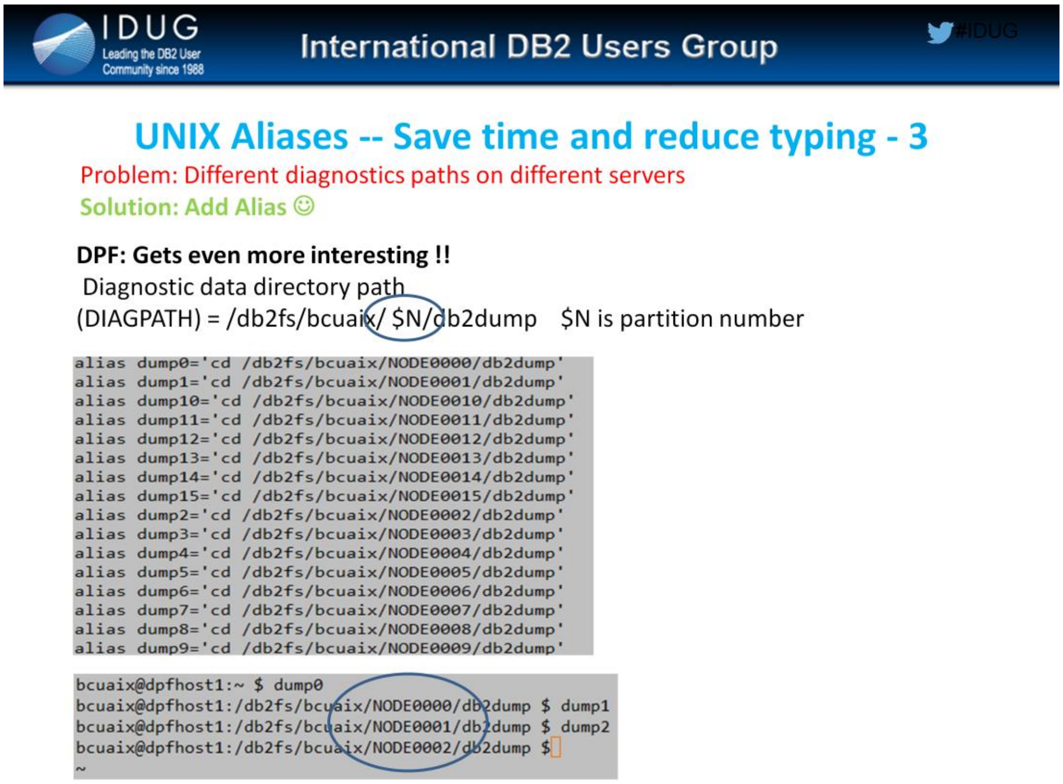
After adding alias

```
db2inst1@xyz_server:~> pwd
/db2home/db2inst1
db2inst1@xyz_server:~> cddump
db2inst1@xyz_server:/db2fs/db2inst1/db2dump> pwd
/db2fs/db2inst1/db2dump
```

Here is an example of how we use more aliases for:

1) Changing to (cd) Diagnostics directory

Instead of typing diagpath, (cd /db2fs/db2inst1/db2dump), we could get away with typing “cddump”.


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

UNIX Aliases -- Save time and reduce typing - 3

Problem: Different diagnostics paths on different servers

Solution: Add Alias ☺

DPF: Gets even more interesting !!

Diagnostic data directory path

(DIAGPATH) = /db2fs/bcuaix/ $N/db2dump $N is partition number

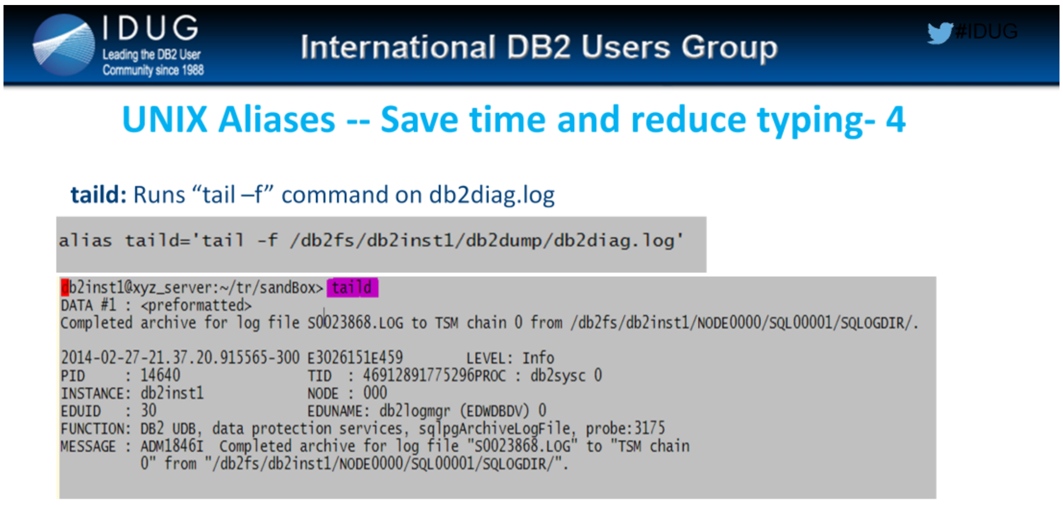
```
alias dump0='cd /db2fs/bcuaix/NODE0000/db2dump'
alias dump1='cd /db2fs/bcuaix/NODE0001/db2dump'
alias dump10='cd /db2fs/bcuaix/NODE0010/db2dump'
alias dump11='cd /db2fs/bcuaix/NODE0011/db2dump'
alias dump12='cd /db2fs/bcuaix/NODE0012/db2dump'
alias dump13='cd /db2fs/bcuaix/NODE0013/db2dump'
alias dump14='cd /db2fs/bcuaix/NODE0014/db2dump'
alias dump15='cd /db2fs/bcuaix/NODE0015/db2dump'
alias dump2='cd /db2fs/bcuaix/NODE0002/db2dump'
alias dump3='cd /db2fs/bcuaix/NODE0003/db2dump'
alias dump4='cd /db2fs/bcuaix/NODE0004/db2dump'
alias dump5='cd /db2fs/bcuaix/NODE0005/db2dump'
alias dump6='cd /db2fs/bcuaix/NODE0006/db2dump'
alias dump7='cd /db2fs/bcuaix/NODE0007/db2dump'
alias dump8='cd /db2fs/bcuaix/NODE0008/db2dump'
alias dump9='cd /db2fs/bcuaix/NODE0009/db2dump'
```

```
bcuaix@dpfhost1:~ $ dump0
bcuaix@dpfhost1:/db2fs/bcuaix/NODE0000/db2dump $ dump1
bcuaix@dpfhost1:/db2fs/bcuaix/NODE0001/db2dump $ dump2
bcuaix@dpfhost1:/db2fs/bcuaix/NODE0002/db2dump $
~
```

Here is an example of how we use more aliases for:

1) Changing to (cd) Diagnostics directory

Instead of typing diagpath, (cd /db2fs/db2inst1/db2dump), we could get away with typing “cddump”.



UNIX Aliases -- Save time and reduce typing- 4

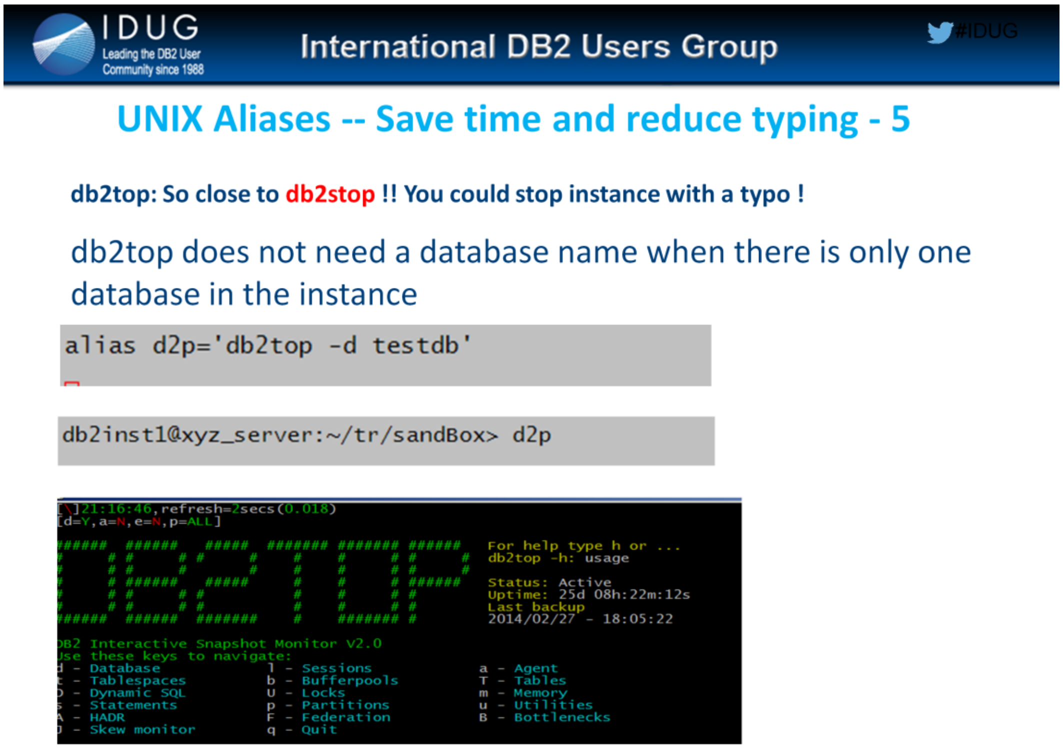
taild: Runs “tail –f” command on db2diag.log

```
alias taild='tail -f /db2fs/db2inst1/db2dump/db2diag.log'
```

```
db2inst1@xyz_server:~/tr/sandBox> taild
DATA #1 : <preformatted>
Completed archive for log file S0023868.LOG to TSM chain 0 from /db2fs/db2inst1/NODE0000/SQL00001/SQLOGDIR/.

2014-02-27-21.37.20.915565-300 E3026151E459         LEVEL: Info
PID     : 14640                TID  : 46912891775296PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID   : 30                   EDUNAME: db2logmgr (EDWDBDV) 0
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3175
MESSAGE : ADM1846I  Completed archive for log file "S0023868.LOG" to "TSM chain
          0" from "/db2fs/db2inst1/NODE0000/SQL00001/SQLOGDIR/".
```

Many a times, as DBA, we have to take a look at latest entries that are being added to db2diag.log file.

A simple alias shown in this slide would let us ‘tail’ on this file with ease. Simply type ‘taild’


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

UNIX Aliases -- Save time and reduce typing - 5

db2top: So close to db2stop !! You could stop instance with a typo !

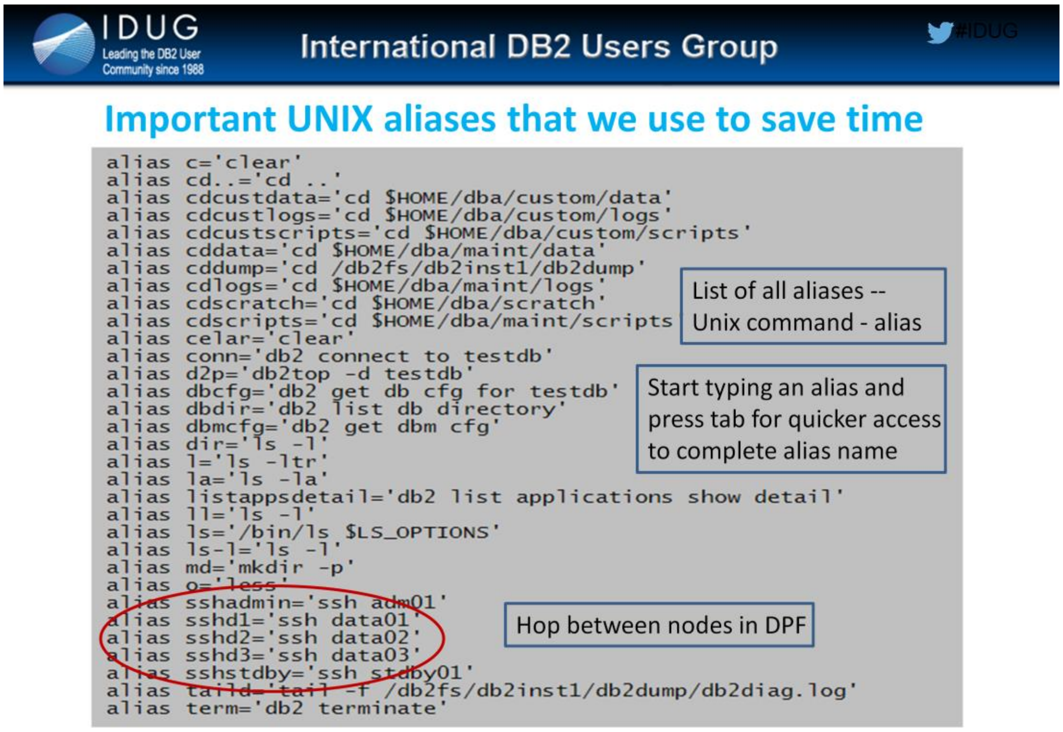
db2top does not need a database name when there is only one database in the instance

```
alias d2p='db2top -d testdb'
```

```
db2inst1@xyz_server:~/tr/sandBox> d2p
```

```
[\]21:16:46,refresh=2secs(0.018)
[d=Y,a=N,e=N,p=ALL]

######  ######   #####  ####### ####### ######   For help type h or ...
#     # #     # #     #    #    #     # #     #  db2top -h: usage
#     # #     #       #    #    #     # #     #
#     # ######   #####     #    #     # ######   Status: Active
#     # #     # #          #    #     # #        Uptime: 25d 08h:22m:12s
#     # #     # #          #    #     # #        Last backup
######  ######  #######    #    ####### #        2014/02/27 - 18:05:22

DB2 Interactive Snapshot Monitor V2.0
Use these keys to navigate:
d - Database              l - Sessions              a - Agent
t - Tablespaces           b - Bufferpools           T - Tables
D - Dynamic SQL           U - Locks                 m - Memory
s - Statements            p - Partitions            u - Utilities
A - HADR                  F - Federation            B - Bottlenecks
J - Skew monitor          q - Quit
```

Shortcut to db2top


IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Important UNIX aliases that we use to save time

```
alias c='clear'
alias cd..='cd ..'
alias cdcustdata='cd $HOME/dba/custom/data'
alias cdcustlogs='cd $HOME/dba/custom/logs'
alias cdcustscripts='cd $HOME/dba/custom/scripts'
alias cddata='cd $HOME/dba/maint/data'
alias cddump='cd /db2fs/db2inst1/db2dump'
alias cdlogs='cd $HOME/dba/maint/logs'
alias cdscratch='cd $HOME/dba/scratch'
alias cdscripts='cd $HOME/dba/maint/scripts'
alias celar='clear'
alias conn='db2 connect to testdb'
alias d2p='db2top -d testdb'
alias dbcfg='db2 get db cfg for testdb'
alias dbdir='db2 list db directory'
alias dbmcfg='db2 get dbm cfg'
alias dir='ls -l'
alias l='ls -ltr'
alias la='ls -la'
alias listappsdetail='db2 list applications show detail'
alias ll='ls -l'
alias ls='/bin/ls $LS_OPTIONS'
alias ls-l='ls -l'
alias md='mkdir -p'
alias o='less'
alias sshadmin='ssh adm01'
alias sshd1='ssh data01'
alias sshd2='ssh data02'
alias sshd3='ssh data03'
alias sshstdby='ssh stdby01'
alias taild='tail -f /db2fs/db2inst1/db2dump/db2diag.log'
alias term='db2 terminate'
```

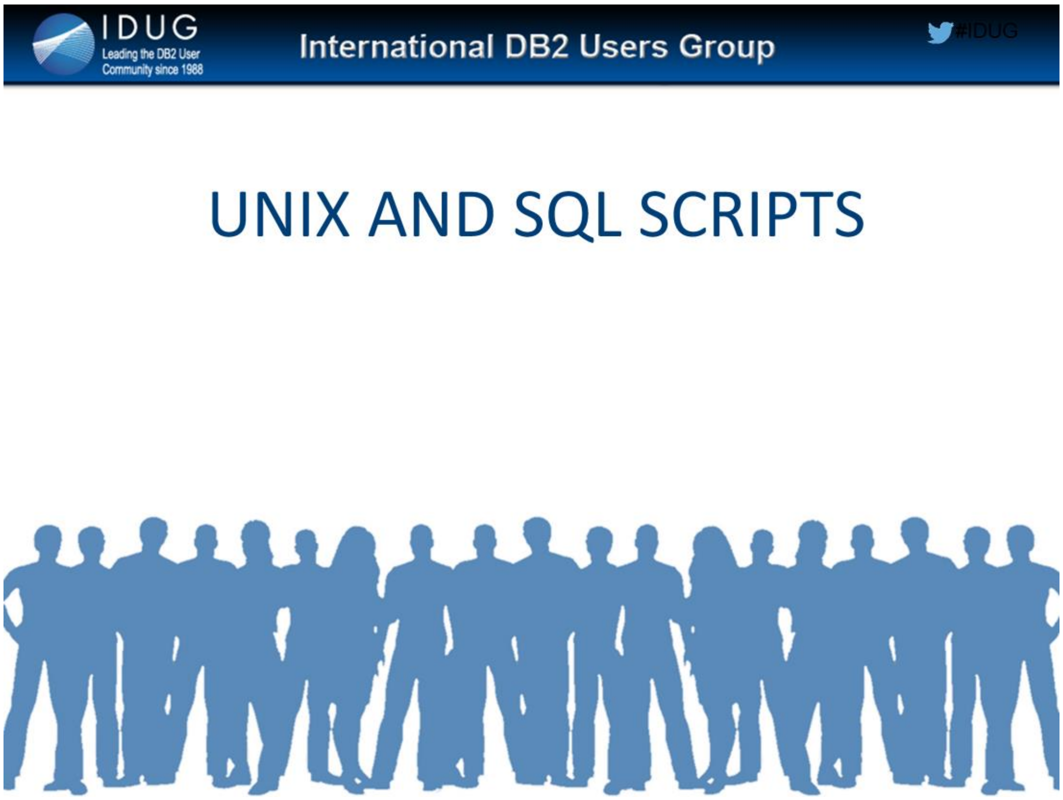
List of all aliases -- Unix command - alias

Start typing an alias and press tab for quicker access to complete alias name

Hop between nodes in DPF

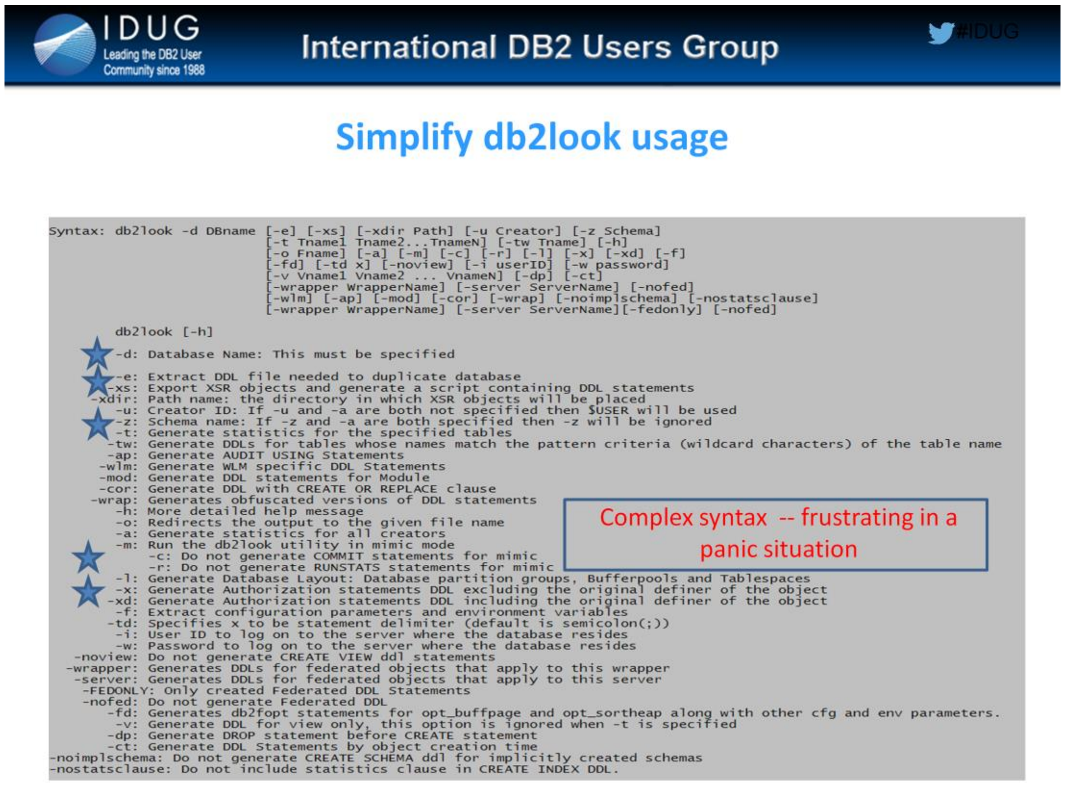
Here are some the aliases that DBAs at Huntington use.

It is common to ‘ssh’ from one host to another within the DPF db environment. Creating alias(es) for these ‘ssh’ commands would help in a big way saving typing time and avoid ‘typos’ in server names.


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

UNIX AND SQL SCRIPTS

High level agenda for the remainder of the presentation. We will have few minutes for questions at the end.



IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Simplify db2look usage

```
Syntax: db2look -d DBname [-e] [-xs] [-xdir Path] [-u Creator] [-z Schema]
                          [-t Tname1 Tname2...TnameN] [-tw Tname] [-h]
                          [-o Fname] [-a] [-m] [-c] [-r] [-l] [-x] [-xd] [-f]
                          [-fd] [-td x] [-noview] [-i userID] [-w password]
                          [-v Vname1 Vname2 ... VnameN] [-dp] [-ct]
                          [-wrapper WrapperName] [-server ServerName] [-nofed]
                          [-wlm] [-ap] [-mod] [-cor] [-wrap] [-noimplschema] [-nostatsclause]
                          [-wrapper WrapperName] [-server ServerName][-fedonly] [-nofed]

        db2look [-h]

        -d: Database Name: This must be specified

        -e: Extract DDL file needed to duplicate database
       -xs: Export XSR objects and generate a script containing DDL statements
     -xdir: Path name: the directory in which XSR objects will be placed
        -u: Creator ID: If -u and -a are both not specified then $USER will be used
        -z: Schema name: If -z and -a are both specified then -z will be ignored
        -t: Generate statistics for the specified tables
       -tw: Generate DDLs for tables whose names match the pattern criteria (wildcard characters) of the table name
       -ap: Generate AUDIT USING Statements
      -wlm: Generate WLM specific DDL Statements
      -mod: Generate DDL statements for Module
      -cor: Generate DDL with CREATE OR REPLACE clause
     -wrap: Generates obfuscated versions of DDL statements
        -h: More detailed help message
        -o: Redirects the output to the given file name
        -a: Generate statistics for all creators
        -m: Run the db2look utility in mimic mode
            -c: Do not generate COMMIT statements for mimic
            -r: Do not generate RUNSTATS statements for mimic
        -l: Generate Database Layout: Database partition groups, Bufferpools and Tablespaces
        -x: Generate Authorization statements DDL excluding the original definer of the object
       -xd: Generate Authorization statements DDL including the original definer of the object
        -f: Extract configuration parameters and environment variables
       -td: Specifies x to be statement delimiter (default is semicolon(;))
        -i: User ID to log on to the server where the database resides
        -w: Password to log on to the server where the database resides
   -noview: Do not generate CREATE VIEW ddl statements
  -wrapper: Generates DDLs for federated objects that apply to this wrapper
   -server: Generates DDLs for federated objects that apply to this server
    -FEDONLY: Only created Federated DDL Statements
    -nofed: Do not generate Federated DDL
       -fd: Generates db2fopt statements for opt_buffpage and opt_sortheap along with other cfg and env parameters.
        -v: Generate DDL for view only, this option is ignored when -t is specified
       -dp: Generate DROP statement before CREATE statement
       -ct: Generate DDL Statements by object creation time
-noimplschema: Do not generate CREATE SCHEMA ddl for implicitly created schemas
-nostatsclause: Do not include statistics clause in CREATE INDEX DDL.
```

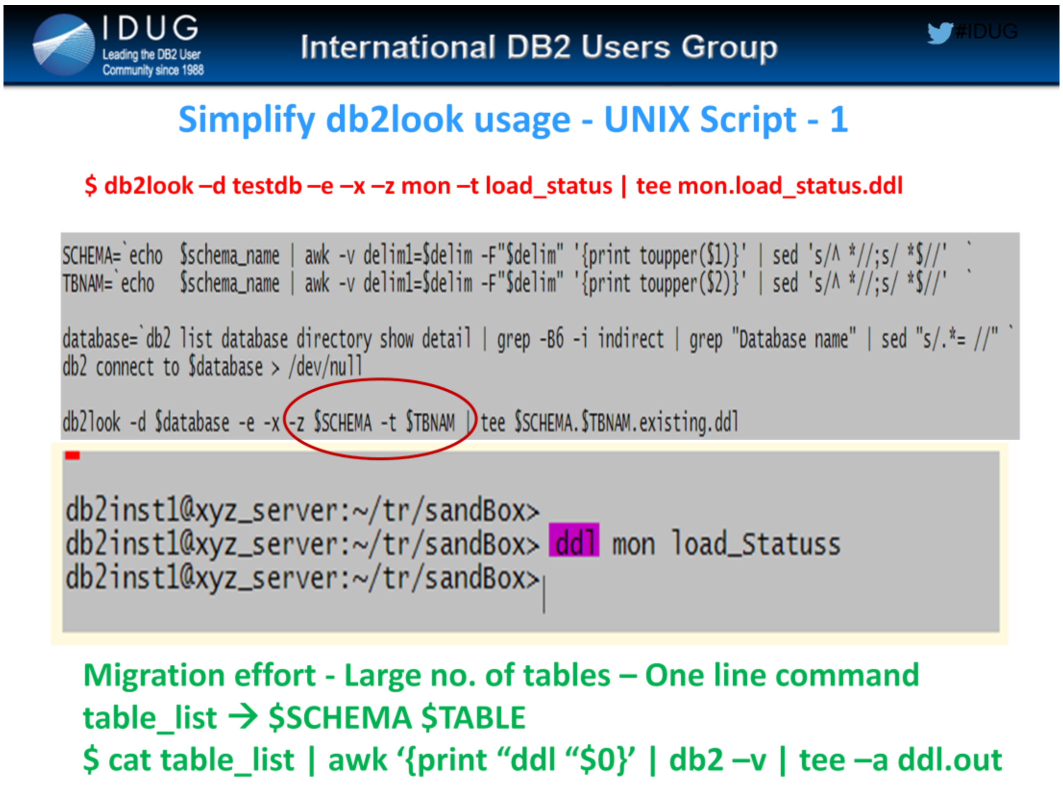
Complex syntax -- frustrating in a panic situation

DB2's native command 'db2look' is primarily used to extract ddl for database objects (tables / views).

db2look has lots of options in it and thus can become a long command to type.

When migrating tables from lower to high environments, we use db2look to extract ddl from lower environment (dev) and then to reply the ddl in higher environment.

A UNIX script that would avoid repetitious typing of complex typing is shared in next slide.


IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Simplify db2look usage - UNIX Script - 1

$ db2look –d testdb –e –x –z mon –t load_status | tee mon.load_status.ddl

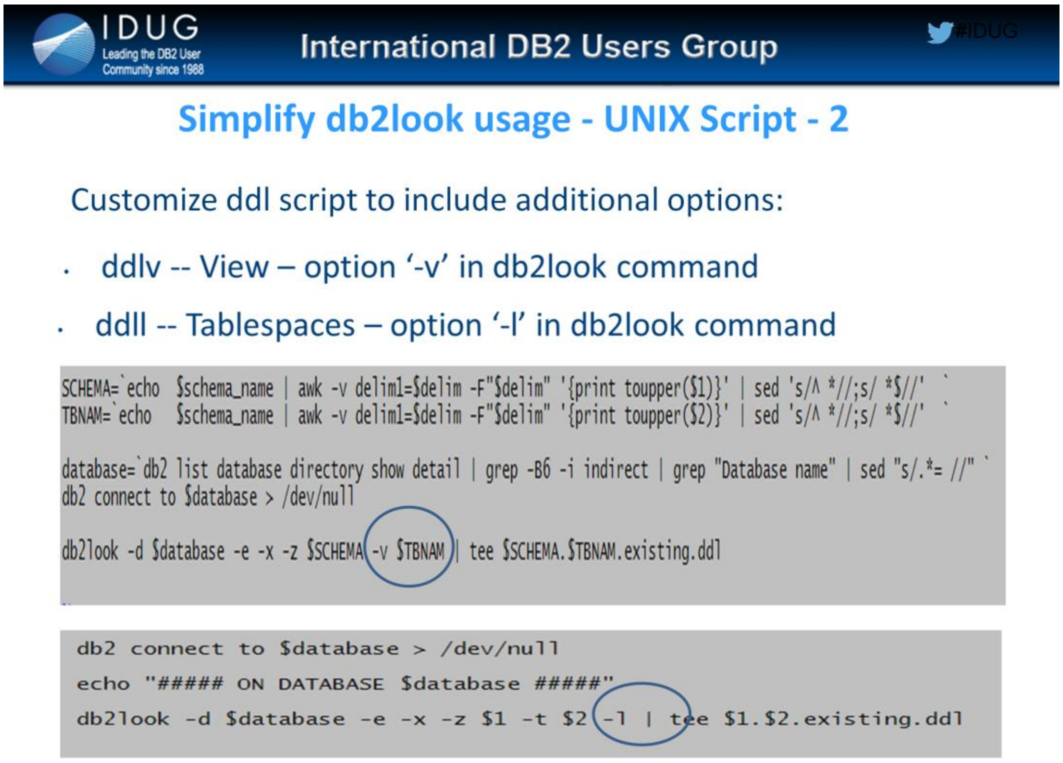
```
SCHEMA=`echo  $schema_name | awk -v deliml=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v deliml=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `

database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `
db2 connect to $database > /dev/null

db2look -d $database -e -x -z $SCHEMA -t $TBNAM | tee $SCHEMA.$TBNAM.existing.ddl
```

```
db2inst1@xyz_server:~/tr/sandBox>
db2inst1@xyz_server:~/tr/sandBox> ddl mon load_Statuss
db2inst1@xyz_server:~/tr/sandBox>
```

Migration effort - Large no. of tables – One line command
table_list → $SCHEMA $TABLE
$ cat table_list | awk '{print "ddl "$0}' | db2 –v | tee –a ddl.out

This slide shows the core part of a script "ddl" which is used to extract ddl for a 'table'

Usage: ddl $schema $tablename


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Simplify db2look usage - UNIX Script - 2

Customize ddl script to include additional options:

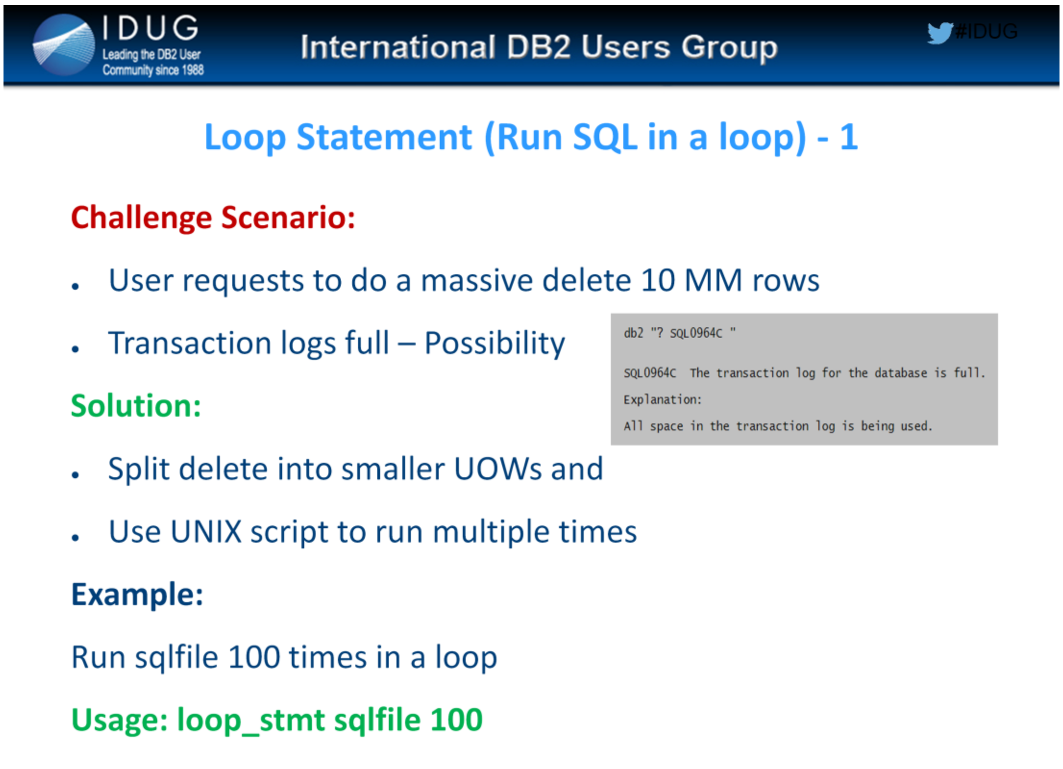
- ddlv -- View – option '-v' in db2look command
- ddll -- Tablespaces – option '-l' in db2look command

```
SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `

database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `
db2 connect to $database > /dev/null

db2look -d $database -e -x -z $SCHEMA -v $TBNAM | tee $SCHEMA.$TBNAM.existing.ddl
```

```
db2 connect to $database > /dev/null
echo "##### ON DATABASE $database #####"
db2look -d $database -e -x -z $1 -t $2 -l | tee $1.$2.existing.ddl
```

Usage:
ddlv $schema $viewname

ddll $schema $tablename



International DB2 Users Group

Loop Statement (Run SQL in a loop) - 1

Challenge Scenario:

- User requests to do a massive delete 10 MM rows
- Transaction logs full – Possibility

```
db2 "? SQL0964C "

SQL0964C  The transaction log for the database is full.

Explanation:

All space in the transaction log is being used.
```

Solution:

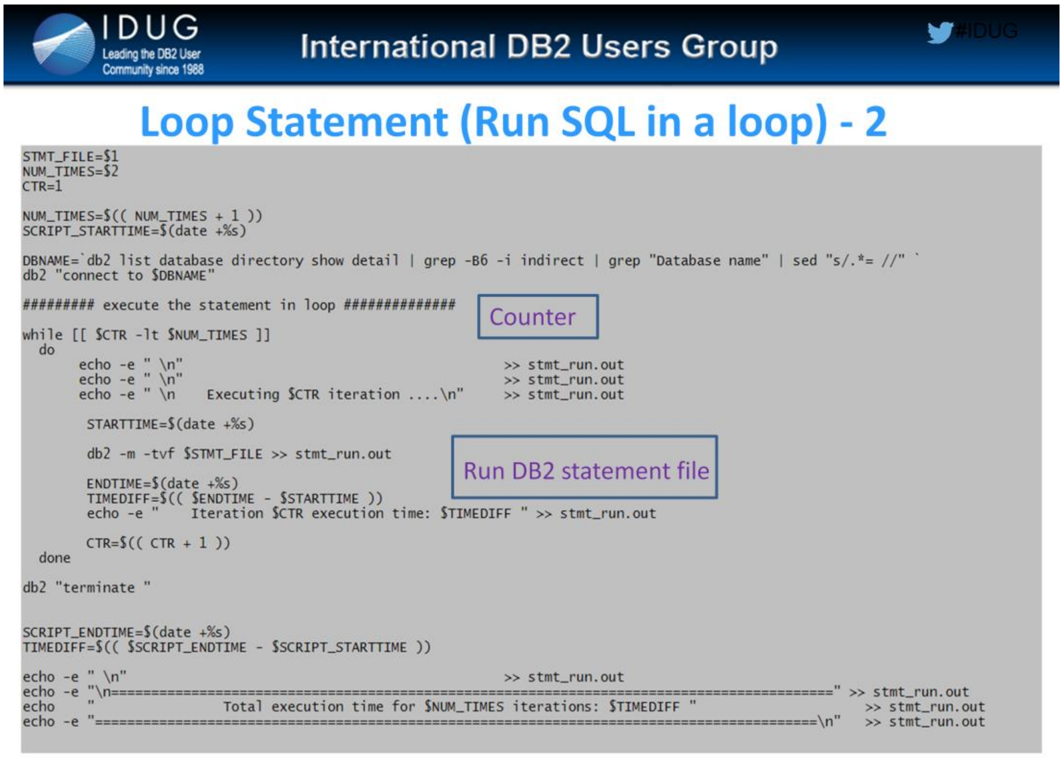
- Split delete into smaller UOWs and
- Use UNIX script to run multiple times

Example:

Run sqlfile 100 times in a loop

Usage: loop_stmt sqlfile 100

Use loop_stmt to run a small UOW multiple times instead of running a large UOW which has potential to fill up logs

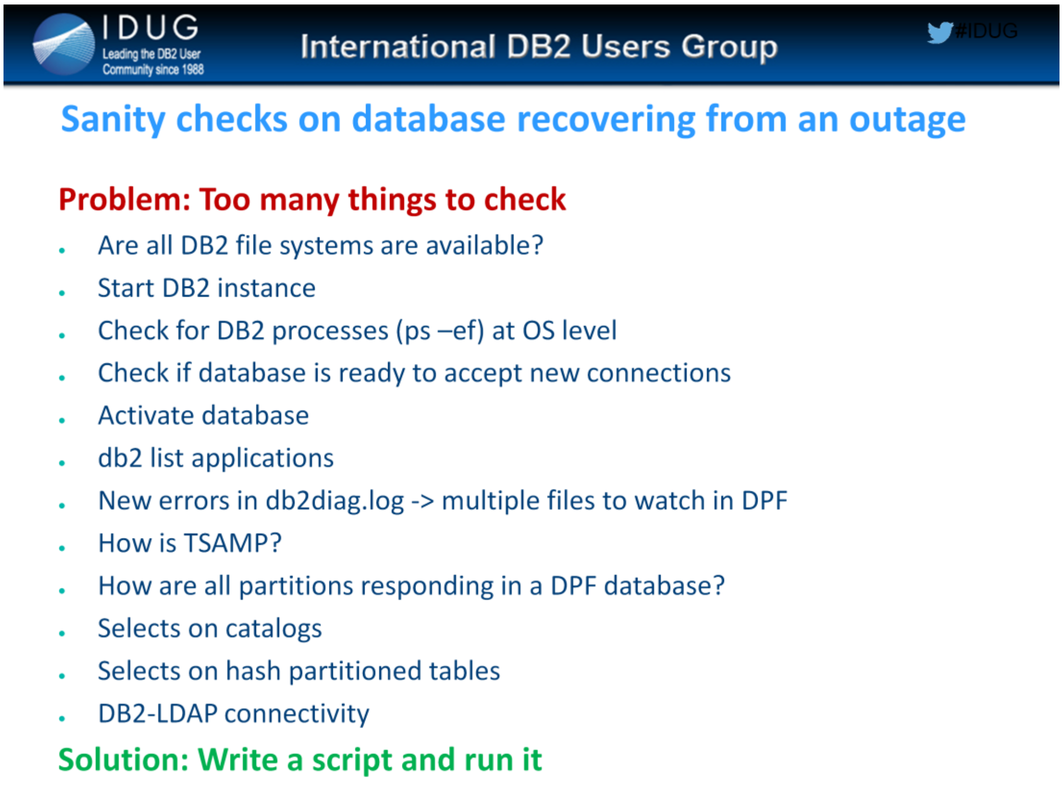

IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG
Counter
Run DB2 statement file

Loop Statement (Run SQL in a loop) - 2

```
STMT_FILE=$1
NUM_TIMES=$2
CTR=1

NUM_TIMES=$(( NUM_TIMES + 1 ))
SCRIPT_STARTTIME=$(date +%s)

DBNAME=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `
db2 "connect to $DBNAME"

######### execute the statement in loop ##############

while [[ $CTR -lt $NUM_TIMES ]]
  do
       echo -e " \n"                                       >> stmt_run.out
       echo -e " \n"                                       >> stmt_run.out
       echo -e " \n    Executing $CTR iteration ....\n"     >> stmt_run.out

        STARTTIME=$(date +%s)

        db2 -m -tvf $STMT_FILE >> stmt_run.out

        ENDTIME=$(date +%s)
        TIMEDIFF=$(( $ENDTIME - $STARTTIME ))
        echo -e "    Iteration $CTR execution time: $TIMEDIFF " >> stmt_run.out

        CTR=$(( CTR + 1 ))
  done

db2 "terminate "


SCRIPT_ENDTIME=$(date +%s)
TIMEDIFF=$(( $SCRIPT_ENDTIME - $SCRIPT_STARTTIME ))

echo -e " \n"                                              >> stmt_run.out
echo -e "\n==========================================================================================" >> stmt_run.out
echo    "                    Total execution time for $NUM_TIMES iterations: $TIMEDIFF "              >> stmt_run.out
echo -e "==========================================================================================\n"   >> stmt_run.out
```

Usage:

loop_stmt $sqlfile $no_of_times_to_run


IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Sanity checks on database recovering from an outage

Problem: Too many things to check

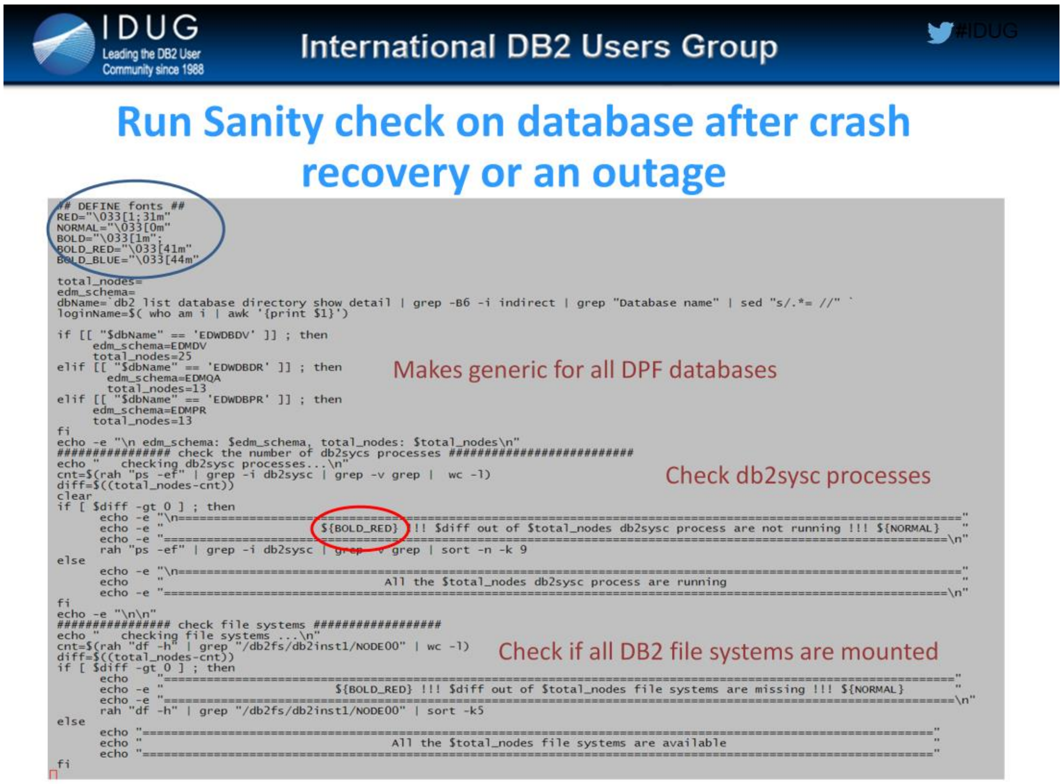
- Are all DB2 file systems are available?
- Start DB2 instance
- Check for DB2 processes (ps –ef) at OS level
- Check if database is ready to accept new connections
- Activate database
- db2 list applications
- New errors in db2diag.log -> multiple files to watch in DPF
- How is TSAMP?
- How are all partitions responding in a DPF database?
- Selects on catalogs
- Selects on hash partitioned tables
- DB2-LDAP connectivity

Solution: Write a script and run it

Normally as DBAs we run through a list of exhaustive steps to make sure database health is okay once database recovers from an outage.

This slide has 10+ steps that we do manually every time an outage works.

Instead of doing this, a script to perform all the checks would be a good idea. This is shared in the next slide.



IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Run Sanity check on database after crash recovery or an outage

```
## DEFINE fonts ##
RED="\033[1;31m"
NORMAL="\033[0m"
BOLD="\033[1m";
BOLD_RED="\033[41m"
BOLD_BLUE="\033[44m"

total_nodes=
edm_schema=
dbName=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `
loginName=$( who am i | awk '{print $1}')

if [[ "$dbName" == 'EDWDBDV' ]] ; then
     edm_schema=EDMDV
     total_nodes=25
elif [[ "$dbName" == 'EDWDBDR' ]] ; then
       edm_schema=EDMQA
       total_nodes=13
elif [[ "$dbName" == 'EDWDBPR' ]] ; then
     edm_schema=EDMPR
     total_nodes=13
fi
echo -e "\n edm_schema: $edm_schema, total_nodes: $total_nodes\n"
################# check the number of db2sycs processes ##########################
echo "   checking db2sysc processes...\n"
cnt=$(rah "ps -ef" | grep -i db2sysc | grep -v grep |  wc -l)
diff=$((total_nodes-cnt))
clear
if [ $diff -gt 0 ] ; then
      echo -e "\n================================================================================================"
      echo -e "                           ${BOLD_RED} !!! $diff out of $total_nodes db2sysc process are not running !!! ${NORMAL}    "
      echo -e "================================================================================================\n"
      rah "ps -ef" | grep -i db2sysc | grep -v grep | sort -n -k 9
else
      echo -e "\n================================================================================================"
      echo    "                                  All the $total_nodes db2sysc process are running                  "
      echo -e "================================================================================================\n"
fi
echo -e "\n\n"
################# check file systems ##################
echo "   checking file systems ...\n"
cnt=$(rah "df -h" | grep "/db2fs/db2inst1/NODE00" | wc -l)
diff=$((total_nodes-cnt))
if [ $diff -gt 0 ] ; then
      echo    "================================================================================================"
      echo -e "                           ${BOLD_RED} !!! $diff out of $total_nodes file systems are missing !!! ${NORMAL}    "
      echo -e "================================================================================================\n"
      rah "df -h" | grep "/db2fs/db2inst1/NODE00" | sort -k5
else
      echo "============================================================================================="
      echo "                                   All the $total_nodes file systems are available               "
      echo "============================================================================================="
fi
```

Makes generic for all DPF databases

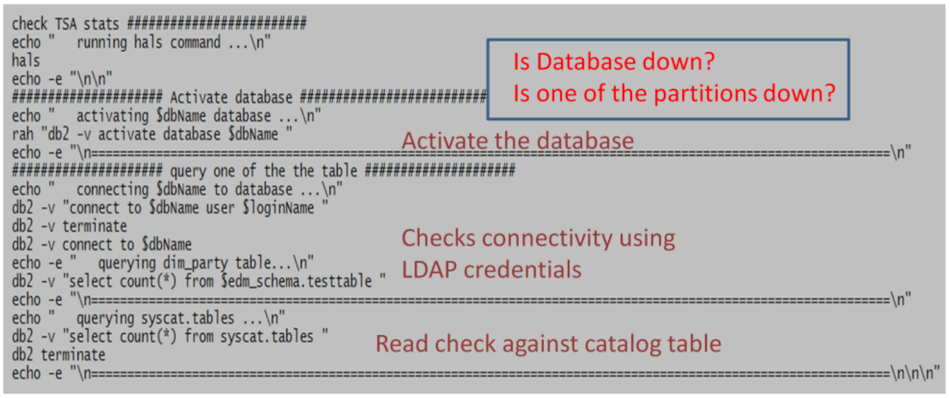
Check db2sysc processes

Check if all DB2 file systems are mounted

This script quickly checks the database availability with following checks:

1. Checks db2sysc procs on all nodes.
2. Checks the file system availability.
3. Checks the TSA status.
4. Activates database (if it is already not). Queries a table.
5. Lists the applications
6. Queries a catalog table
7. Tails the diag log

Run Sanity check on database after crash recovery



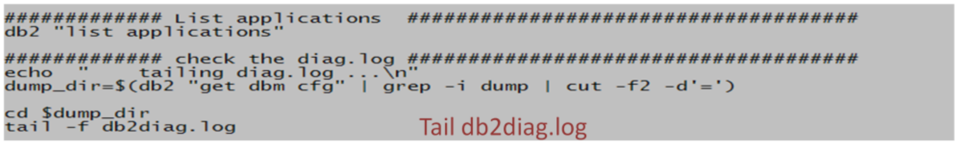
```
check TSA stats #########################
echo "   running hals command ...\n"
hals
echo -e "\n\n"
#################### Activate database #########################
echo "   activating $dbName database ...\n"
rah "db2 -v activate database $dbName "
echo -e "\n=====================================================================================================\n"
#################### query one of the the table ####################
echo "   connecting $dbName to database ...\n"
db2 -v "connect to $dbName user $loginName "
db2 -v terminate
db2 -v connect to $dbName
echo -e "   querying dim_party table...\n"
db2 -v "select count(*) from $edm_schema.testtable "
echo -e "\n=====================================================================================================\n"
echo "   querying syscat.tables ...\n"
db2 -v "select count(*) from syscat.tables "
db2 terminate
echo -e "\n=====================================================================================================\n\n\n"
```

Is Database down?
Is one of the partitions down?

Activate the database

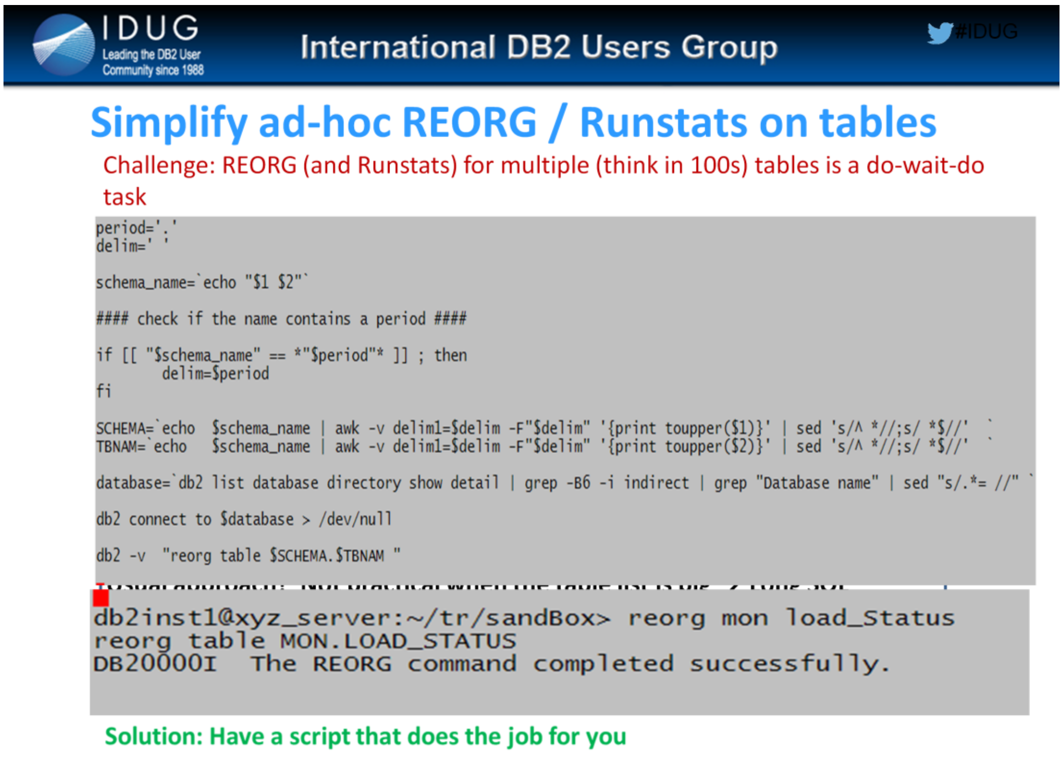
Checks connectivity using
LDAP credentials

Read check against catalog table



```
############# List applications  ######################################
db2 "list applications"

############# check the diag.log ######################################
echo  "     tailing diag.log ...\n"
dump_dir=$(db2 "get dbm cfg" | grep -i dump | cut -f2 -d'=')

cd $dump_dir
tail -f db2diag.log
```

Tail db2diag.log



IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Simplify ad-hoc REORG / Runstats on tables

Challenge: REORG (and Runstats) for multiple (think in 100s) tables is a do-wait-do task

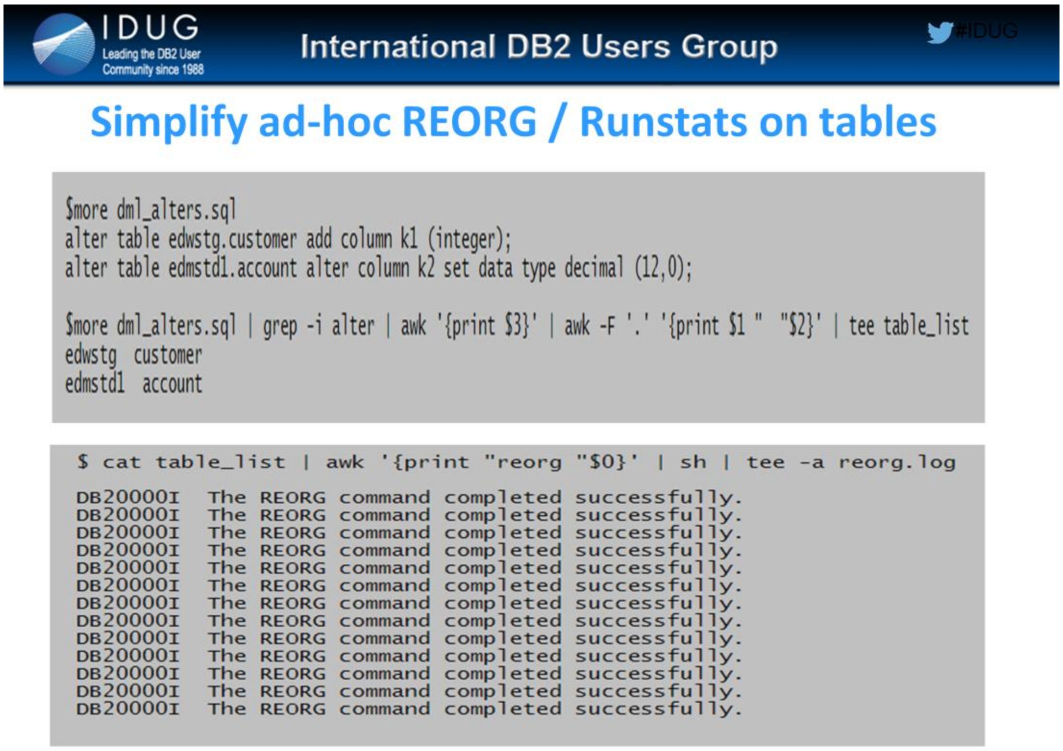
```
period='.'
delim=' '

schema_name=`echo "$1 $2"`

#### check if the name contains a period ####

if [[ "$schema_name" == *"$period"* ]] ; then
        delim=$period
fi

SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `

database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null

db2 -v  "reorg table $SCHEMA.$TBNAM "
```

```
db2inst1@xyz_server:~/tr/sandBox> reorg mon load_Status
reorg table MON.LOAD_STATUS
DB20000I  The REORG command completed successfully.
```

Solution: Have a script that does the job for you

Simple script that would ‘reorg’ a DB2 table.

The main time saving that DBAs would realize is when there are multiple tables that need to be reorg’d.


IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

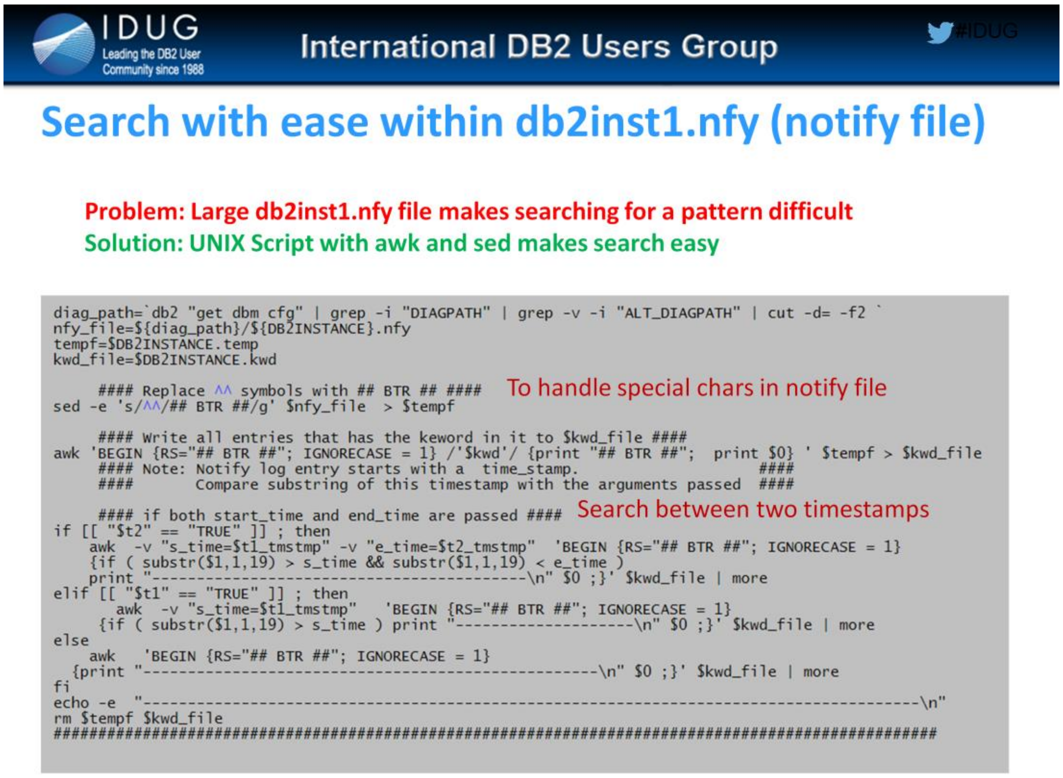
Simplify ad-hoc REORG / Runstats on tables

```
$more dml_alters.sql
alter table edwstg.customer add column k1 (integer);
alter table edmstd1.account alter column k2 set data type decimal (12,0);

$more dml_alters.sql | grep -i alter | awk '{print $3}' | awk -F '.' '{print $1 "  "$2}' | tee table_list
edwstg  customer
edmstd1  account
```

```
$ cat table_list | awk '{print "reorg "$0}' | sh | tee -a reorg.log

DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
DB20000I  The REORG command completed successfully.
```

Simple script that would ‘reorg’ a DB2 table.

The main time saving that DBAs would realize is when there are multiple tables that need to be reorg’d.

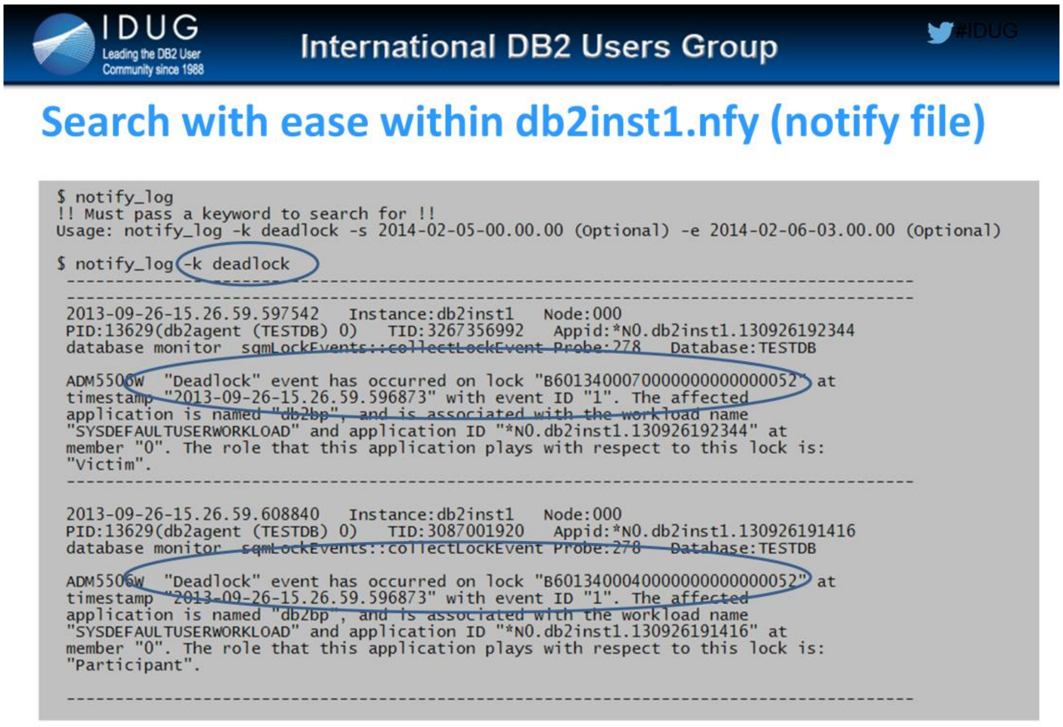

IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

Search with ease within db2inst1.nfy (notify file)

Problem: Large db2inst1.nfy file makes searching for a pattern difficult
Solution: UNIX Script with awk and sed makes search easy

```
diag_path=`db2 "get dbm cfg" | grep -i "DIAGPATH" | grep -v -i "ALT_DIAGPATH" | cut -d= -f2 `
nfy_file=${diag_path}/${DB2INSTANCE}.nfy
tempf=$DB2INSTANCE.temp
kwd_file=$DB2INSTANCE.kwd

     #### Replace ^^ symbols with ## BTR ## ####   To handle special chars in notify file
sed -e 's/^^/## BTR ##/g' $nfy_file  > $tempf

     #### Write all entries that has the keword in it to $kwd_file ####
awk 'BEGIN {RS="## BTR ##"; IGNORECASE = 1} /'$kwd'/ {print "## BTR ##";  print $0} ' $tempf > $kwd_file
     #### Note: Notify log entry starts with a  time_stamp.                  ####
     ####       Compare substring of this timestamp with the arguments passed  ####

     #### if both start_time and end_time are passed ####  Search between two timestamps
if [[ "$t2" == "TRUE" ]] ; then
    awk  -v "s_time=$t1_tmstmp" -v "e_time=$t2_tmstmp"  'BEGIN {RS="## BTR ##"; IGNORECASE = 1}
    {if ( substr($1,1,19) > s_time && substr($1,1,19) < e_time )
    print "------------------------------------------\n" $0 ;}' $kwd_file | more
elif [[ "$t1" == "TRUE" ]] ; then
       awk  -v "s_time=$t1_tmstmp"   'BEGIN {RS="## BTR ##"; IGNORECASE = 1}
     {if ( substr($1,1,19) > s_time ) print "--------------------\n" $0 ;}' $kwd_file | more
else
    awk   'BEGIN {RS="## BTR ##"; IGNORECASE = 1}
  {print "---------------------------------------------------\n" $0 ;}' $kwd_file | more
fi
echo -e  "-----------------------------------------------------------------------------------\n"
rm $tempf $kwd_file
#################################################################################################
```

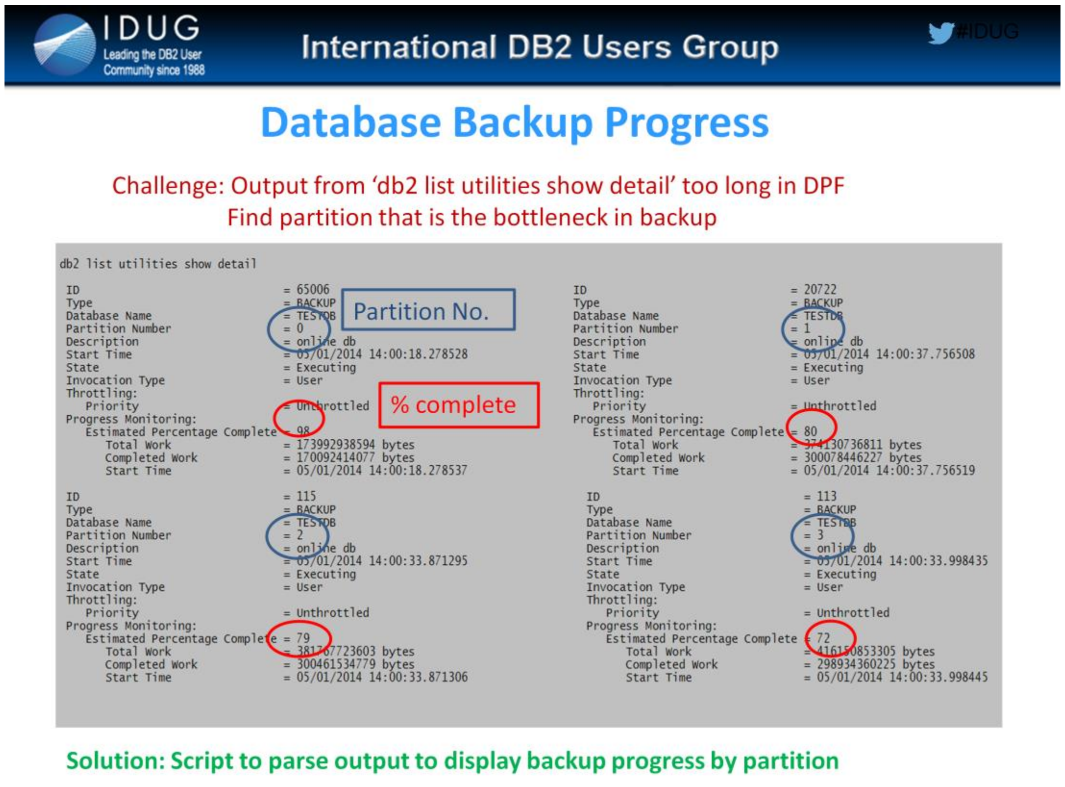
Browsing through the notify log for a specific thing is hard. Especially when you are diagnosing some problem/incident, it takes quite some time to nail it down. This script filters the notify log entries based on the key word you specified. You can also specify the time frame for limiting the matching entries into that specific time frame.


