**Circular Logging Vs Archival Logging:**

**Below command will let to convert from Circular logging to Archival logging.**

CONNECT TO TC2;  
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;  
UNQUIESCE DATABASE;  
CONNECT RESET;  
UPDATE DB CFG FOR TC2 USING logarchmeth1 "DISK:c:\ibm" logprimary 15 logsecond 4 logfilsiz 1024;  
BACKUP DATABASE TC2 TO "C:\IBM" WITH 2 BUFFERS BUFFER 9 PARALLELISM 1 WITHOUT PROMPTING;

**drop trigger TEMP\_NC\_INSERT ;**

**CREATE TRIGGER TEMP\_NC\_INSERT NO CASCADE BEFORE INSERT ON SCHEMANAME.TEMP\_NC**

**REFERENCING NEW AS NEW\_EM\_NC FOR EACH ROW MODE DB2SQL**

**BEGIN ATOMIC**

**SET NEW\_EM\_NC.createdon = CURRENT TIMESTAMP ;--**

**END**

**CREATE TRIGGER INT016\_E03**

**AFTER INSERT ON schemaname.CLIENTNOTICES\_EXCP**

**FOR EACH ROW MODE DB2SQL**

**BEGIN ATOMIC**

**delete from schemaname.TABLENAMEwhere caseid = 1061714844 ;**

**END#**

**CREATE TRIGGER INT016\_E03\_NEW**

**AFTER INSERT ON schemaname.CLIENTNOTICES\_EXCP**

**FOR EACH ROW MODE DB2SQL**

**BEGIN ATOMIC**

**WHEN (CASEID = 1061714844)**

**UPDATE schemaname.TABLENAMESET errorcode = 'XXX' ;**

**END#**

**drop trigger TEMP\_NC\_UPDATE ;**

**CREATE TRIGGER TEMP\_NC\_UPDATE NO CASCADE BEFORE UPDATE ON SCHEMANAME.TEMP\_NC**

**REFERENCING NEW AS NEW\_EM\_NC FOR EACH ROW MODE DB2SQL**

**BEGIN ATOMIC**

**SET NEW\_EM\_NC.lastupdatedon = CURRENT TIMESTAMP ;--**

**END**

**LOCATE QUERY:**

**export to /etldata/export/data/1.del of del**

**select a.call\_id,date(a.connected\_dt) as CALL\_DATE,time(a.call\_start\_ts) AS CALL\_START\_TIME,time(a.call\_end\_ts) AS CALL\_END\_TIME,**

**a.CALL\_DURATION,a.WRAP\_UP\_CD AS WRAP\_UP\_CODE, REPLACE( substr( substr(a.CALL\_EVENT\_LOG,LOCATE('Entered Workgroup Local\_',a.CALL\_EVENT\_LOG), length(a.CALL\_EVENT\_LOG) -LOCATE('Entered Workgroup Local\_',a.CALL\_EVENT\_LOG)),1, posstr(substr(a.CALL\_EVENT\_LOG,LOCATE('Entered Workgroup Local\_',a.CALL\_EVENT\_LOG), length(a.CALL\_EVENT\_LOG) -LOCATE('Entered Workgroup Local\_',a.CALL\_EVENT\_LOG)),chr(10))**

**)**

**,'Entered Workgroup ','') as QUEUE\_NAME,a.local\_user\_id AS USER\_ID**

**FROM SCHEMANAME.T\_DW\_CALL\_DETAIL a**

**where (a.call\_event\_log like '%Local\\_%' ESCAPE '\')**

**AND date(a.as\_of\_dt)='2012-06-21'**

**AND a.CALL\_EVENT\_LOG not LIKE '%HIP\_Tier2%'**

**UNION**

**select a.call\_id,date(a.connected\_dt) as CALL\_DATE,time(a.call\_start\_ts) AS CALL\_START\_TIME,time(a.call\_end\_ts) AS CALL\_END\_TIME, a.CALL\_DURATION,**

**a.WRAP\_UP\_CD AS WRAP\_UP\_CODE, LEFT((SUBSTR(a.CALL\_EVENT\_LOG,LOCATE('RCC',a.CALL\_EVENT\_LOG),LOCATE('RCC',a.CALL\_EVENT\_LOG))),5) as QUEUE\_NAME,a.local\_user\_id AS USER\_ID**

**FROM SCHEMANAME.T\_DW\_CALL\_DETAIL a**

**where (a.call\_event\_log like '%RCC%')**

**AND date(a.as\_of\_dt)='2012-06-21'**

**AND a.CALL\_EVENT\_LOG not LIKE '%HIP\_Tier2%'**

**with ur;**

**SESSION TABLE:**

**DECLARE GLOBAL TEMPORARY TABLE session.TEMP\_MR28CLEANUP(**

**caseid BIGINT )**

**NOT LOGGED WITH replace ON COMMIT PRESERVE ROWS ;**

Check if REORG needed:

**CALL SYSPROC.REORGCHK\_TB\_STATS('T','JESCOTT.EMPLOYEE')**

**Alter Primary Key Command:**

**ALTER TABLE SCHEMANAME.T\_EM\_MAD\_BCKLG\_SCORECARD DROP PRIMARY KEY ADD CONSTRAINT MAD\_BCKLG\_PK PRIMARY KEY ( ICES\_RID, ICES\_CASE\_REF, ICES\_CAT\_CD, ICES\_CAT\_SEQ, AS\_OF\_DT) ;**

How to get All columns from a Table for Archival and Export

**SELECT 'SELECT ' || substr(replace(replace( xmlserialize(content xmlagg(**

**xmlelement(name "c", colname) order by colno)**

**as varchar(4500)), '<c>', ','), '</c>', ''), 2) || ' FROM ' ||**

**tabname**

**FROM syscat.columns**

**where**

**tabschema= 'SCHEMANAME'**

**GROUP BY tabname**

**How to find unused indexes?**

**select \* from sysibm.sysplandep**

**select \* from sysibm.syspackdep**

**step 1)**

db2pd -db empscdb1 -tcbstats all tbspaceid=13 tableid=3865 -file db2pd\_tab\_attachment.txt

**step 2)**

db2 "SELECT INDSCHEMA, INDNAME

FROM SYSCAT.INDEXES

WHERE TABNAME = 'TAB1' AND IID = 11"

**--Abobe with Zero scanes are unuxed indexes.**

**Another method to find unused index:**

*SELECT SUBSTR(SI.INDSCHEMA, 1, 30) AS INDSCHEMA, SUBSTR(SI.INDNAME, 1, 30) AS INDNAME, MGI.INDEX\_SCANS, MGI.INDEX\_ONLY\_SCANS*

*FROM TABLE(MON\_GET\_INDEX('SCHEMANAME', 'ATTACHMENT', -1)) as MGI, SYSCAT.INDEXES AS SIWHERE MGI.TABSCHEMA = SI.TABSCHEMA AND MGI.TABNAME = SI.TABNAME AND MGI.IID = SI.IIDORDER BY MGI.INDEX\_SCANS DESC;*

**To Install license for DB2 :**

**Step 1**

**To see the status of license product**

**Db2licl –l**

**To update the license for DB2 with a license key issue below:**

For example, if the license file db2ese.lic is copied to /tmp on the machine where DB2 ESE is installed, run the following command to update the license file:

**db2licm -a /tmp/db2ese.lic**

**STMM setup**

**if [[ $# != 2 || $1 != "-d" ]];then**

**print "Invalid Arguments! - Usage: stmminstall.ksh -d [DATABASENAME]"**

**exit**

**fi**

**dbname=$2**

**echo "Start time: $(date)\n\n" > STMM.$dbname.out**

**db2 connect to $dbname >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit;**

**fi**

**echo "\nBacking up DB and DBM CFGs" >> STMM.$dbname.out**

**db2 get db cfg > dbcfg.$dbname.$(date +%Y%m%d-%H%M%S).txt**

**db2 get dbm cfg > dbmcfg.$LOGNAME.$(date +%Y%m%d-%H%M%S).txt**

**echo "\nBackups complete" >> STMM.$dbname.out**

**echo "\nStarting STMM Implementation" >> STMM.$dbname.out**

**echo "\n\ndb2 update database configuration for $dbname using SELF\_TUNING\_MEM ON" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using SELF\_TUNING\_MEM ON >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database configuration for $dbname using PCKCACHESZ AUTOMATIC" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using PCKCACHESZ AUTOMATIC >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database configuration for $dbname using LOCKLIST automatic MAXLOCKS automatic" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using LOCKLIST automatic MAXLOCKS automatic >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database configuration for $dbname using SORTHEAP AUTOMATIC" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using SORTHEAP AUTOMATIC >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database manager configuration using SHEAPTHRES 0" >> STMM.$dbname.out**