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

Search with ease within db2inst1.nfy (notify file)

```
$ notify_log
!! Must pass a keyword to search for !!
Usage: notify_log -k deadlock -s 2014-02-05-00.00.00 (Optional) -e 2014-02-06-03.00.00 (Optional)

$ notify_log -k deadlock
 ------------------------------------------------------------------------------------
 ------------------------------------------------------------------------------------
 2013-09-26-15.26.59.597542   Instance:db2inst1   Node:000
 PID:13629(db2agent (TESTDB) 0)   TID:3267356992   Appid:*N0.db2inst1.130926192344
 database monitor  sqmLockEvents::collectLockEvent Probe:278   Database:TESTDB

 ADM5506W  "Deadlock" event has occurred on lock "B6013400070000000000000052" at
 timestamp "2013-09-26-15.26.59.596873" with event ID "1". The affected
 application is named "db2bp", and is associated with the workload name
 "SYSDEFAULTUSERWORKLOAD" and application ID "*N0.db2inst1.130926192344" at
 member "0". The role that this application plays with respect to this lock is:
 "Victim".
 ------------------------------------------------------------------------------------

 2013-09-26-15.26.59.608840   Instance:db2inst1   Node:000
 PID:13629(db2agent (TESTDB) 0)   TID:3087001920   Appid:*N0.db2inst1.130926191416
 database monitor  sqmLockEvents::collectLockEvent Probe:278   Database:TESTDB

 ADM5506W  "Deadlock" event has occurred on lock "B6013400040000000000000052" at
 timestamp "2013-09-26-15.26.59.596873" with event ID "1". The affected
 application is named "db2bp", and is associated with the workload name
 "SYSDEFAULTUSERWORKLOAD" and application ID "*N0.db2inst1.130926191416" at
 member "0". The role that this application plays with respect to this lock is:
 "Participant".

 ------------------------------------------------------------------------------------
```

Sample usage of script notify_log



IDUG
Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Database Backup Progress

Challenge: Output from ‘db2 list utilities show detail’ too long in DPF
Find partition that is the bottleneck in backup

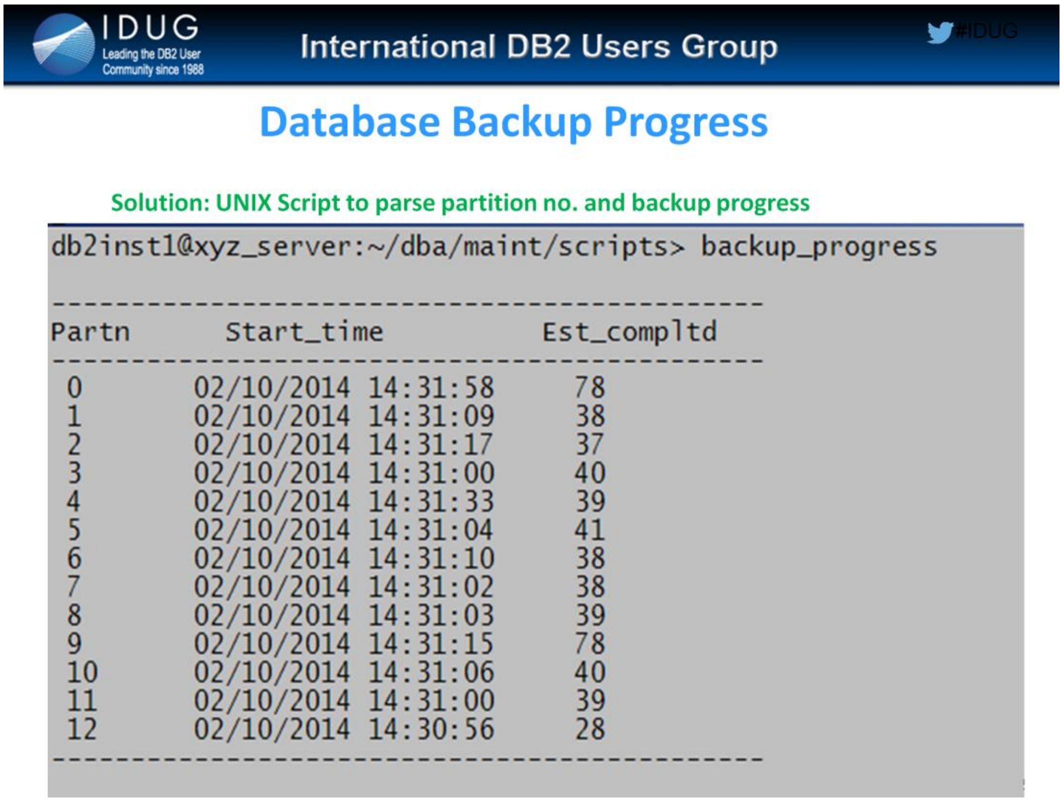
```
db2 list utilities show detail

 ID                                = 65006
 Type                              = BACKUP
 Database Name                     = TESTDB
 Partition Number                  = 0
 Description                       = online db
 Start Time                        = 05/01/2014 14:00:18.278528
 State                             = Executing
 Invocation Type                   = User
 Throttling:
    Priority                       = Unthrottled
 Progress Monitoring:
    Estimated Percentage Complete = 98
       Total Work                  = 173992938594 bytes
       Completed Work              = 170092414077 bytes
       Start Time                  = 05/01/2014 14:00:18.278537

 ID                                = 20722
 Type                              = BACKUP
 Database Name                     = TESTDB
 Partition Number                  = 1
 Description                       = online db
 Start Time                        = 05/01/2014 14:00:37.756508
 State                             = Executing
 Invocation Type                   = User
 Throttling:
    Priority                       = Unthrottled
 Progress Monitoring:
    Estimated Percentage Complete = 80
       Total Work                  = 374130736811 bytes
       Completed Work              = 300078446227 bytes
       Start Time                  = 05/01/2014 14:00:37.756519

 ID                                = 115
 Type                              = BACKUP
 Database Name                     = TESTDB
 Partition Number                  = 2
 Description                       = online db
 Start Time                        = 05/01/2014 14:00:33.871295
 State                             = Executing
 Invocation Type                   = User
 Throttling:
    Priority                       = Unthrottled
 Progress Monitoring:
    Estimated Percentage Complete = 79
       Total Work                  = 381767723603 bytes
       Completed Work              = 300461534779 bytes
       Start Time                  = 05/01/2014 14:00:33.871306

 ID                                = 113
 Type                              = BACKUP
 Database Name                     = TESTDB
 Partition Number                  = 3
 Description                       = online db
 Start Time                        = 05/01/2014 14:00:33.998435
 State                             = Executing
 Invocation Type                   = User
 Throttling:
    Priority                       = Unthrottled
 Progress Monitoring:
    Estimated Percentage Complete = 72
       Total Work                  = 416150853305 bytes
       Completed Work              = 298934360225 bytes
       Start Time                  = 05/01/2014 14:00:33.998445
```

Partition No.

% complete

Solution: Script to parse output to display backup progress by partition

The output of “db2 list utilities show detail” on a DPF database could span multiple pages and this makes it challenging to browse for backup progress.

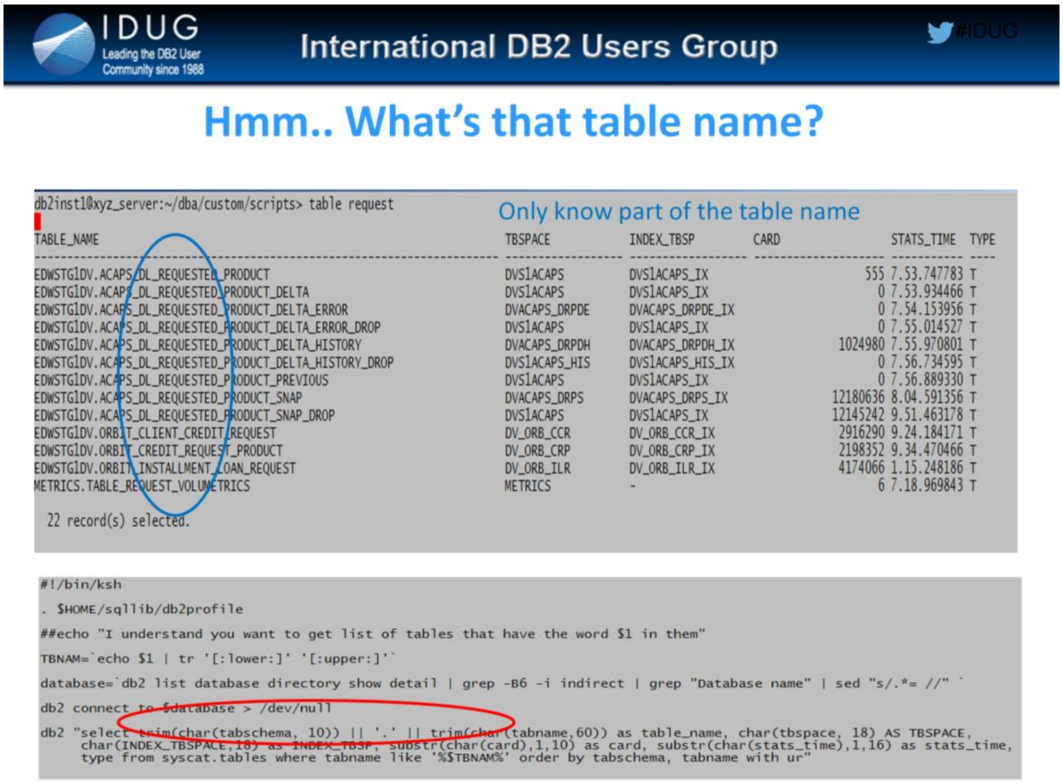

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Database Backup Progress

Solution: UNIX Script to parse partition no. and backup progress

```
db2inst1@xyz_server:~/dba/maint/scripts> backup_progress

---------------------------------------------
Partn       Start_time           Est_compltd
---------------------------------------------
 0       02/10/2014 14:31:58      78
 1       02/10/2014 14:31:09      38
 2       02/10/2014 14:31:17      37
 3       02/10/2014 14:31:00      40
 4       02/10/2014 14:31:33      39
 5       02/10/2014 14:31:04      41
 6       02/10/2014 14:31:10      38
 7       02/10/2014 14:31:02      38
 8       02/10/2014 14:31:03      39
 9       02/10/2014 14:31:15      78
 10      02/10/2014 14:31:06      40
 11      02/10/2014 14:31:00      39
 12      02/10/2014 14:30:56      28
---------------------------------------------
```

UNIX script to help to identify backup progress across all database partitions. This also helps to identify bottleneck partitions for backup.


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

Hmm.. What's that table name?

Only know part of the table name

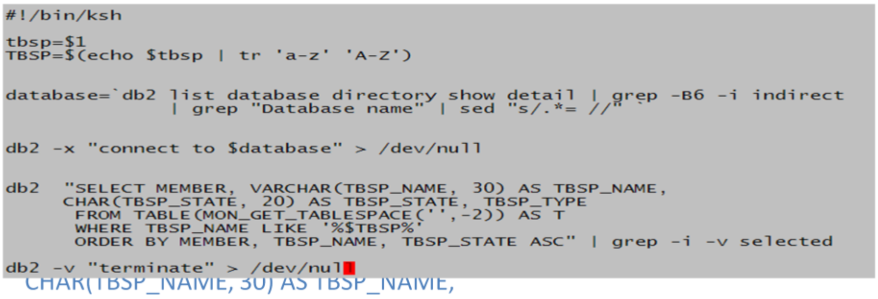
```
db2inst1@xyz_server:~/dba/custom/scripts> table request

TABLE_NAME                                                              TBSPACE            INDEX_TBSP         CARD                 STATS_TIME  TYPE
----------------------------------------------------------------------- ------------------ ------------------ -------------------- ----------- ----
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT                                    DVS1ACAPS          DVS1ACAPS_IX                        555 7.53.747783 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_DELTA                              DVS1ACAPS          DVS1ACAPS_IX                          0 7.53.934466 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_DELTA_ERROR                        DVACAPS_DRPDE      DVACAPS_DRPDE_IX                      0 7.54.153956 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_DELTA_ERROR_DROP                   DVS1ACAPS          DVS1ACAPS_IX                          0 7.55.014527 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_DELTA_HISTORY                      DVACAPS_DRPDH      DVACAPS_DRPDH_IX                1024980 7.55.970801 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_DELTA_HISTORY_DROP                 DVS1ACAPS_HIS      DVS1ACAPS_HIS_IX                      0 7.56.734595 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_PREVIOUS                           DVS1ACAPS          DVS1ACAPS_IX                          0 7.56.889330 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_SNAP                               DVACAPS_DRPS       DVACAPS_DRPS_IX                12180636 8.04.591356 T
EDWSTG1DV.ACAPS_DL_REQUESTED_PRODUCT_SNAP_DROP                          DVS1ACAPS          DVS1ACAPS_IX                   12145242 9.51.463178 T
EDWSTG1DV.ORBIT_CLIENT_CREDIT_REQUEST                                   DV_ORB_CCR         DV_ORB_CCR_IX                   2916290 9.24.184171 T
EDWSTG1DV.ORBIT_CREDIT_REQUEST_PRODUCT                                  DV_ORB_CRP         DV_ORB_CRP_IX                   2198352 9.34.470466 T
EDWSTG1DV.ORBIT_INSTALLMENT_LOAN_REQUEST                                DV_ORB_ILR         DV_ORB_ILR_IX                   4174066 1.15.248186 T
METRICS.TABLE_REQUEST_VOLUMETRICS                                       METRICS            -                                     6 7.18.969843 T

  22 record(s) selected.
```

```
#!/bin/ksh
. $HOME/sqllib/db2profile
##echo "I understand you want to get list of tables that have the word $1 in them"
TBNAM=`echo $1 | tr '[:lower:]' '[:upper:]'`
database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `
db2 connect to $database > /dev/null
db2 "select trim(char(tabschema, 10)) || '.' || trim(char(tabname,60)) as table_name, char(tbspace, 18) AS TBSPACE,
     char(INDEX_TBSPACE,18) as INDEX_TBSP, substr(char(card),1,10) as card, substr(char(stats_time),1,16) as stats_time,
     type from syscat.tables where tabname like '%$TBNAM%' order by tabschema, tabname with ur"
```

This script helps in finding a table name, when you are not sure of the full name of it, but you know a part of it. It list all the tables that matches the word you provide.

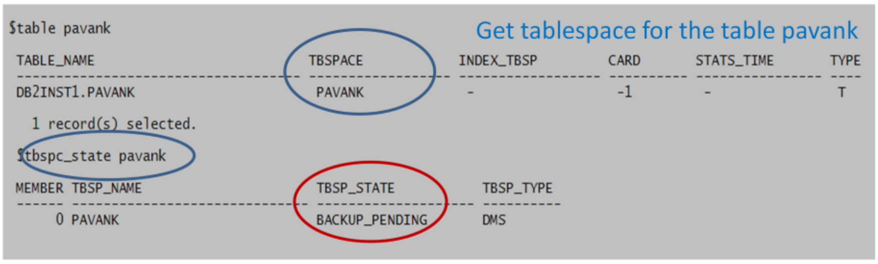
Tablespace state

Problem: User cannot run DML statements. Error Message SQL0290N



```
#!/bin/ksh
tbsp=$1
TBSP=$(echo $tbsp | tr 'a-z' 'A-Z')

database=`db2 list database directory show detail | grep -B6 -i indirect
              | grep "Database name" | sed "s/.*= //"`

db2 -x "connect to $database" > /dev/null

db2  "SELECT MEMBER, VARCHAR(TBSP_NAME, 30) AS TBSP_NAME,
     CHAR(TBSP_STATE, 20) AS TBSP_STATE, TBSP_TYPE
      FROM TABLE(MON_GET_TABLESPACE('',-2)) AS T
      WHERE TBSP_NAME LIKE '%$TBSP%'
      ORDER BY MEMBER, TBSP_NAME, TBSP_STATE ASC" | grep -i -v selected

db2 -v "terminate" > /dev/null
```

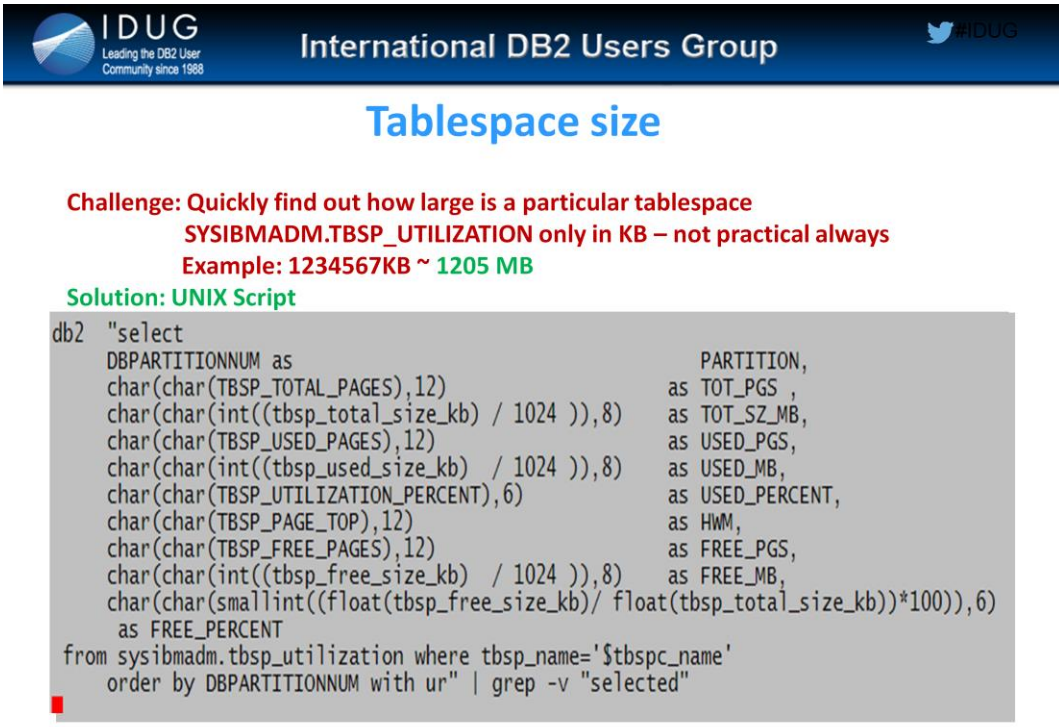
CHAR(TBSP_NAME, 30) AS TBSP_NAME,



Get tablespace for the table pavank

```
$table pavank

TABLE_NAME                          TBSPACE            INDEX_TBSP         CARD       STATS_TIME        TYPE
----------------------------------- ------------------ ------------------ ---------- ----------------- ----
DB2INST1.PAVANK                     PAVANK             -                  -1         -                 T

  1 record(s) selected.

$tbspc_state pavank

MEMBER TBSP_NAME                     TBSP_STATE             TBSP_TYPE
------ ----------------------------- ---------------------- ----------
     0 PAVANK                        BACKUP_PENDING         DMS
```

This script, gives the state of a tablespace in all the participating partitions.


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

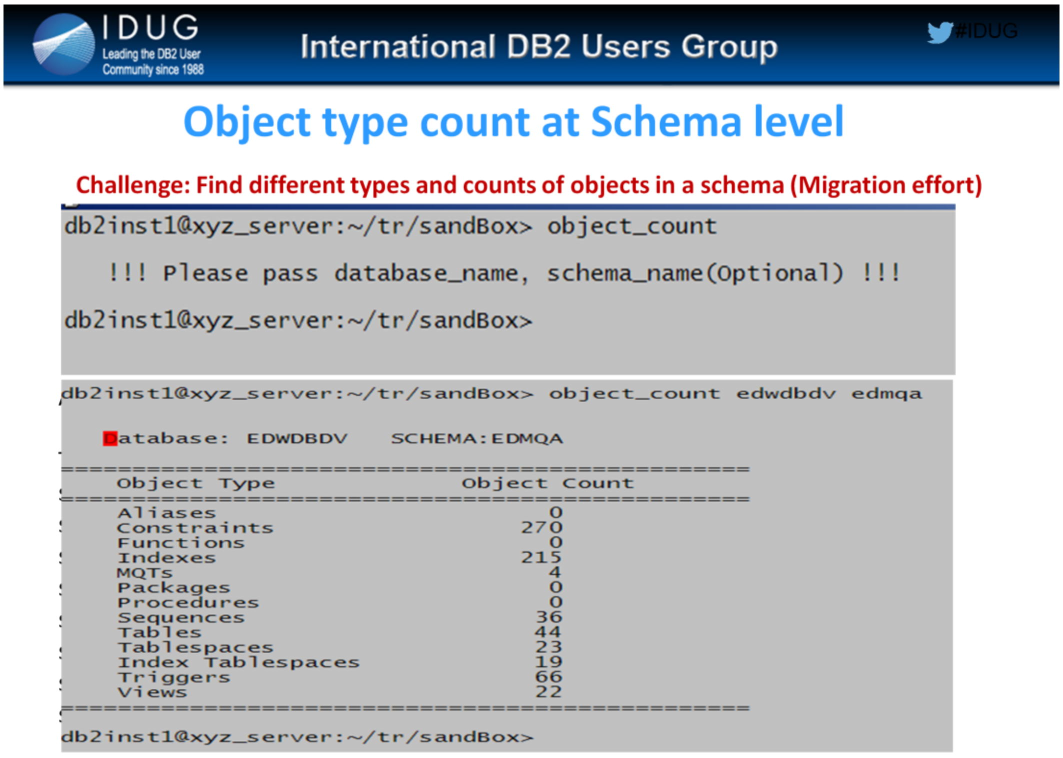
Tablespace size

Challenge: Quickly find out how large is a particular tablespace
SYSIBMADM.TBSP_UTILIZATION only in KB – not practical always
Example: 1234567KB ~ 1205 MB

Solution: UNIX Script

```
db2  "select
     DBPARTITIONNUM as                                      PARTITION,
     char(char(TBSP_TOTAL_PAGES),12)                     as TOT_PGS ,
     char(char(int((tbsp_total_size_kb) / 1024 )),8)     as TOT_SZ_MB,
     char(char(TBSP_USED_PAGES),12)                      as USED_PGS,
     char(char(int((tbsp_used_size_kb)  / 1024 )),8)     as USED_MB,
     char(char(TBSP_UTILIZATION_PERCENT),6)              as USED_PERCENT,
     char(char(TBSP_PAGE_TOP),12)                        as HWM,
     char(char(TBSP_FREE_PAGES),12)                      as FREE_PGS,
     char(char(int((tbsp_free_size_kb)  / 1024 )),8)     as FREE_MB,
     char(char(smallint((float(tbsp_free_size_kb)/ float(tbsp_total_size_kb))*100)),6)
      as FREE_PERCENT
 from sysibmadm.tbsp_utilization where tbsp_name='$tbspc_name'
     order by DBPARTITIONNUM with ur" | grep -v "selected"
```

This script, gives you the details of tablespace like total pages, total space allocated, used space, high water mark, used percent, free pages, free space and free space percent at partition level. It also gives the overall size of the tablespace.



Object type count at Schema level

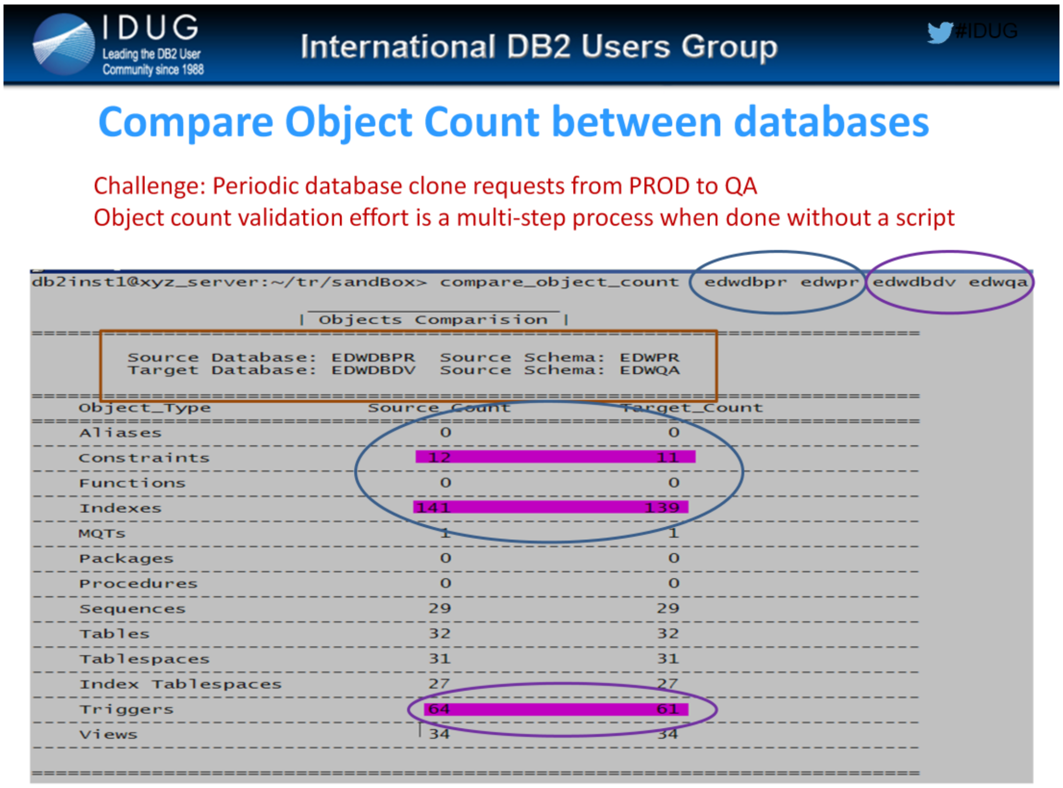
Challenge: Find different types and counts of objects in a schema (Migration effort)

```
db2inst1@xyz_server:~/tr/sandBox> object_count

   !!! Please pass database_name, schema_name(Optional) !!!

db2inst1@xyz_server:~/tr/sandBox>
```

```
db2inst1@xyz_server:~/tr/sandBox> object_count edwdbdv edmqa

   Database:  EDWDBDV    SCHEMA:EDMQA
==================================================
    Object Type                    Object Count
==================================================
    Aliases                              0
    Constraints                        270
    Functions                            0
    Indexes                            215
    MQTs                                 4
    Packages                             0
    Procedures                           0
    Sequences                           36
    Tables                              44
    Tablespaces                         23
    Index Tablespaces                   19
    Triggers                            66
    Views                               22
==================================================
db2inst1@xyz_server:~/tr/sandBox>
```

This script gives a snapshot of counts of different database object types in a given schema.