**db2 update database manager configuration using SHEAPTHRES 0 >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database configuration for $dbname using SHEAPTHRES\_SHR AUTOMATIC" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using SHEAPTHRES\_SHR AUTOMATIC >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\ndb2 update database configuration for $dbname using DATABASE\_MEMORY AUTOMATIC" >> STMM.$dbname.out**

**db2 update database configuration for $dbname using DATABASE\_MEMORY AUTOMATIC >> STMM.$dbname.out**

**if [ $? != 0 ]; then**

**cat STMM.$dbname.out**

**exit**

**fi**

**echo "\nDB and DBM Config Settings Completed" >> STMM.$dbname.out**

**echo "\nSTMM Implementation Complete" >> STMM.$dbname.out**

**echo "\n\nStop time: $(date)" >> STMM.$dbname.out**

**cat STMM.$dbname.out**

**To find big files in AIX:**

# du -a /var | sort -n -r | head -n 10

**FKC Foreign Key checker**

**select 'alter table' || ' id' || '.' || tabname || ' alter foreign key ' || constname || ' NOT ENFORCED;'**

**from syscat.references WHERE TABSCHEMA = 'ID'**

**Grant priviliges**

**select 'GRANT SELECT ON' || ' '|| rtrim(tabschema) || '.' || char(tabname,40) || 'TO USERID||';' FROM syscat.tables**

**WHERE tabschema='SCHEMANAME' ORDER BY tabschema, tabname;**

select 'GRANT SELECT,UPDATE,DELETE,INSERT ON' || ' '|| rtrim(tabschema) || '.' || char(tabname,40) || 'TO username1,username2'||';' FROM syscat.tables

WHERE tabschema='SCHEMANAME' ORDER BY tabschema, tabname;

select 'GRANT EXECUTE ON PROCEDURE' || ' '|| rtrim(routineschema) || '.' || char(routinename,40) || 'TO emsscin4'||';' FROM syscat.routines;

select 'GRANT usage ON sequence' || ' '|| rtrim(seqschema) || '.' || char(seqname,40) || 'TO emsscin4'||';' FROM syscat.sequences ;

**select 'GRANT' || ' '|| rtrim(privilege) || ' ON' || ' ' || rtrim(char(objectschema,40)) || '.' || char(objectname,40) || 'TO EMSSCIW5'||';' FROM sysibmadm.PRIVILEGES**

**WHERE authid = 'USERID' and privilege <> 'USAGE' ;**

**commit;**

**select 'GRANT' || ' '|| rtrim(privilege) || ' ON' || ' SEQUENCE ' || rtrim(char(objectschema,40)) || '.' || char(objectname,40) || 'TO EMSSCIW5'||';' FROM sysibmadm.PRIVILEGES**

**WHERE authid = 'USERID' and privilege = 'USAGE' ;**

**select 'create alias emsscwu1.'|| char(tabname,40) ||'for '||rtrim(tabschema) || '.' || char(tabname,40)||' ;' FROM syscat.tables**

**WHERE tabschema='SCHEMANAME' ORDER BY tabschema, tabname;**

**To Give Privileges:**

**select 'GRANT' || ' '|| rtrim(PRIVILEGE) || ' '|| 'ON' || ' '|| rtrim(objectschema) || '.' || rtrim(OBJECTNAME) || ' ' || 'TO' || ' ' || rtrim(AUTHID) || ';'**

**FROM sysibmadm.PRIVILEGES**

**WHERE**

**objectschema in (userid)**

**and authid in(userid);**

**select 'GRANT' || ' '|| rtrim(PRIVILEGE) || ' '|| 'ON' || ' '|| 'SEQUENCE' || ' '|| rtrim(objectschema) || '.' || rtrim(OBJECTNAME) || ' ' || 'TO' || ' ' || rtrim(AUTHID) || ';'**

**FROM sysibmadm.PRIVILEGES**

**WHERE**

**objectschema in (userid)**

**and authid in(userid) and privilege = 'USAGE' ;**

**commit;**

**select 'GRANT' || ' '|| rtrim(PRIVILEGE) || ' '|| 'ON' || ' '|| 'PROCEDURE' || ' '|| rtrim(objectschema) || '.' || rtrim(OBJECTNAME) || ' ' ||**