This script is a lengthy one to include as a snapshot or in the notes. Please email if you want the source code for this one.

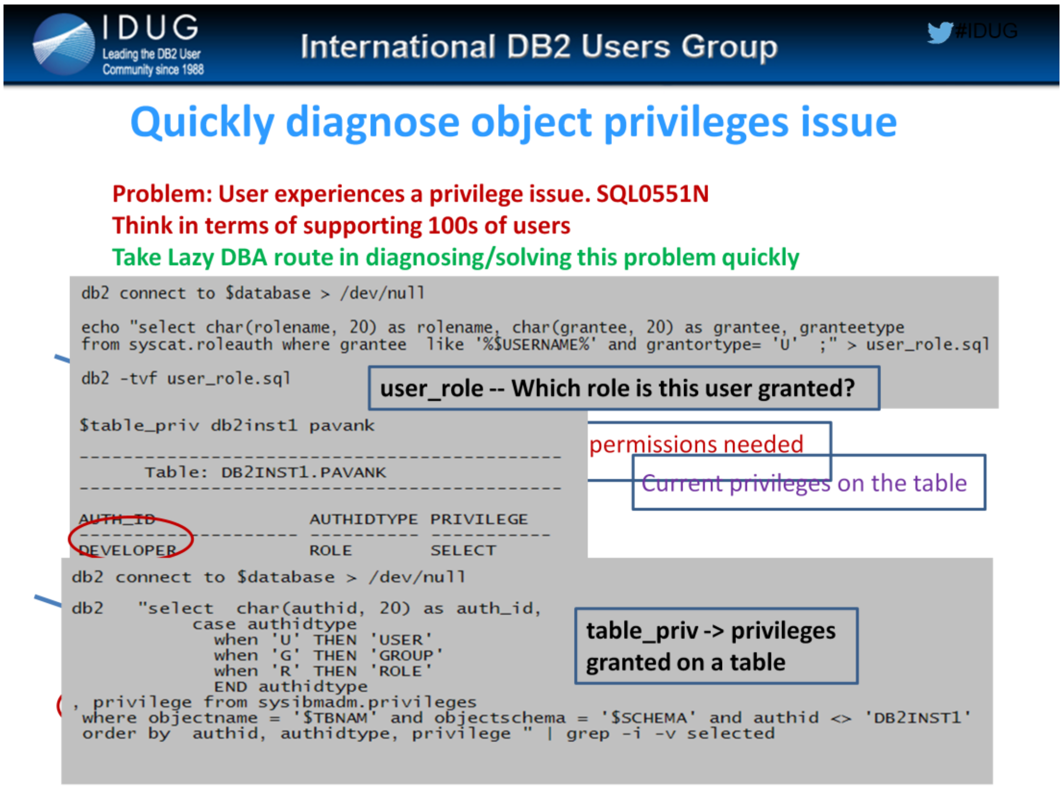

IDUG Leading the DB2 User Community since 1988 — International DB2 Users Group — #IDUG

Compare Object Count between databases

Challenge: Periodic database clone requests from PROD to QA
Object count validation effort is a multi-step process when done without a script

```
db2inst1@xyz_server:~/tr/sandBox> compare_object_count  edwdbpr  edwpr  edwdbdv  edwqa

                                  | Objects Comparision |
===========================================================================================
          Source Database:  EDWDBPR   Source Schema:  EDWPR
          Target Database:  EDWDBDV   Source Schema:  EDWQA

===========================================================================================
     Object_Type                      Source_Count                Target_Count
===========================================================================================
     Aliases                                   0                        0
-------------------------------------------------------------------------------------------
     Constraints                              12                       11
-------------------------------------------------------------------------------------------
     Functions                                 0                        0
-------------------------------------------------------------------------------------------
     Indexes                                 141                      139
-------------------------------------------------------------------------------------------
     MQTs                                      1                        1
-------------------------------------------------------------------------------------------
     Packages                                  0                        0
-------------------------------------------------------------------------------------------
     Procedures                                0                        0
-------------------------------------------------------------------------------------------
     Sequences                                29                       29
-------------------------------------------------------------------------------------------
     Tables                                   32                       32
-------------------------------------------------------------------------------------------
     Tablespaces                              31                       31
-------------------------------------------------------------------------------------------
     Index Tablespaces                        27                       27
-------------------------------------------------------------------------------------------
     Triggers                                 64                       61
-------------------------------------------------------------------------------------------
     Views                                    34                       34
-------------------------------------------------------------------------------------------

===========================================================================================
```

This script comes in handy, if you ever wanted to compare number of objects across the environments between two schemas and to find if there are any missing objects between two databases.


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Quickly diagnose object privileges issue

Problem: User experiences a privilege issue. SQL0551N
Think in terms of supporting 100s of users
Take Lazy DBA route in diagnosing/solving this problem quickly

```
db2 connect to $database > /dev/null

echo "select char(rolename, 20) as rolename, char(grantee, 20) as grantee, granteetype
from syscat.roleauth where grantee  like '%$USERNAME%' and grantortype= 'U'  ;" > user_role.sql

db2 -tvf user_role.sql
```

user_role -- Which role is this user granted?

```
$table_priv db2inst1 pavank

--------------------------------------------
      Table: DB2INST1.PAVANK
--------------------------------------------

AUTH_ID              AUTHIDTYPE  PRIVILEGE
-------------------- ----------  -----------
DEVELOPER            ROLE        SELECT
```

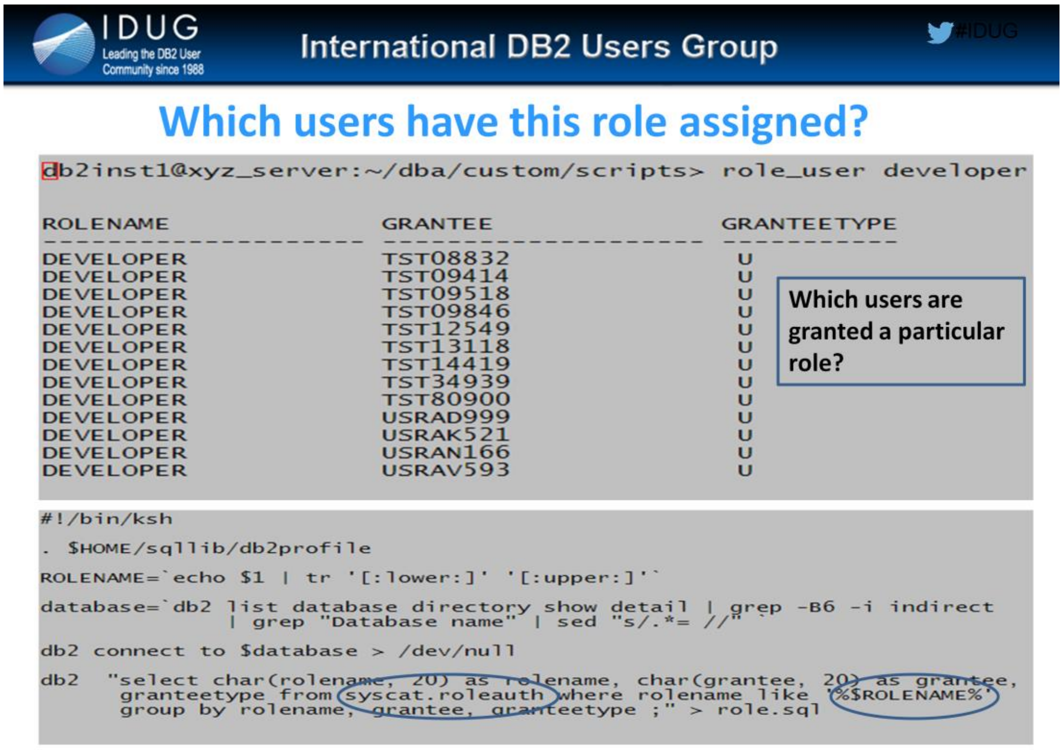
permissions needed

Current privileges on the table

```
db2 connect to $database > /dev/null

db2   "select  char(authid, 20) as auth_id,
           case authidtype
             when 'U' THEN 'USER'
             when 'G' THEN 'GROUP'
             when 'R' THEN 'ROLE'
             END authidtype
, privilege from sysibmadm.privileges
 where objectname = '$TBNAM' and objectschema = '$SCHEMA' and authid <> 'DB2INST1'
 order by  authid, authidtype, privilege " | grep -i -v selected
```

table_priv -> privileges granted on a table

This scripts helps in finding what database roles is a user assigned to. This comes handy to check privileges for users.



International DB2 Users Group

#IDUG

Which users have this role assigned?

```
db2inst1@xyz_server:~/dba/custom/scripts> role_user developer

ROLENAME                  GRANTEE                   GRANTEETYPE
---------------------     --------------------      -----------
DEVELOPER                 TST08832                    U
DEVELOPER                 TST09414                    U
DEVELOPER                 TST09518                    U
DEVELOPER                 TST09846                    U
DEVELOPER                 TST12549                    U
DEVELOPER                 TST13118                    U
DEVELOPER                 TST14419                    U
DEVELOPER                 TST34939                    U
DEVELOPER                 TST80900                    U
DEVELOPER                 USRAD999                    U
DEVELOPER                 USRAK521                    U
DEVELOPER                 USRAN166                    U
DEVELOPER                 USRAV593                    U
```

Which users are granted a particular role?

```
#!/bin/ksh

. $HOME/sqllib/db2profile

ROLENAME=`echo $1 | tr '[:lower:]' '[:upper:]'`

database=`db2 list database directory show detail | grep -B6 -i indirect
              | grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null

db2  "select char(rolename, 20) as rolename, char(grantee, 20) as grantee,
      granteetype from syscat.roleauth where rolename like '%$ROLENAME%'
      group by rolename, grantee, granteetype ;" > role.sql
```

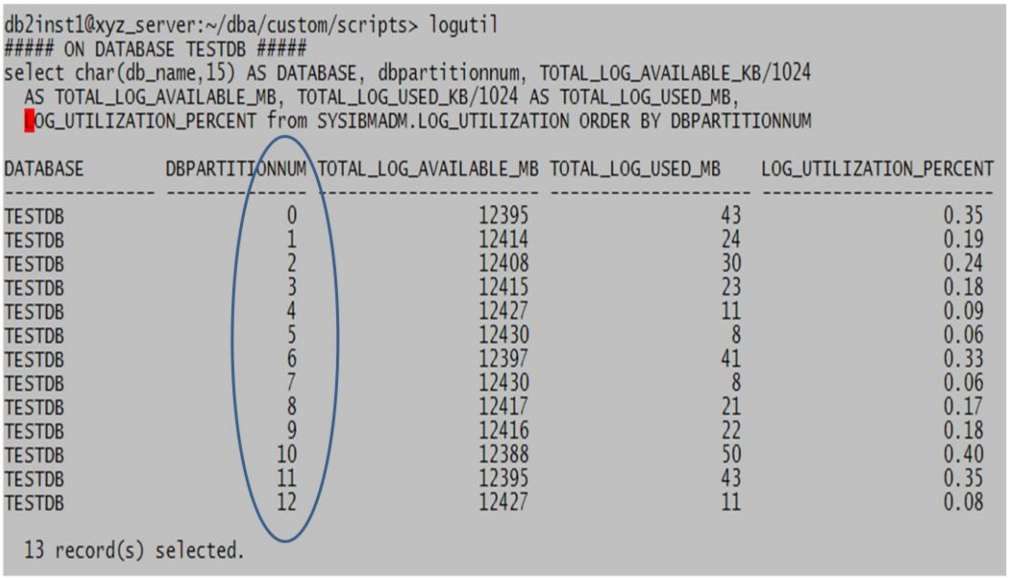
This script lists all the users who are assigned a given role.

IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Active Log Space Utilization



```
db2inst1@xyz_server:~/dba/custom/scripts> logutil
##### ON DATABASE TESTDB #####
select char(db_name,15) AS DATABASE, dbpartitionnum, TOTAL_LOG_AVAILABLE_KB/1024
  AS TOTAL_LOG_AVAILABLE_MB, TOTAL_LOG_USED_KB/1024 AS TOTAL_LOG_USED_MB,
  LOG_UTILIZATION_PERCENT from SYSIBMADM.LOG_UTILIZATION ORDER BY DBPARTITIONNUM

DATABASE        DBPARTITIONNUM TOTAL_LOG_AVAILABLE_MB TOTAL_LOG_USED_MB    LOG_UTILIZATION_PERCENT
--------------- -------------- ---------------------- -------------------- ------------------------
TESTDB                       0                  12395                   43                     0.35
TESTDB                       1                  12414                   24                     0.19
TESTDB                       2                  12408                   30                     0.24
TESTDB                       3                  12415                   23                     0.18
TESTDB                       4                  12427                   11                     0.09
TESTDB                       5                  12430                    8                     0.06
TESTDB                       6                  12397                   41                     0.33
TESTDB                       7                  12430                    8                     0.06
TESTDB                       8                  12417                   21                     0.17
TESTDB                       9                  12416                   22                     0.18
TESTDB                      10                  12388                   50                     0.40
TESTDB                      11                  12395                   43                     0.35
TESTDB                      12                  12427                   11                     0.08

  13 record(s) selected.
```

This script, gives the snap shot of logspace utilization by each partition at the point of time.

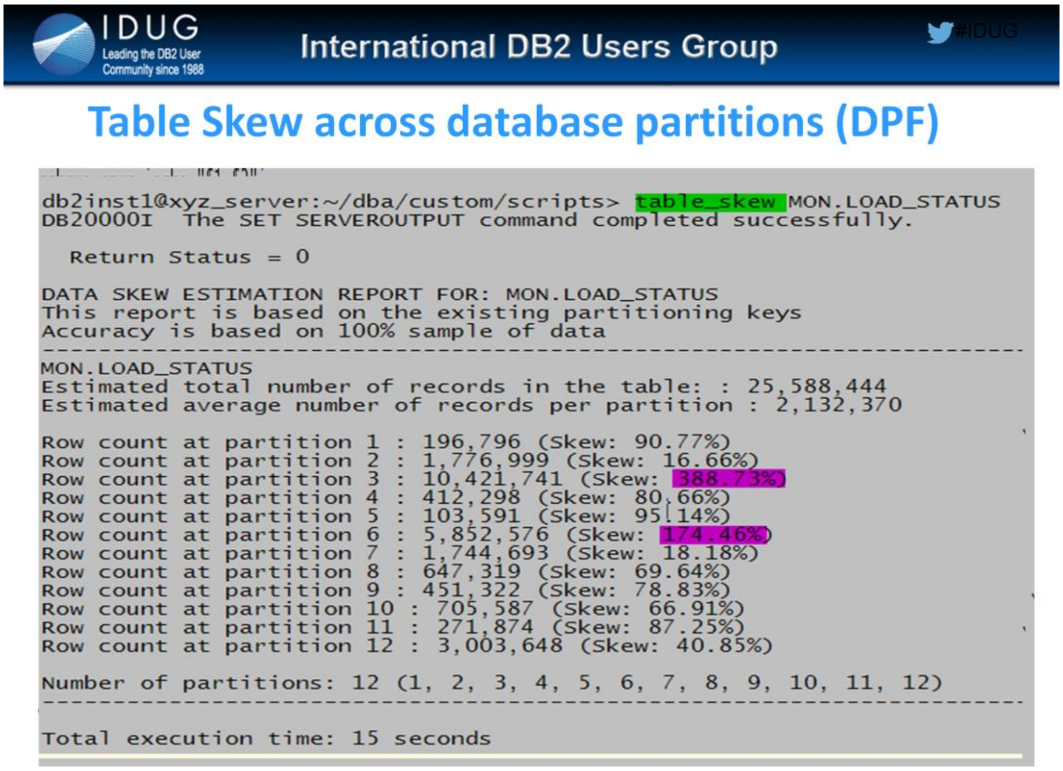

IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

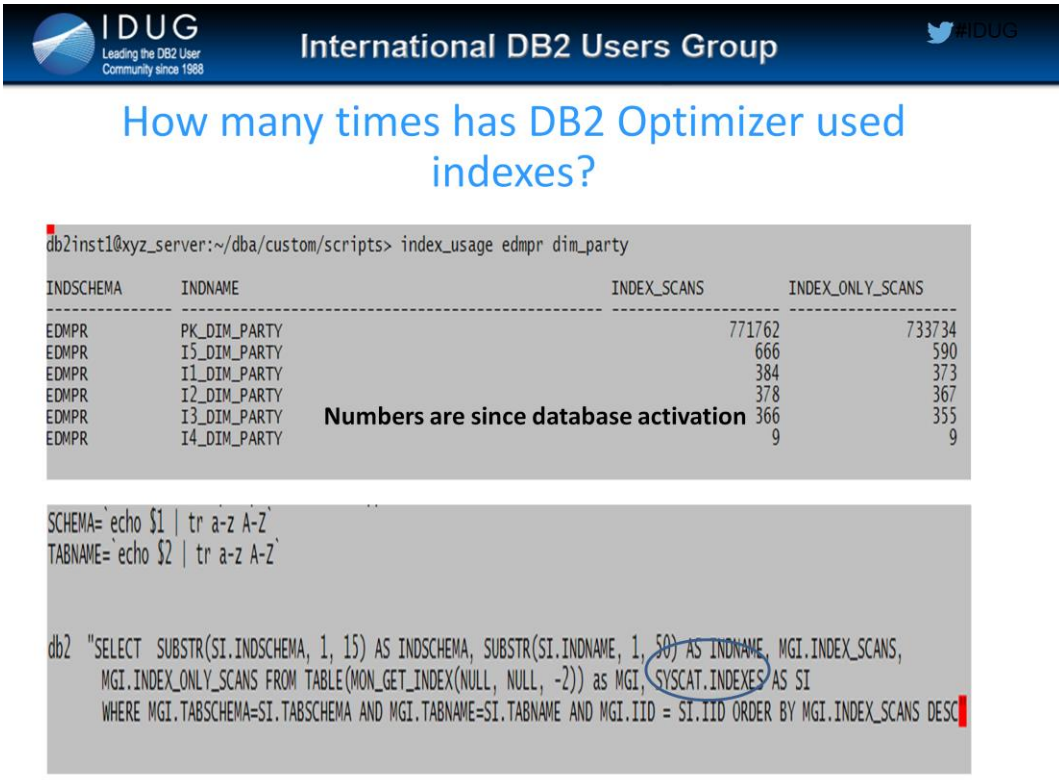
Table Skew across database partitions (DPF)

```
db2inst1@xyz_server:~/dba/custom/scripts> table_skew MON.LOAD_STATUS
DB20000I  The SET SERVEROUTPUT command completed successfully.

  Return Status = 0

DATA SKEW ESTIMATION REPORT FOR: MON.LOAD_STATUS
This report is based on the existing partitioning keys
Accuracy is based on 100% sample of data
----------------------------------------------------------------------
MON.LOAD_STATUS
Estimated total number of records in the table: : 25,588,444
Estimated average number of records per partition : 2,132,370

Row count at partition 1 : 196,796 (Skew: 90.77%)
Row count at partition 2 : 1,776,999 (Skew: 16.66%)
Row count at partition 3 : 10,421,741 (Skew: 388.73%)
Row count at partition 4 : 412,298 (Skew: 80.66%)
Row count at partition 5 : 103,591 (Skew: 95.14%)
Row count at partition 6 : 5,852,576 (Skew: 174.46%)
Row count at partition 7 : 1,744,693 (Skew: 18.18%)
Row count at partition 8 : 647,319 (Skew: 69.64%)
Row count at partition 9 : 451,322 (Skew: 78.83%)
Row count at partition 10 : 705,587 (Skew: 66.91%)
Row count at partition 11 : 271,874 (Skew: 87.25%)
Row count at partition 12 : 3,003,648 (Skew: 40.85%)

Number of partitions: 12 (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
----------------------------------------------------------------------

Total execution time: 15 seconds
```

This script, gives the data skew in a table over all the partitions .

URL to download stored procedure:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1005partitioningkeys/


IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

How many times has DB2 Optimizer used indexes?

```
db2inst1@xyz_server:~/dba/custom/scripts> index_usage edmpr dim_party

INDSCHEMA       INDNAME                                             INDEX_SCANS          INDEX_ONLY_SCANS
--------------- -------------------------------------------------- -------------------- --------------------
EDMPR           PK_DIM_PARTY                                                     771762               733734
EDMPR           I5_DIM_PARTY                                                        666                  590
EDMPR           I1_DIM_PARTY                                                        384                  373
EDMPR           I2_DIM_PARTY                                                        378                  367
EDMPR           I3_DIM_PARTY                                                        366                  355
EDMPR           I4_DIM_PARTY                                                          9                    9
```

Numbers are since database activation

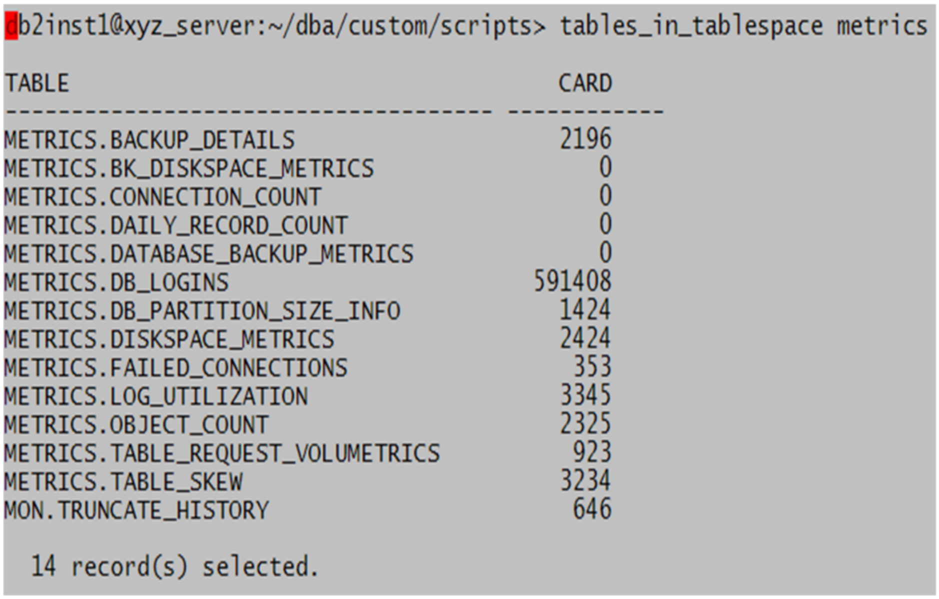
```
SCHEMA=`echo $1 | tr a-z A-Z`
TABNAME=`echo $2 | tr a-z A-Z`


db2  "SELECT  SUBSTR(SI.INDSCHEMA, 1, 15) AS INDSCHEMA, SUBSTR(SI.INDNAME, 1, 50) AS INDNAME, MGI.INDEX_SCANS,
      MGI.INDEX_ONLY_SCANS FROM TABLE(MON_GET_INDEX(NULL, NULL, -2)) as MGI, SYSCAT.INDEXES AS SI
      WHERE MGI.TABSCHEMA=SI.TABSCHEMA AND MGI.TABNAME=SI.TABNAME AND MGI.IID = SI.IID ORDER BY MGI.INDEX_SCANS DESC
```

This script lists how many times an index is scanned since the last time that instance started.

If no argument is passed, then the script lists # of index_scans for all tables in the database.
Above example shows a scenarios where index_usage is used for a specific table.