**'TO' || ' ' || rtrim(AUTHID) || ';'**

**FROM sysibmadm.PRIVILEGES**

**WHERE**

**objectschema in (userid)**

**and authid in(userid) and privilege = 'EXECUTE' ;**

**commit;**

**DB2Functions for Priority -- (works)**

drop FUNCTION SCHEMANAME.newfun~

CREATE FUNCTION SCHEMANAME.newfun

( P\_CON\_ID INT

)

RETURNS CHAR(5)

LANGUAGE SQL

NOT DETERMINISTIC

EXTERNAL ACTION

READS SQL DATA

P:BEGIN ATOMIC

RETURN

select t1.a\_type from SCHEMANAME.temp t1,

(values (1,'typ3'), (2,'typ4'), (3,'typ1'), (4,'typ2')) as t2(rnk, des)

where t1.a\_type = t2.des

and id = P\_CON\_ID

order by rnk asc

fetch first 1 row only;

END P~

commit~

select \* from SCHEMANAME.temp~

select distinct id, SCHEMANAME.newfun(id) as type from SCHEMANAME.temp~

commit~

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

select \* from SCHEMANAME.temp

ID SUB\_ID C\_NAME A\_TYPE

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

1 1 name1 typ1

1 2 name1 typ2

1 3 name1 typ3

2 1 name2 typ2

2 2 name2 typ3

3 1 name3 typ1

4 1 name4 typ2

5 1 name5 typ2

5 2 name5 typ3

9 record(s) selected.

select distinct id, SCHEMANAME.newfun(id) as type from SCHEMANAME.temp

ID TYPE

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

3 typ1

4 typ2

1 typ3

2 typ3

5 typ3

5 record(s) selected.

**can you please run this query, it will give tablename, sequencename and other details. We probably set-up to run everyday via automation.**

select seqname Seq\_name,c.tbcreator Table\_schema, c.tbname table\_name, c.name seq\_column\_name, c.identity ,lastassignedval SeqValuedUsed,s.start, s.increment, s.minvalue,s.maxvalue, s.cycle, s.cache, s.seqid

from sysibm.syscolumns as c,sysibm.sysdependencies as d, sysibm.syssequences as s

where c.tbcreator = d.dschema and c.tbname = d.dname and d.bname = s.seqname and d.bschema = s.seqschema --and c.identity = 'Y' and d.dtype = 'T' and d.btype ='Q' and s.seqtype = 'I' with ur;\

Here is the query to find out sequence details and related table.

select seqname Seq\_name,c.tbcreator Table\_schema, c.tbname table\_name, c.name seq\_column\_name, c.identity ,lastassignedval SeqValuedUsed,s.start, s.increment, s.minvalue,s.maxvalue, s.cycle, s.cache, s.seqid

from sysibm.syscolumns as c,sysibm.sysdependencies as d, sysibm.syssequences as s

where c.tbcreator = d.dschema and c.tbname = d.dname and d.bname = s.seqname and d.bschema = s.seqschema --and c.identity = 'Y' and d.dtype = 'T' and d.btype ='Q' and s.seqtype = 'I' with ur;

**Db2 Export Comamnd for LOBS:**

**DROP PRIMARY KEY COMMAND:**

**ALTER TABLE SCHEMANAME.ALERT\_TASK\_DETAIL DROP PRIMARY KEY ;**

**export to 'c:\task.del' of del lobs to 'c:\' lobfile lobs1 modified by lobsinfile select \* from SCHEMANAME.archive\_task fetch first 10 rows only with ur**

**Chinking and streaming:**

**ALTER TABLE BATCHPROCESS VOLATILE CARDINALITY;**

**ALTER TABLE BATCHPROCESSCHUNK VOLATILE CARDINALITY;**

**SQL to FIND IP ADDRESS**

select agent\_id, agent\_usr\_cpu\_time\_s, agent\_usr\_cpu\_time\_ms, appl\_con\_time,conn\_complete\_time,inbound\_comm\_address from table(SNAPSHOT\_APPL('EMPSCDB1',-1)) as applsnap;