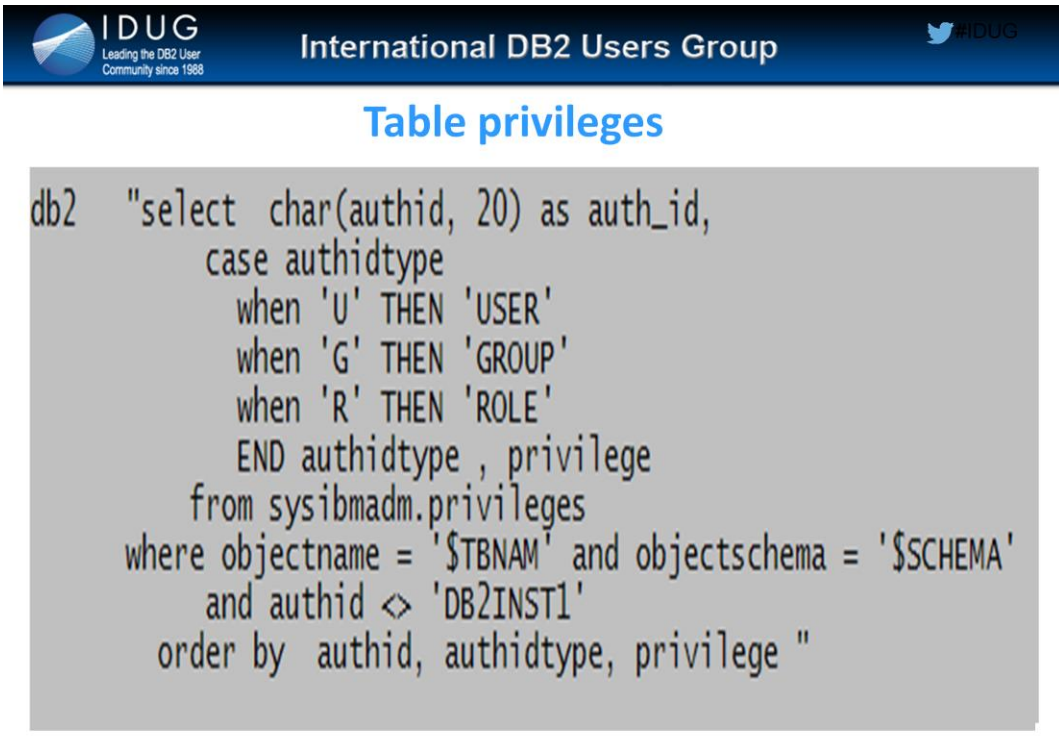
International DB2 Users Group

Find Tables in a Tablespace



```
db2inst1@xyz_server:~/dba/custom/scripts> tables_in_tablespace metrics

TABLE                                         CARD
------------------------------------- ------------
METRICS.BACKUP_DETAILS                        2196
METRICS.BK_DISKSPACE_METRICS                     0
METRICS.CONNECTION_COUNT                         0
METRICS.DAILY_RECORD_COUNT                       0
METRICS.DATABASE_BACKUP_METRICS                  0
METRICS.DB_LOGINS                           591408
METRICS.DB_PARTITION_SIZE_INFO                1424
METRICS.DISKSPACE_METRICS                     2424
METRICS.FAILED_CONNECTIONS                     353
METRICS.LOG_UTILIZATION                       3345
METRICS.OBJECT_COUNT                          2325
METRICS.TABLE_REQUEST_VOLUMETRICS              923
METRICS.TABLE_SKEW                            3234
MON.TRUNCATE_HISTORY                           646

  14 record(s) selected.
```

As the name suggests, this script lists all the tables a tablespace consists.


IDUG
Leading the DB2 User Community since 1988

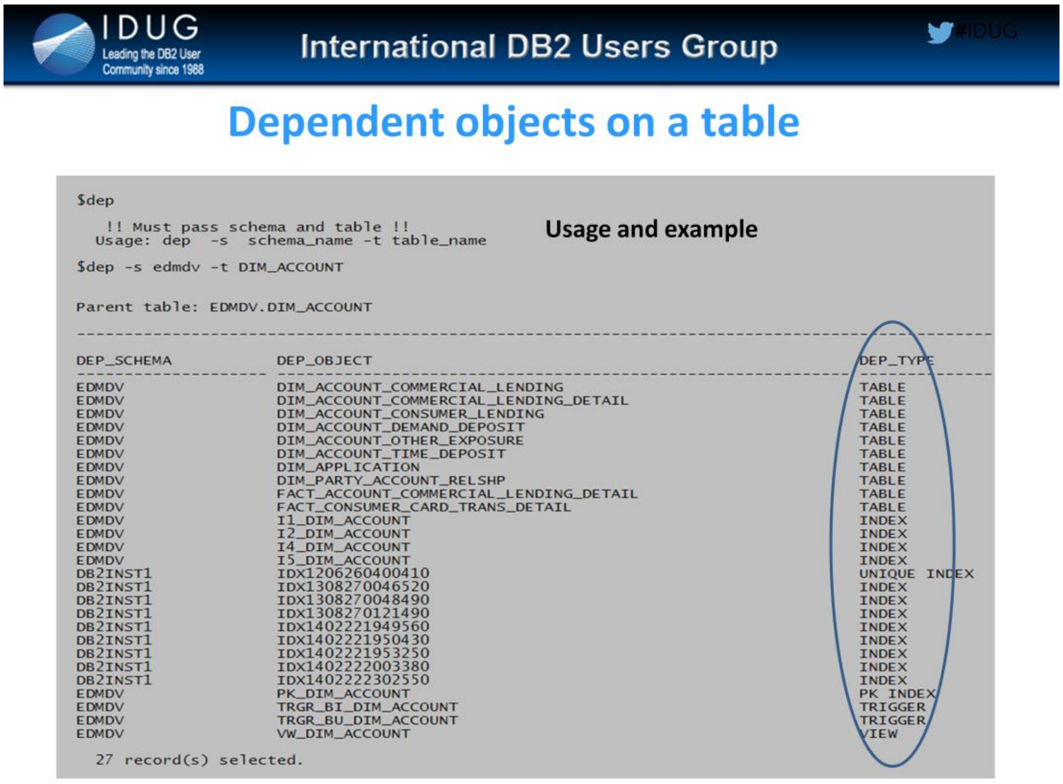
International DB2 Users Group

#IDUG

Table privileges

```
db2  "select  char(authid, 20) as auth_id,
          case authidtype
            when 'U' THEN 'USER'
            when 'G' THEN 'GROUP'
            when 'R' THEN 'ROLE'
            END authidtype , privilege
         from sysibmadm.privileges
      where objectname = '$TBNAM' and objectschema = '$SCHEMA'
          and authid <> 'DB2INST1'
        order by  authid, authidtype, privilege "
```

Simple script that lists privileges that users hold for a given table.


International DB2 Users Group

#IDUG

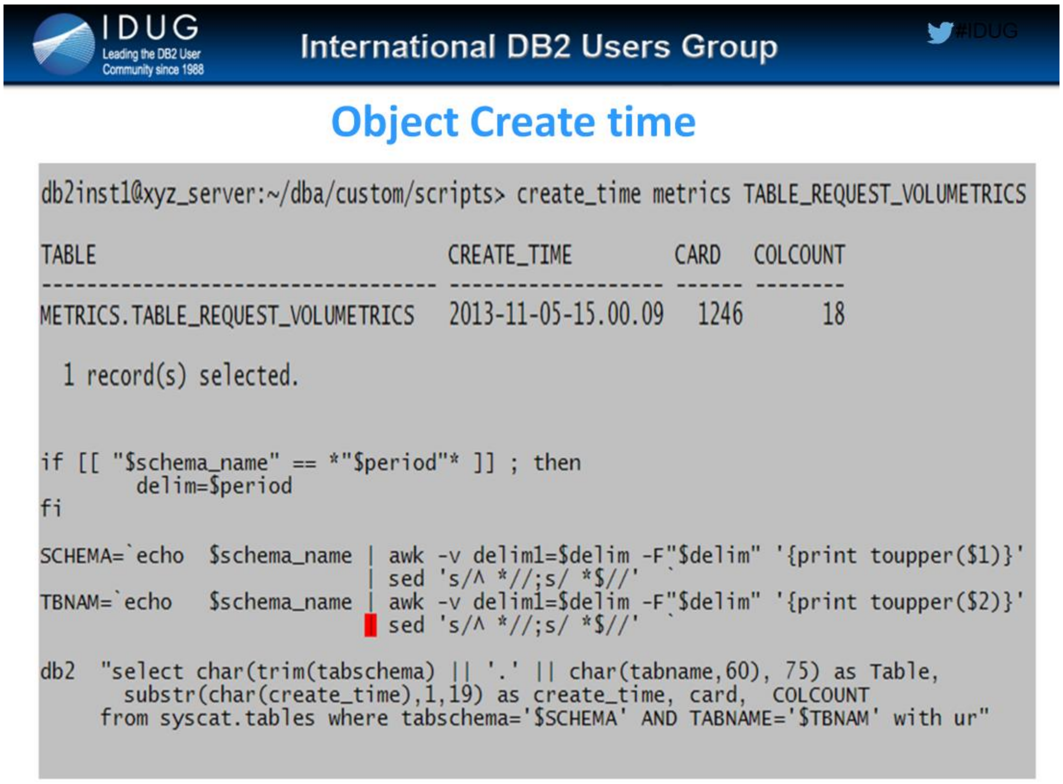
Dependent objects on a table

Usage and example

```
$dep

   !! Must pass schema and table !!
  Usage: dep   -s  schema_name -t table_name

$dep -s edmdv -t DIM_ACCOUNT


Parent table: EDMDV.DIM_ACCOUNT

------------------------------------------------------------------------------------------------

DEP_SCHEMA            DEP_OBJECT                                                  DEP_TYPE
--------------------  ----------------------------------------------------------  --------------
EDMDV                 DIM_ACCOUNT_COMMERCIAL_LENDING                              TABLE
EDMDV                 DIM_ACCOUNT_COMMERCIAL_LENDING_DETAIL                       TABLE
EDMDV                 DIM_ACCOUNT_CONSUMER_LENDING                                TABLE
EDMDV                 DIM_ACCOUNT_DEMAND_DEPOSIT                                  TABLE
EDMDV                 DIM_ACCOUNT_OTHER_EXPOSURE                                  TABLE
EDMDV                 DIM_ACCOUNT_TIME_DEPOSIT                                    TABLE
EDMDV                 DIM_APPLICATION                                             TABLE
EDMDV                 DIM_PARTY_ACCOUNT_RELSHP                                    TABLE
EDMDV                 FACT_ACCOUNT_COMMERCIAL_LENDING_DETAIL                      TABLE
EDMDV                 FACT_CONSUMER_CARD_TRANS_DETAIL                             TABLE
EDMDV                 I1_DIM_ACCOUNT                                              INDEX
EDMDV                 I2_DIM_ACCOUNT                                              INDEX
EDMDV                 I4_DIM_ACCOUNT                                              INDEX
EDMDV                 I5_DIM_ACCOUNT                                              INDEX
DB2INST1              IDX1206260400410                                            UNIQUE INDEX
DB2INST1              IDX1308270046520                                            INDEX
DB2INST1              IDX1308270048490                                            INDEX
DB2INST1              IDX1308270121490                                            INDEX
DB2INST1              IDX1402221949560                                            INDEX
DB2INST1              IDX1402221950430                                            INDEX
DB2INST1              IDX1402221953250                                            INDEX
DB2INST1              IDX1402222003380                                            INDEX
DB2INST1              IDX1402222302550                                            INDEX
EDMDV                 PK_DIM_ACCOUNT                                              PK INDEX
EDMDV                 TRGR_BI_DIM_ACCOUNT                                         TRIGGER
EDMDV                 TRGR_BU_DIM_ACCOUNT                                         TRIGGER
EDMDV                 VW_DIM_ACCOUNT                                              VIEW

  27 record(s) selected.
```

This script lists different types of a dependant objects like child tables, indexes, views, triggers

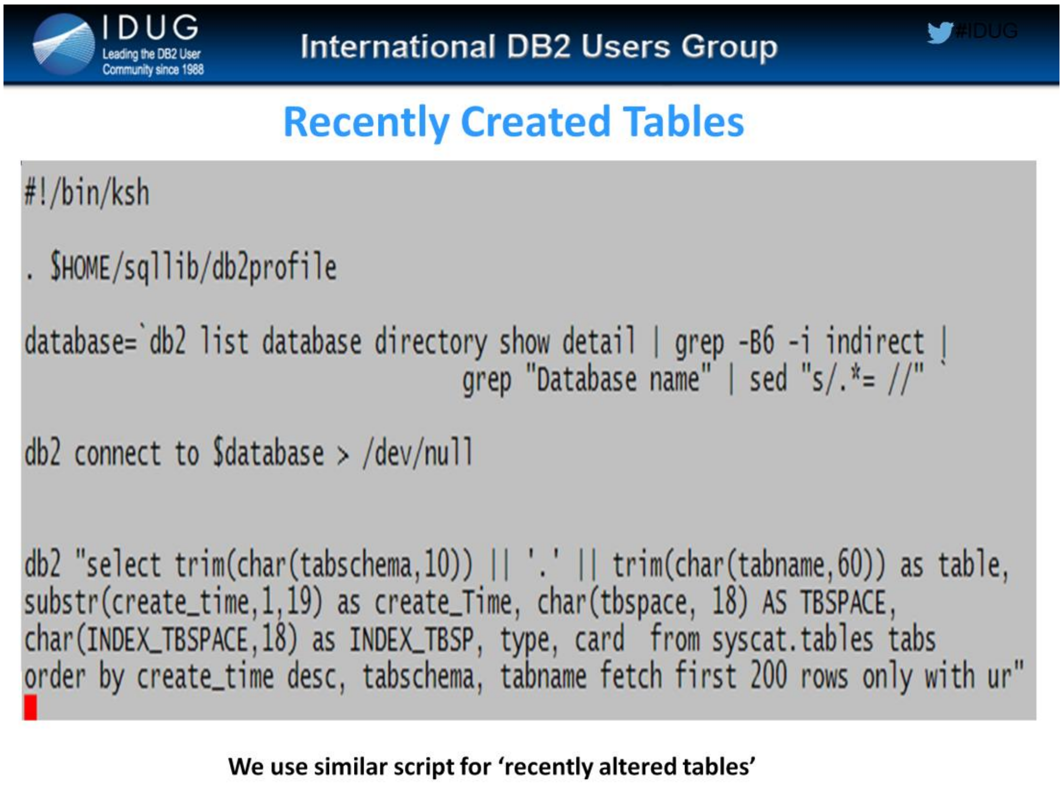
This script is a lengthy one. If you need source code for this one, feel free to email.



Object Create time

```
db2inst1@xyz_server:~/dba/custom/scripts> create_time metrics TABLE_REQUEST_VOLUMETRICS

TABLE                              CREATE_TIME         CARD   COLCOUNT
---------------------------------- ------------------- ------ --------
METRICS.TABLE_REQUEST_VOLUMETRICS  2013-11-05-15.00.09   1246       18

  1 record(s) selected.


if [[ "$schema_name" == *"$period"* ]] ; then
        delim=$period
fi

SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}'
                           | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}'
                           | sed 's/^ *//;s/ *$//'  `

db2  "select char(trim(tabschema) || '.' || char(tabname,60), 75) as Table,
       substr(char(create_time),1,19) as create_time, card,  COLCOUNT
     from syscat.tables where tabschema='$SCHEMA' AND TABNAME='$TBNAM' with ur"
```

This script gives some key properties of a table / view like **create_time**, number of columns in it and cardinality.

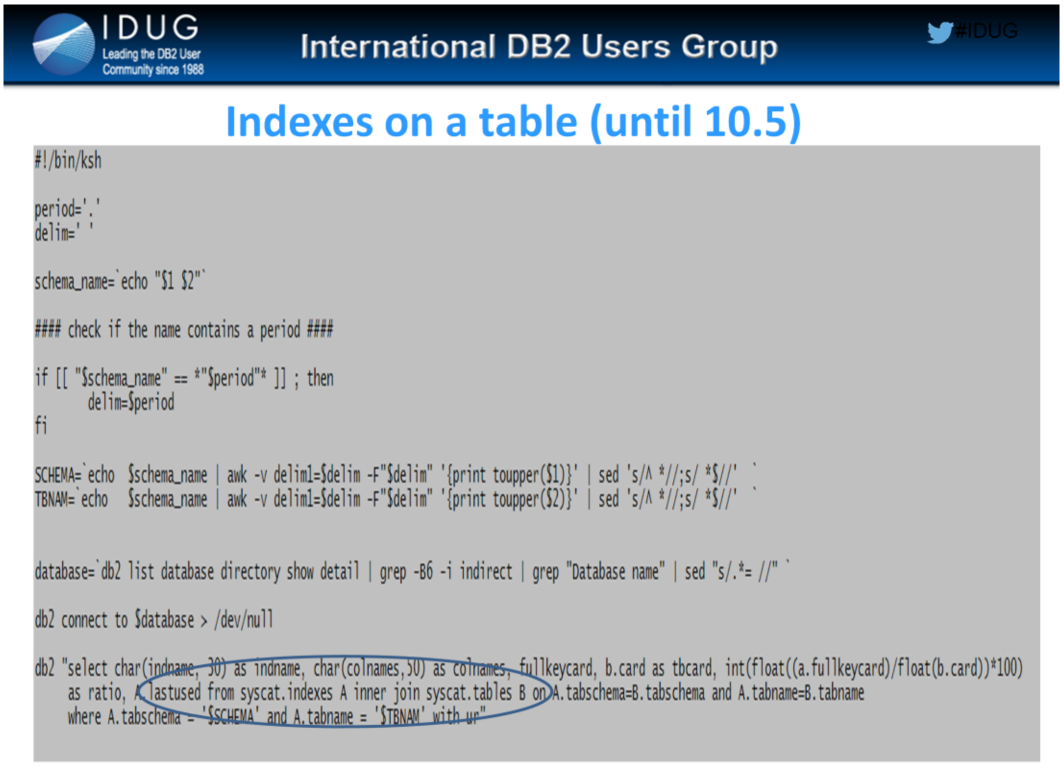

IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Recently Created Tables

```
#!/bin/ksh

. $HOME/sqllib/db2profile

database=`db2 list database directory show detail | grep -B6 -i indirect |
                                   grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null


db2 "select trim(char(tabschema,10)) || '.' || trim(char(tabname,60)) as table,
substr(create_time,1,19) as create_Time, char(tbspace, 18) AS TBSPACE,
char(INDEX_TBSPACE,18) as INDEX_TBSP, type, card  from syscat.tables tabs
order by create_time desc, tabschema, tabname fetch first 200 rows only with ur"
```

We use similar script for 'recently altered tables'

As the name suggests, this script lists the recently created tables in the database ordered by create_time descending.


IDUG Leading the DB2 User Community since 1988

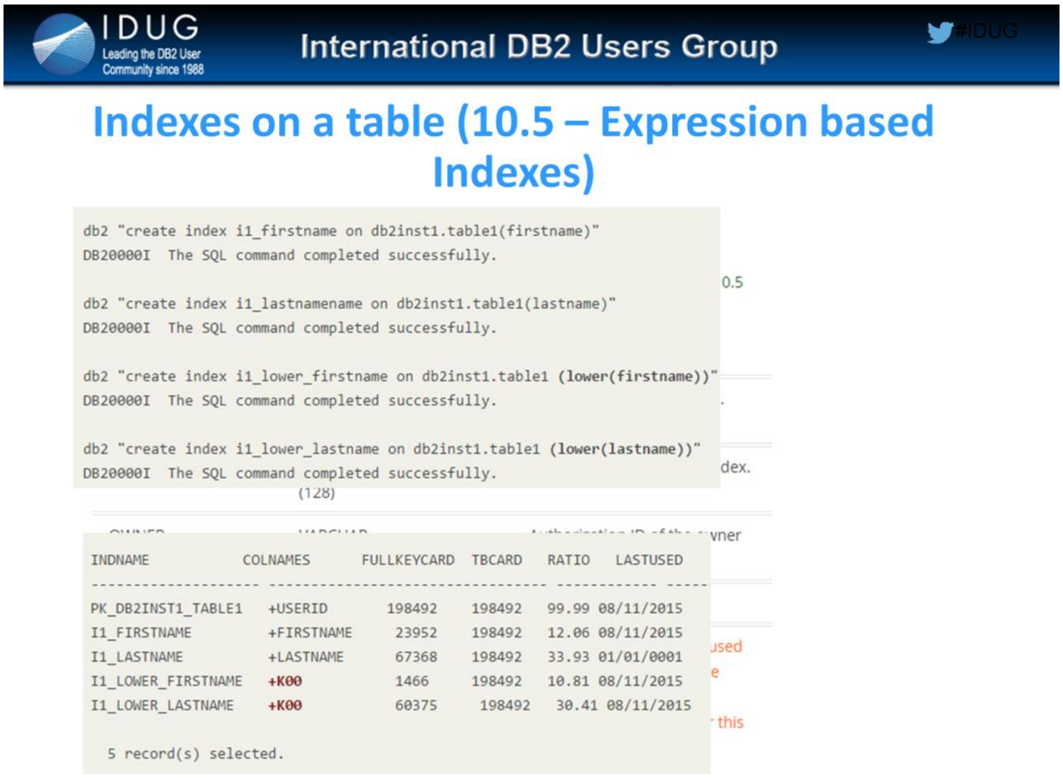
International DB2 Users Group

#IDUG

Indexes on a table (until 10.5)

```
#!/bin/ksh

period='.'
delim=' '

schema_name=`echo "$1 $2"`

#### check if the name contains a period ####

if [[ "$schema_name" == *"$period"* ]] ; then
        delim=$period
fi

SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `


database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null

db2 "select char(indname, 30) as indname, char(colnames,50) as colnames, fullkeycard, b.card as tbcard, int(float((a.fullkeycard)/float(b.card))*100)
     as ratio, A.lastused from syscat.indexes A inner join syscat.tables B on A.tabschema=B.tabschema and A.tabname=B.tabname
     where A.tabschema = '$SCHEMA' and A.tabname = '$TBNAM' with ur"
```

This script lists the existing indexes on a table with indexed column list, cardinality and index last used date. It comes in very handy, when you are in middle of troubleshooting a performance issue.



IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

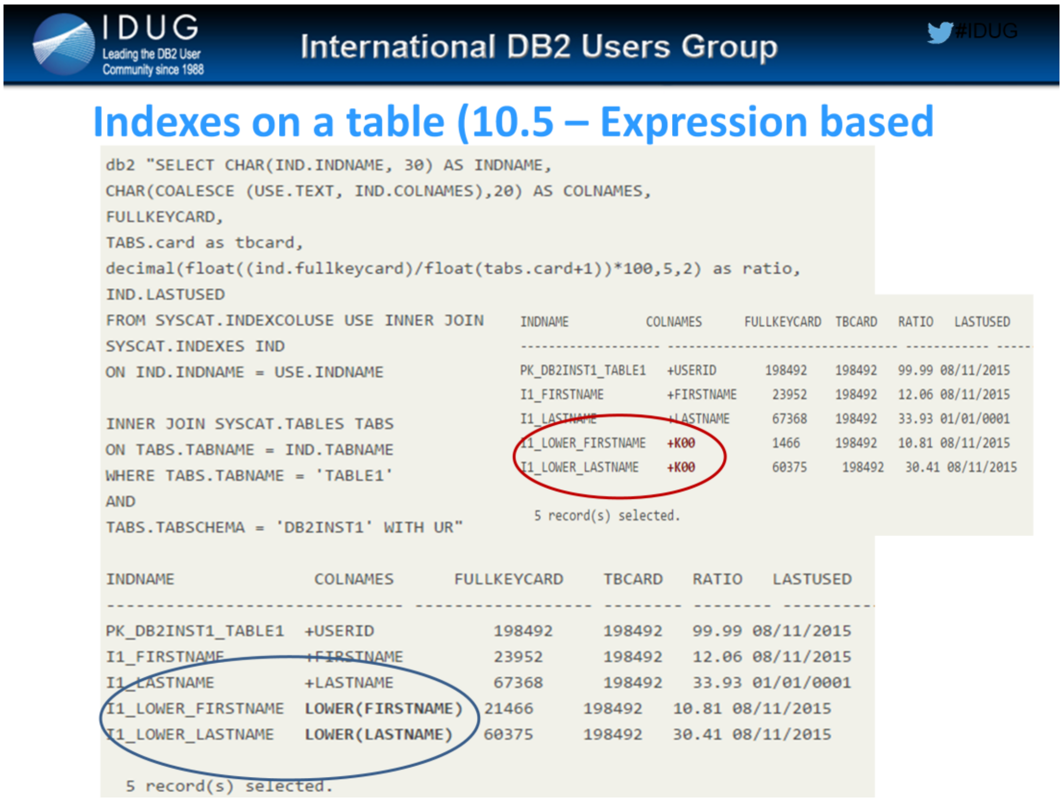
#IDUG

Indexes on a table (10.5 – Expression based Indexes)

```
db2 "create index i1_firstname on db2inst1.table1(firstname)"
DB20000I  The SQL command completed successfully.

db2 "create index i1_lastnamename on db2inst1.table1(lastname)"
DB20000I  The SQL command completed successfully.

db2 "create index i1_lower_firstname on db2inst1.table1 (lower(firstname))"
DB20000I  The SQL command completed successfully.

db2 "create index i1_lower_lastname on db2inst1.table1 (lower(lastname))"
DB20000I  The SQL command completed successfully.
```

```
INDNAME             COLNAMES         FULLKEYCARD  TBCARD   RATIO  LASTUSED
------------------- -------------------------------- ------------ -----
PK_DB2INST1_TABLE1   +USERID           198492     198492   99.99 08/11/2015
I1_FIRSTNAME         +FIRSTNAME         23952     198492   12.06 08/11/2015
I1_LASTNAME          +LASTNAME          67368     198492   33.93 01/01/0001
I1_LOWER_FIRSTNAME   +K00                1466     198492   10.81 08/11/2015
I1_LOWER_LASTNAME    +K00               60375      198492   30.41 08/11/2015

  5 record(s) selected.
```

This script lists the existing indexes on a table with indexed column list, cardinality and index last used date. It comes in very handy, when you are in middle of troubleshooting a performance issue.



IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

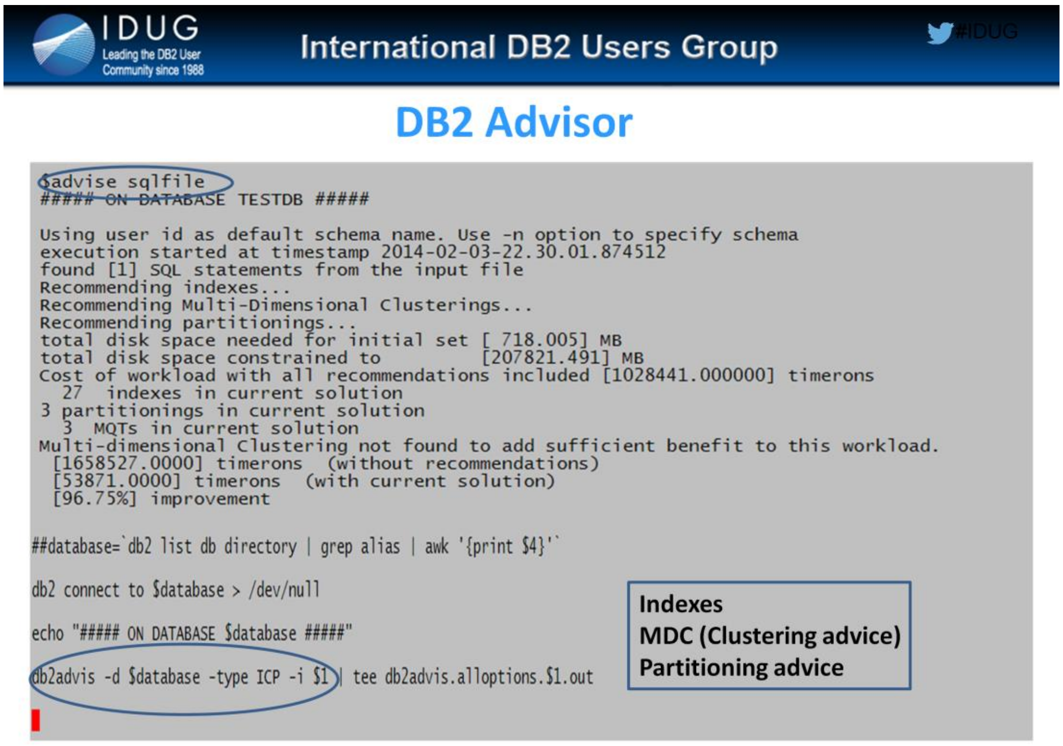
Indexes on a table (10.5 – Expression based

```
db2 "SELECT CHAR(IND.INDNAME, 30) AS INDNAME,
CHAR(COALESCE (USE.TEXT, IND.COLNAMES),20) AS COLNAMES,
FULLKEYCARD,
TABS.card as tbcard,
decimal(float((ind.fullkeycard)/float(tabs.card+1))*100,5,2) as ratio,
IND.LASTUSED
FROM SYSCAT.INDEXCOLUSE USE INNER JOIN
SYSCAT.INDEXES IND
ON IND.INDNAME = USE.INDNAME

INNER JOIN SYSCAT.TABLES TABS
ON TABS.TABNAME = IND.TABNAME
WHERE TABS.TABNAME = 'TABLE1'
AND
TABS.TABSCHEMA = 'DB2INST1' WITH UR"
```

```
INDNAME            COLNAMES        FULLKEYCARD  TBCARD   RATIO   LASTUSED
------------------ ------------------------------ ----------- -----
PK_DB2INST1_TABLE1   +USERID          198492    198492   99.99 08/11/2015
I1_FIRSTNAME         +FIRSTNAME        23952    198492   12.06 08/11/2015
I1_LASTNAME          +LASTNAME         67368    198492   33.93 01/01/0001
I1_LOWER_FIRSTNAME   +K00              1466     198492   10.81 08/11/2015
I1_LOWER_LASTNAME    +K00              60375     198492   30.41 08/11/2015

  5 record(s) selected.
```

```
INDNAME                  COLNAMES         FULLKEYCARD     TBCARD    RATIO    LASTUSED
------------------------------ ------------------ -------- -------- ---------
PK_DB2INST1_TABLE1  +USERID              198492     198492   99.99 08/11/2015
I1_FIRSTNAME        +FIRSTNAME            23952     198492   12.06 08/11/2015
I1_LASTNAME         +LASTNAME             67368     198492   33.93 01/01/0001
I1_LOWER_FIRSTNAME  LOWER(FIRSTNAME) 21466     198492   10.81 08/11/2015
I1_LOWER_LASTNAME   LOWER(LASTNAME)  60375     198492   30.41 08/11/2015

  5 record(s) selected.
```

This script lists the existing indexes on a table with indexed column list, cardinality and index last used date. It comes in very handy, when you are in middle of troubleshooting a performance issue.



IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

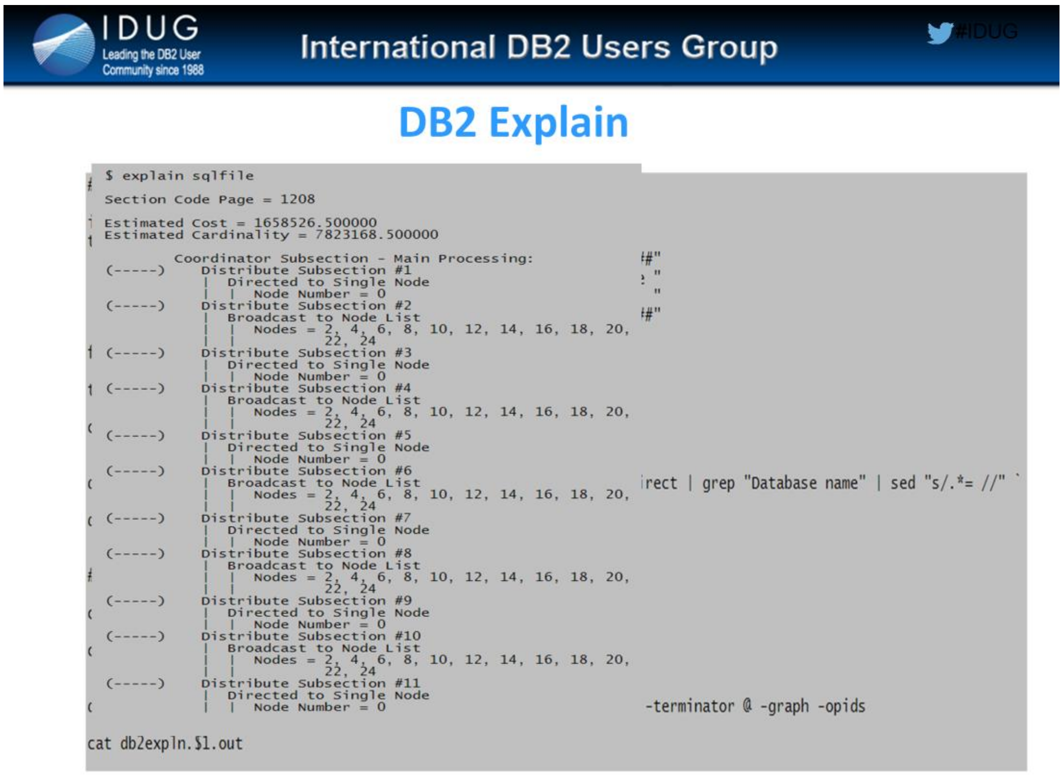
DB2 Advisor

```
$advise sqlfile
##### ON DATABASE TESTDB #####

Using user id as default schema name. Use -n option to specify schema
execution started at timestamp 2014-02-03-22.30.01.874512
found [1] SQL statements from the input file
Recommending indexes...
Recommending Multi-Dimensional Clusterings...
Recommending partitionings...
total disk space needed for initial set [ 718.005] MB
total disk space constrained to         [207821.491] MB
Cost of workload with all recommendations included [1028441.000000] timerons
  27  indexes in current solution
3 partitionings in current solution
  3  MQTs in current solution
Multi-dimensional Clustering not found to add sufficient benefit to this workload.
 [1658527.0000] timerons  (without recommendations)
 [53871.0000] timerons  (with current solution)
 [96.75%] improvement
```

```
##database=`db2 list db directory | grep alias | awk '{print $4}'`

db2 connect to $database > /dev/null

echo "##### ON DATABASE $database #####"

db2advis -d $database -type ICP -i $1 | tee db2advis.alloptions.$1.out
```

Indexes
MDC (Clustering advice)
Partitioning advice

This script takes a file with sql statemenet(s) and issues db2advis command for IBM’s Design Advisor recommendations on indexes, repartitioning of tables, MQTs and MDCs.


International DB2 Users Group

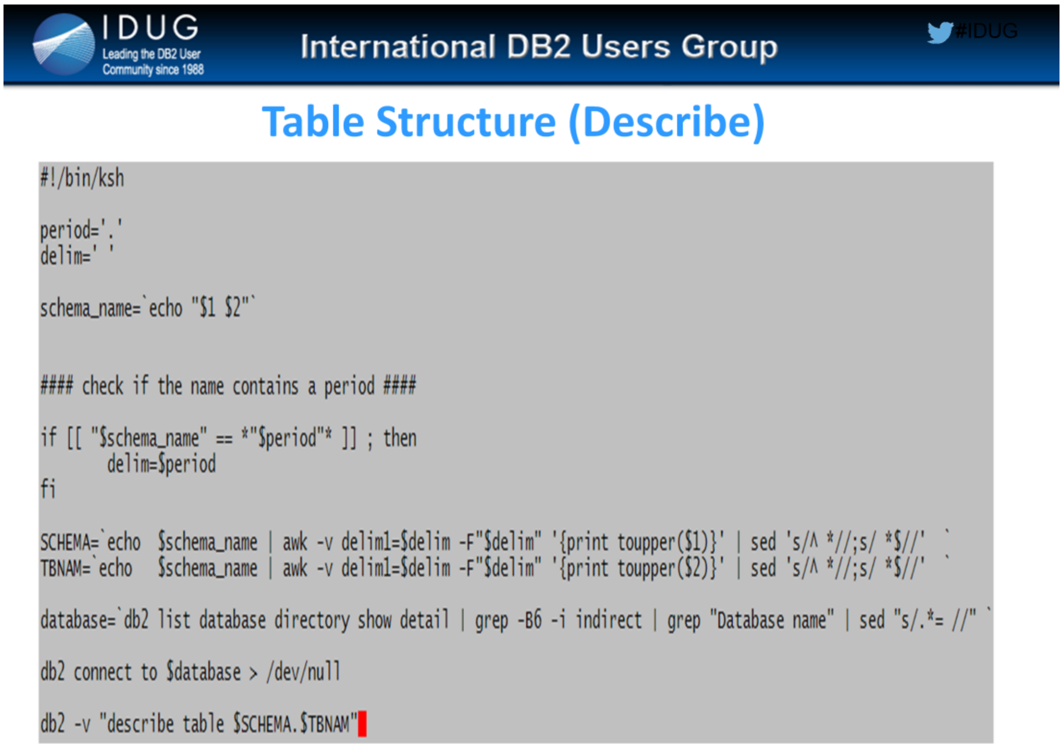
#IDUG

DB2 Explain

```
$ explain sqlfile

Section Code Page = 1208

Estimated Cost = 1658526.500000
Estimated Cardinality = 7823168.500000

          Coordinator Subsection - Main Processing:
(-----)       Distribute Subsection #1
              |   Directed to Single Node
              |   |   Node Number = 0
(-----)       Distribute Subsection #2
              |   Broadcast to Node List
              |   |   Nodes = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
              |   |           22, 24
(-----)       Distribute Subsection #3
              |   Directed to Single Node
              |   |   Node Number = 0
(-----)       Distribute Subsection #4
              |   Broadcast to Node List
              |   |   Nodes = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
              |   |           22, 24
(-----)       Distribute Subsection #5
              |   Directed to Single Node
              |   |   Node Number = 0
(-----)       Distribute Subsection #6
              |   Broadcast to Node List
              |   |   Nodes = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
              |   |           22, 24
(-----)       Distribute Subsection #7
              |   Directed to Single Node
              |   |   Node Number = 0
(-----)       Distribute Subsection #8
              |   Broadcast to Node List
              |   |   Nodes = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
              |   |           22, 24
(-----)       Distribute Subsection #9
              |   Directed to Single Node
              |   |   Node Number = 0
(-----)       Distribute Subsection #10
              |   Broadcast to Node List
              |   |   Nodes = 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
              |   |           22, 24
(-----)       Distribute Subsection #11
              |   Directed to Single Node
              |   |   Node Number = 0
```

```
#"
"
"
#"
irect | grep "Database name" | sed "s/.*= //" `
-terminator @ -graph -opids
cat db2expln.$1.out
```

This script takes a file with SQL statement(s) and prints explain plan for the statement.



IDUG Leading the DB2 User Community since 1988

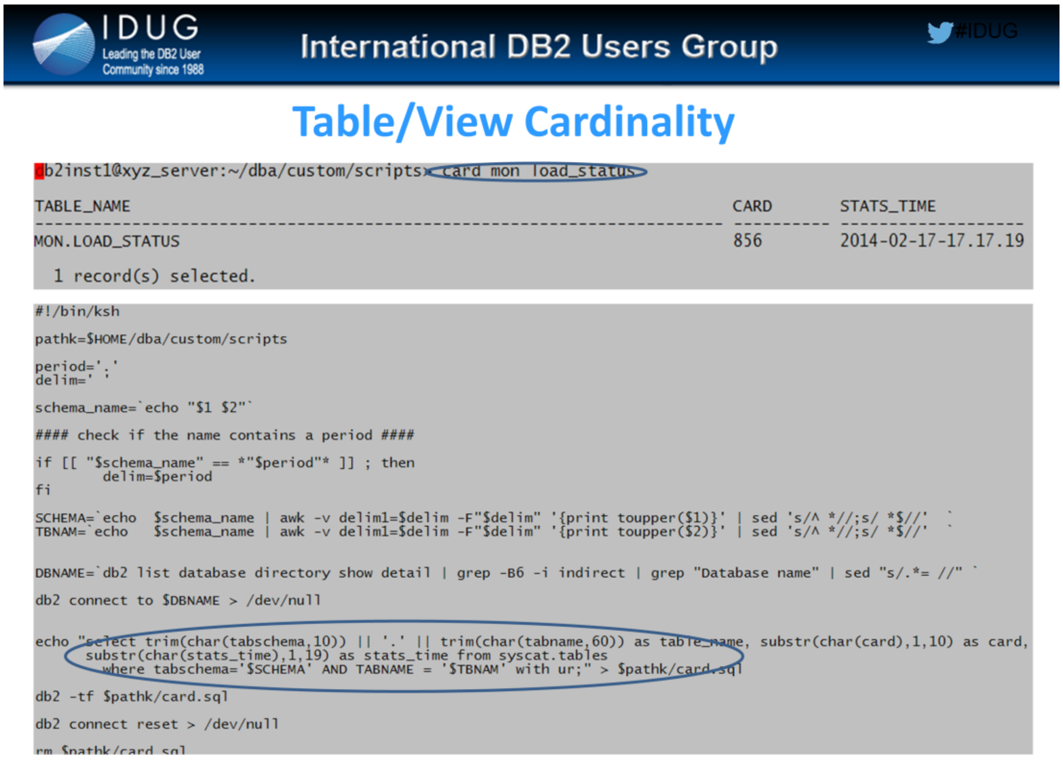
International DB2 Users Group

#IDUG

Table Structure (Describe)

```
#!/bin/ksh

period='.'
delim=' '

schema_name=`echo "$1 $2"`


#### check if the name contains a period ####

if [[ "$schema_name" == *"$period"* ]] ; then
       delim=$period
fi

SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `

database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null

db2 -v "describe table $SCHEMA.$TBNAM"
```

This script lists the column properties (column name, data types, length, null or not-null etc) in a table.



International DB2 Users Group

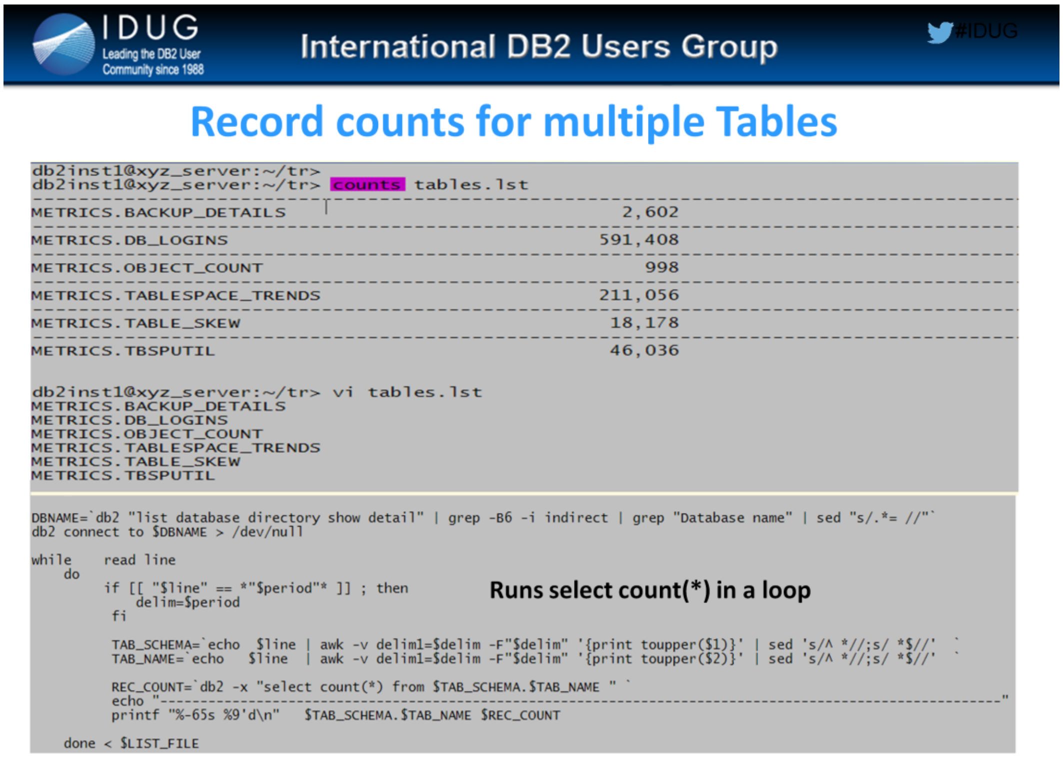
#IDUG

Table/View Cardinality

```
db2inst1@xyz_server:~/dba/custom/scripts> card mon load_status

TABLE_NAME                                                                CARD       STATS_TIME
------------------------------------------------------------------------- ---------- -------------------
MON.LOAD_STATUS                                                           856        2014-02-17-17.17.19

  1 record(s) selected.
```

```
#!/bin/ksh

pathk=$HOME/dba/custom/scripts

period='.'
delim=' '

schema_name=`echo "$1 $2"`

#### check if the name contains a period ####

if [[ "$schema_name" == *"$period"* ]] ; then
        delim=$period
fi

SCHEMA=`echo  $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
TBNAM=`echo   $schema_name | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `


DBNAME=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `

db2 connect to $DBNAME > /dev/null


echo "select trim(char(tabschema,10)) || '.' || trim(char(tabname,60)) as table_name, substr(char(card),1,10) as card,
      substr(char(stats_time),1,19) as stats_time from syscat.tables
        where tabschema='$SCHEMA' AND TABNAME = '$TBNAM' with ur;" > $pathk/card.sql

db2 -tf $pathk/card.sql

db2 connect reset > /dev/null

rm $pathk/card.sql
```

This script gives 'cardinality' information of an object.


IDUG
Leading the DB2 User Community since 1988
International DB2 Users Group
#IDUG

Record counts for multiple Tables

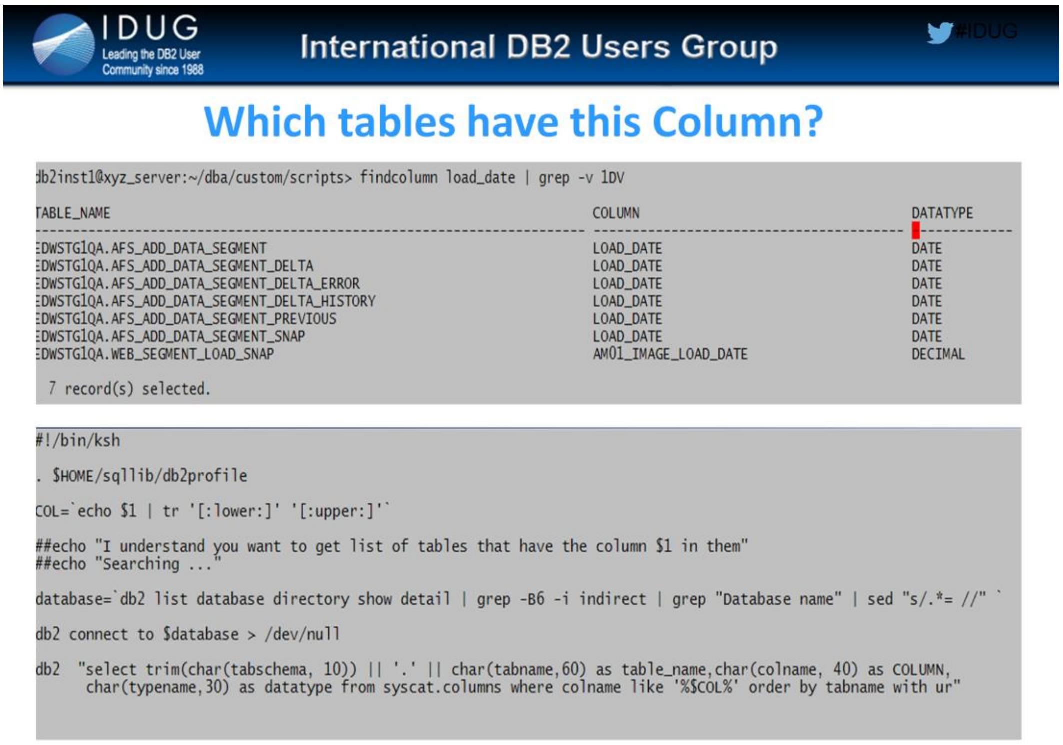
```
db2inst1@xyz_server:~/tr>
db2inst1@xyz_server:~/tr> counts tables.lst
------------------------------------------------------------------------------------------
METRICS.BACKUP_DETAILS                                                  2,602
------------------------------------------------------------------------------------------
METRICS.DB_LOGINS                                                     591,408
------------------------------------------------------------------------------------------
METRICS.OBJECT_COUNT                                                      998
------------------------------------------------------------------------------------------
METRICS.TABLESPACE_TRENDS                                             211,056
------------------------------------------------------------------------------------------
METRICS.TABLE_SKEW                                                     18,178
------------------------------------------------------------------------------------------
METRICS.TBSPUTIL                                                       46,036

db2inst1@xyz_server:~/tr> vi tables.lst
METRICS.BACKUP_DETAILS
METRICS.DB_LOGINS
METRICS.OBJECT_COUNT
METRICS.TABLESPACE_TRENDS
METRICS.TABLE_SKEW
METRICS.TBSPUTIL
```

```
DBNAME=`db2 "list database directory show detail" | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //"`
db2 connect to $DBNAME > /dev/null

while    read line
    do
         if [[ "$line" == *"$period"* ]] ; then
             delim=$period
          fi

          TAB_SCHEMA=`echo  $line | awk -v delim1=$delim -F"$delim" '{print toupper($1)}' | sed 's/^ *//;s/ *$//'  `
          TAB_NAME=`echo   $line  | awk -v delim1=$delim -F"$delim" '{print toupper($2)}' | sed 's/^ *//;s/ *$//'  `

          REC_COUNT=`db2 -x "select count(*) from $TAB_SCHEMA.$TAB_NAME " `
          echo "-----------------------------------------------------------------------------------------------"
          printf "%-65s %9'd\n"   $TAB_SCHEMA.$TAB_NAME  $REC_COUNT

    done < $LIST_FILE
```

Runs select count(*) in a loop

This script gives the quick record counts multiple tables. Needs to pass tables list as argument. Table names should be in one of the following format:

METRICS.OBJECT_COUNT ($schema.table) or
METRICS OBJECT_COUNT ($schema table)



IDUG Leading the DB2 User Community since 1988

International DB2 Users Group

#IDUG

Which tables have this Column?

```
db2inst1@xyz_server:~/dba/custom/scripts> findcolumn load_date | grep -v 1DV

TABLE_NAME                                                        COLUMN                                   DATATYPE
----------------------------------------------------------------- ---------------------------------------- ------------
EDWSTG1QA.AFS_ADD_DATA_SEGMENT                                    LOAD_DATE                                DATE
EDWSTG1QA.AFS_ADD_DATA_SEGMENT_DELTA                              LOAD_DATE                                DATE
EDWSTG1QA.AFS_ADD_DATA_SEGMENT_DELTA_ERROR                        LOAD_DATE                                DATE
EDWSTG1QA.AFS_ADD_DATA_SEGMENT_DELTA_HISTORY                      LOAD_DATE                                DATE
EDWSTG1QA.AFS_ADD_DATA_SEGMENT_PREVIOUS                           LOAD_DATE                                DATE
EDWSTG1QA.AFS_ADD_DATA_SEGMENT_SNAP                               LOAD_DATE                                DATE
EDWSTG1QA.WEB_SEGMENT_LOAD_SNAP                                   AM01_IMAGE_LOAD_DATE                     DECIMAL

  7 record(s) selected.
```

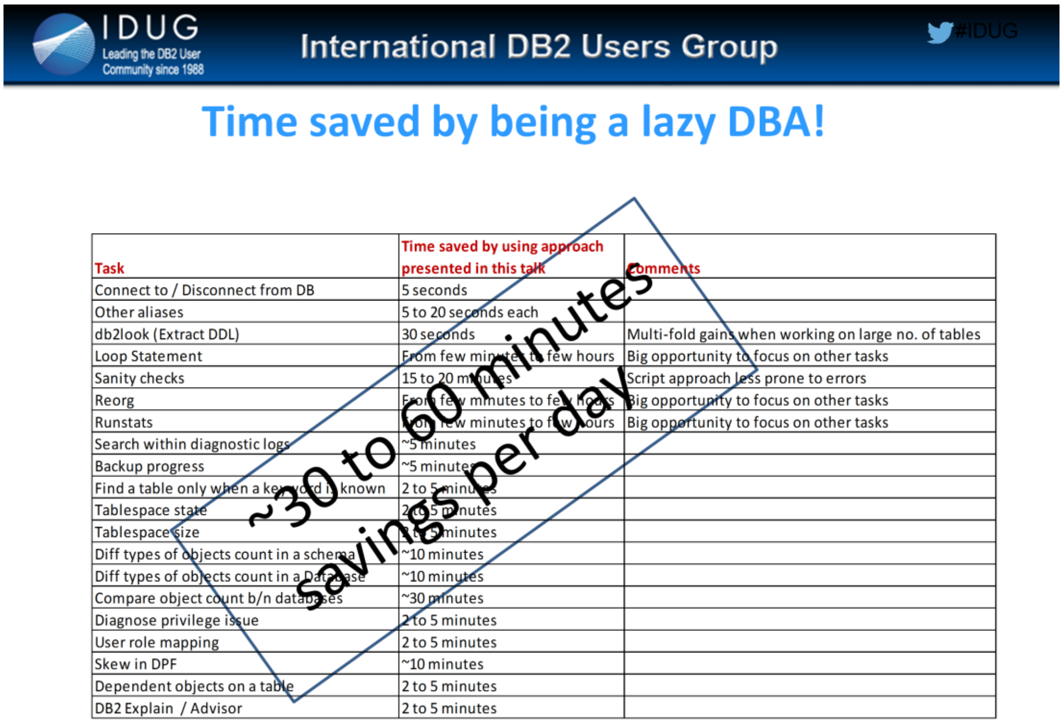
```
#!/bin/ksh

. $HOME/sqllib/db2profile

COL=`echo $1 | tr '[:lower:]' '[:upper:]'`

##echo "I understand you want to get list of tables that have the column $1 in them"
##echo "Searching ..."

database=`db2 list database directory show detail | grep -B6 -i indirect | grep "Database name" | sed "s/.*= //" `

db2 connect to $database > /dev/null

db2  "select trim(char(tabschema, 10)) || '.' || char(tabname,60) as table_name,char(colname, 40) as COLUMN,
      char(typename,30) as datatype from syscat.columns where colname like '%$COL%' order by tabname with ur"
```

This script lists all the tables that has matching column name in it.

Other Miscellaneous scripts

- emailfile – Emails a file as an attachment
- ibmftp – upload diagnostic data (db2support for example) to IBM PMR repository
- reorg_pending in crontab – Checks for reorg pending tables every few minutes
- Range_count – for range partitioning tables
- High_cpu_apps -- prints list of applications by cpu usage
- devtoqa.ksh -- migrates from dev to qa environment
- pks, uks, fks – display 'keys' information on a table
- Readtable – quickly displays data from the table
- Count – does select count(*) for a table / view

More than 50+ homegrown scripts



International DB2 Users Group

IDUG Leading the DB2 User Community since 1988

#IDUG

Time saved by being a lazy DBA!

Task	Time saved by using approach presented in this talk	Comments
Connect to / Disconnect from DB	5 seconds	
Other aliases	5 to 20 seconds each	
db2look (Extract DDL)	30 seconds	Multi-fold gains when working on large no. of tables
Loop Statement	From few minutes to few hours	Big opportunity to focus on other tasks
Sanity checks	15 to 20 minutes	Script approach less prone to errors
Reorg	From few minutes to few hours	Big opportunity to focus on other tasks
Runstats	From few minutes to few hours	Big opportunity to focus on other tasks
Search within diagnostic logs	~5 minutes	
Backup progress	~5 minutes	
Find a table only when a keyword is known	2 to 5 minutes	
Tablespace state	2 to 5 minutes	
Tablespace size	2 to 5 minutes	
Diff types of objects count in a schema	~10 minutes	
Diff types of objects count in a Database	~10 minutes	
Compare object count b/n databases	~30 minutes	
Diagnose privilege issue	2 to 5 minutes	
User role mapping	2 to 5 minutes	
Skew in DPF	~10 minutes	
Dependent objects on a table	2 to 5 minutes	
DB2 Explain / Advisor	2 to 5 minutes	

~30 to 60 minutes savings per day

Summary of time / effort savings when scripts mentioned in this presentation are used.

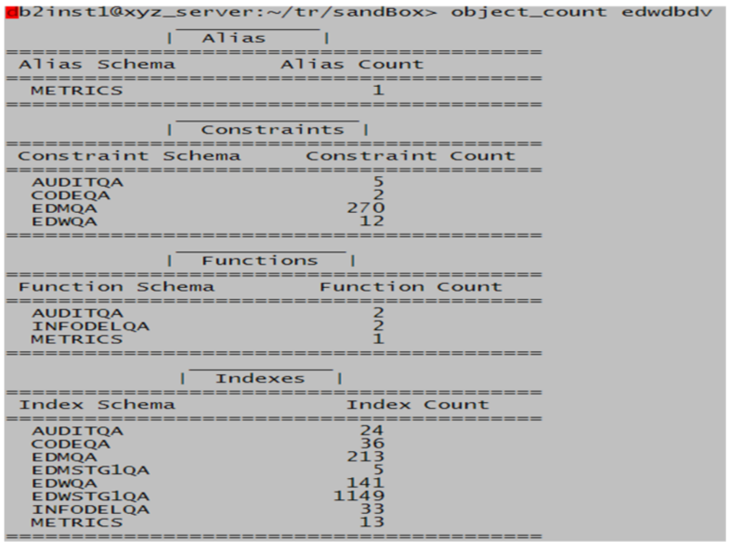
Thank you for attending.

Any Questions?

BONUS

Object Count at Database Level - 1

Challenge: Find types of objects in a database (Migration effort)



```
db2inst1@xyz_server:~/tr/sandBox> object_count edwdbdv
              | Alias      |
=========================================
 Alias Schema           Alias Count
=========================================
  METRICS                         1
=========================================
              | Constraints |
=========================================
 Constraint Schema     Constraint Count
=========================================
  AUDITQA                         5
  CODEQA                          2
  EDMQA                         270
  EDWQA                          12
=========================================
              | Functions |
=========================================
 Function Schema         Function Count
=========================================
  AUDITQA                         2
  INFODELQA                       2
  METRICS                         1
=========================================
               | Indexes |
=========================================
 Index Schema              Index Count
=========================================
  AUDITQA                        24
  CODEQA                         36
  EDMQA                         213
  EDMSTG1QA                       5
  EDWQA                         141
  EDWSTG1QA                    1149
  INFODELQA                      33
  METRICS                        13
=========================================
```

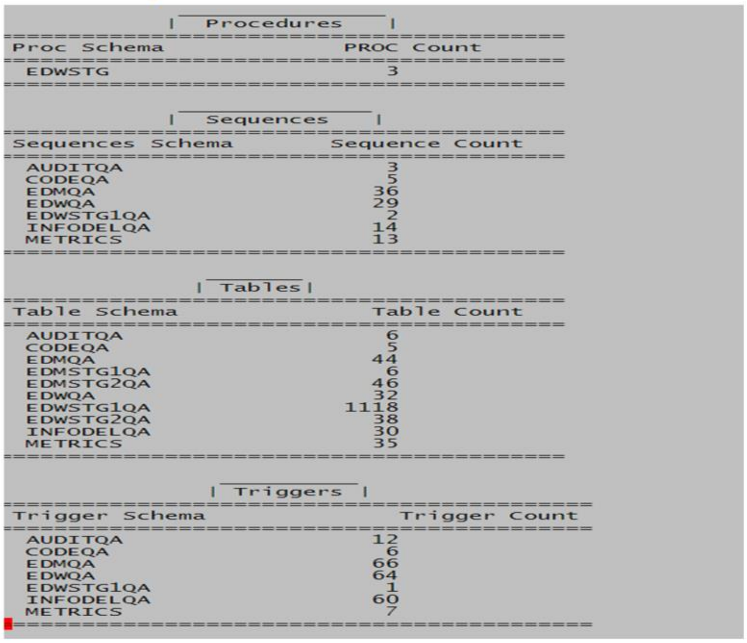
This script gives a snapshot of counts of different database object types in the database.

This script is a lengthy one to include as a snapshot or in the notes. Please email if you want the source code for this one.

International DB2 Users Group

#IDUG

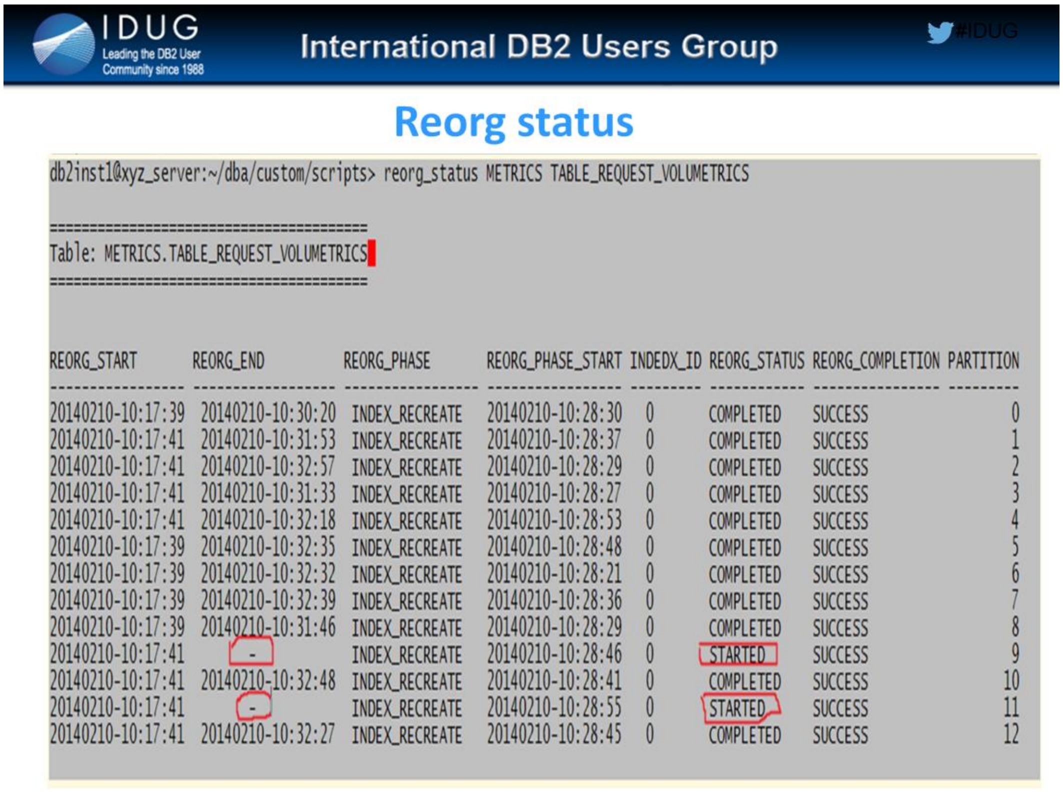
Object Count at Database Level - 2



```
             |   Procedures   |
========================================
Proc Schema                  PROC Count
========================================
  EDWSTG                            3
========================================

             |  Sequences   |
========================================
Sequences Schema       Sequence Count
========================================
  AUDITQA                           3
  CODEQA                            5
  EDMQA                            36
  EDWQA                            29
  EDWSTG1QA                         2
  INFODELQA                        14
  METRICS                          13
========================================

               | Tables |
========================================
Table Schema                Table Count
========================================
  AUDITQA                           6
  CODEQA                            5
  EDMQA                            44
  EDMSTG1QA                         6
  EDMSTG2QA                        46
  EDWQA                            32
  EDWSTG1QA                      1118
  EDWSTG2QA                        38
  INFODELQA                        30
  METRICS                          35
========================================

               | Triggers  |
==========================================
Trigger Schema               Trigger Count
==========================================
  AUDITQA                          12
  CODEQA                            6
  EDMQA                            66
  EDWQA                            64
  EDWSTG1QA                         1
  INFODELQA                        60
  METRICS                           7
==========================================
```

This script gives a snapshot of counts of different database object types in the database.

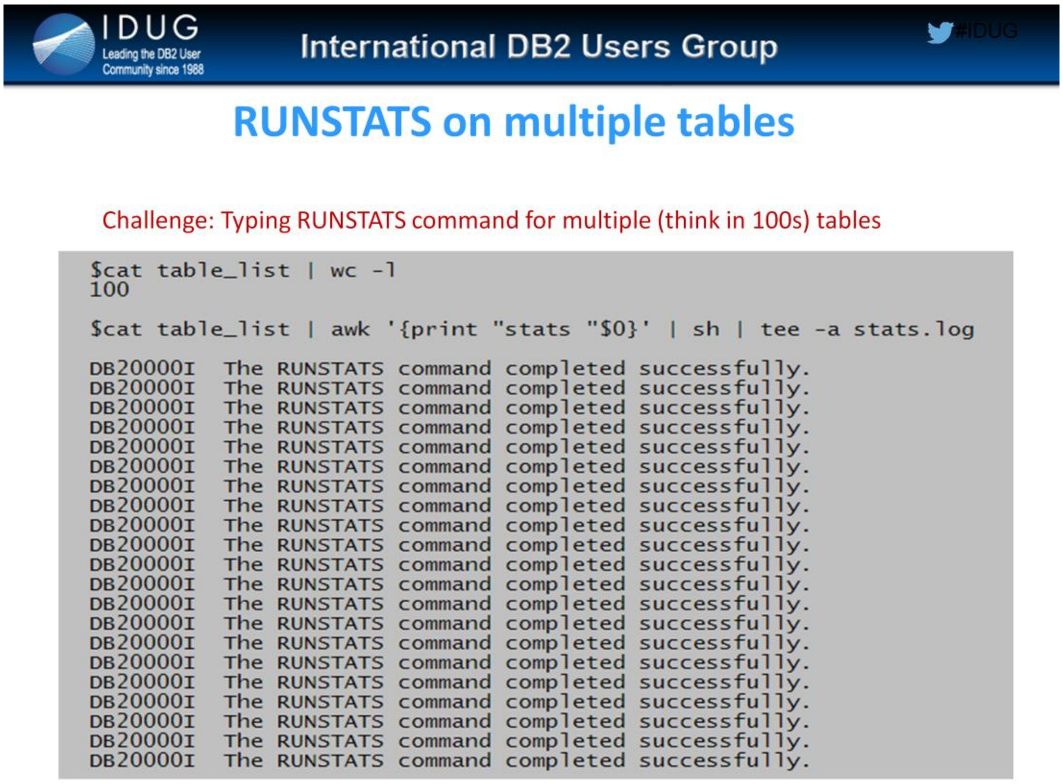
This script is a lengthy one to include as a snapshot or in the notes. Please email if you want the source code for this one.


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

Reorg status

```
db2inst1@xyz_server:~/dba/custom/scripts> reorg_status METRICS TABLE_REQUEST_VOLUMETRICS

=======================================
Table: METRICS.TABLE_REQUEST_VOLUMETRICS
=======================================


REORG_START       REORG_END          REORG_PHASE       REORG_PHASE_START INDEDX_ID REORG_STATUS REORG_COMPLETION PARTITION
----------------- ------------------ ----------------- ----------------- --------- ------------ ---------------- ---------
20140210-10:17:39 20140210-10:30:20  INDEX_RECREATE    20140210-10:28:30 0         COMPLETED    SUCCESS                  0
20140210-10:17:41 20140210-10:31:53  INDEX_RECREATE    20140210-10:28:37 0         COMPLETED    SUCCESS                  1
20140210-10:17:41 20140210-10:32:57  INDEX_RECREATE    20140210-10:28:29 0         COMPLETED    SUCCESS                  2
20140210-10:17:41 20140210-10:31:33  INDEX_RECREATE    20140210-10:28:27 0         COMPLETED    SUCCESS                  3
20140210-10:17:41 20140210-10:32:18  INDEX_RECREATE    20140210-10:28:53 0         COMPLETED    SUCCESS                  4
20140210-10:17:39 20140210-10:32:35  INDEX_RECREATE    20140210-10:28:48 0         COMPLETED    SUCCESS                  5
20140210-10:17:39 20140210-10:32:32  INDEX_RECREATE    20140210-10:28:21 0         COMPLETED    SUCCESS                  6
20140210-10:17:39 20140210-10:32:39  INDEX_RECREATE    20140210-10:28:36 0         COMPLETED    SUCCESS                  7
20140210-10:17:39 20140210-10:31:46  INDEX_RECREATE    20140210-10:28:29 0         COMPLETED    SUCCESS                  8
20140210-10:17:41         -          INDEX_RECREATE    20140210-10:28:46 0         STARTED      SUCCESS                  9
20140210-10:17:41 20140210-10:32:48  INDEX_RECREATE    20140210-10:28:41 0         COMPLETED    SUCCESS                 10
20140210-10:17:41         -          INDEX_RECREATE    20140210-10:28:55 0         STARTED      SUCCESS                 11
20140210-10:17:41 20140210-10:32:27  INDEX_RECREATE    20140210-10:28:45 0         COMPLETED    SUCCESS                 12
```

This script gives the reorg details such as reorg_start time, reorg_end time, reorg_phase, phase_status, phase_start time for each partition.


IDUG Leading the DB2 User Community since 1988 International DB2 Users Group #IDUG

RUNSTATS on multiple tables

Challenge: Typing RUNSTATS command for multiple (think in 100s) tables

```
$cat table_list | wc -l
100

$cat table_list | awk '{print "stats "$0}' | sh | tee -a stats.log

DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
DB20000I  The RUNSTATS command completed successfully.
```

Simple script that would do runstats on a table.

The main time saving that DBAs would realize is when there are multiple tables that need to be reorg’d.