**backup database for a tablespace**

**db2 "backup db dbnameonline tablespace (ODSDEVIXTS) to /dev/null"**

**HISTORY COMMAND FOR BACKUP**

**db2 list history backup all for database dbname**

**GREP COMMANDS:**

**grep 2000044152 sample.del >> sample1.del**

**to delete from input file:**

**vi filename:/grep 3000435689dd**

**TO DISABLE CONSTRAINTS:**

**select 'alter table ' || tabschema || '.' || tabname || ' alter foreign key ' || constname || ' NOT ENFORCED;'**

**from syscat.references WHERE TABSCHEMA = 'ID';**

**TO ENABLE CONSTRAINTS:**

**select 'alter table ' || tabschema || '.' || tabname || ' alter foreign key ' || constname || ' ENFORCED;'**

**To fix BIND issues:**

**db2 terminate  
db2 CONNECT TO DBNAME user <username> using <password>  
db2 bind <BNDPATH>**[/@db2ubind.lst](mailto:/@db2ubind.lst) **blocking all sqlerror continue messages bind.msg grant public  
db2 BIND <BNDPATH>**[/@db2cli.lst](mailto:/@db2cli.lst) **BLOCKING ALL GRANT PUBLIC ACTION ADD  
db2 terminate**

db2 bind @db2ubind.lst blocking all grant public

db2 "rebind package NULLID.SQLC2H21"

**db2 terminate  
db2 CONNECT TO DBNAME user <username> using <password>  
db2 bind** [@db2ubind.lst](mailto:/@db2ubind.lst) **blocking all sqlerror continue messages bind.msg grant public  
db2 BIND BLOCKING ALL GRANT PUBLIC ACTION ADD  
db2 terminate**

**db2 terminate  
db2 CONNECT TO DBNAME user <username> using <password>  
db2 bind** [@db2ubind.lst](mailto:/@db2ubind.lst) **blocking all sqlerror continue messages bind.msg grant public  
db2 BIND** [@db2cli.lst](mailto:/@db2cli.lst) **BLOCKING ALL GRANT PUBLIC ACTION ADD  
db2 terminate**

If you look at this link:  
<https://www-304.ibm.com/support/docview.wss?uid=swg21516702>

**SELECT \* FROM SYSCAT.INDEXES WHERE TABSCHEMA IN ('SCHEMANAME','SCHEMANAME')**

**AND DATE(CREATE\_TIME) BETWEEN '2008-09-25' AND '2008-10-07'**

**ORDER BY CREATE\_TIME;**

**RUNSTATS and REORG**

**select start\_time,end\_time,tabname,sqlcode from table(admin\_list\_hist()) as lh**

**where operation='G' and operationtype='F' and comment='REORG'**

**AND SQLCODE IS NOT NULL**

**AND substr(start\_time,1,8) > '20080901' AND substr(start\_time,1,8) < '20081007'**

**GRANT CHECK COMMAND:**

**select substr(grantee,1,20) grantee,substr(tabname,1,30) tablename, DELETEAUTH,INSERTAUTH ,SELECTAUTH ,UPDATEAUTH**

**from syscat.tabauth where tabname in ('CASEHEADER','EMLOOKAHEADSTAGE','EMTASKAGASSIGNMENT')**

**order by grantee;**

**db2adutl command:**

db2adutl query -== used to see the backup information

**To find IP address of connecting application or user:**

select agent\_id, agent\_usr\_cpu\_time\_s, agent\_usr\_cpu\_time\_ms, appl\_con\_time,conn\_complete\_time,inbound\_comm\_address from table(SNAPSHOT\_APPL('EMPSCDB1',-1)) as applsnap;

**MONITORING SCRIPTS:**

**--To find high table count tables in a tablespace**

select substr(name,1,45) table\_name,card record\_count

from sysibm.systables where tbspace = 'ODSDEVTS'

order by card desc with ur

Here are the monitoring scripts... Try it and have it handy.

--Monitoring scripts

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

# db2diag and db2dart – 2 life saver utilities

by on September 13, 2011

Monitoring diagnostic log is very important task in a DBA`s daily life. When you compare this with onset of a chronic disease like heart attack or cancer you find that they are analogies. For these chronic diseases you will see symptoms like pain, weight loss and other related symptoms before the disease actually reaches serious stage. Similarly for a serious database issue you will have related records as a precursor in diaglog most of the time. So carefully reviewing your diagnostic logs will save you from unexpected dangers.

Most of us look at the diagnostic log when we notice a serious issue. Instead of doing that , if we proactively monitor the diaglog we will be able to prevent many serious issues.

To review diaglogs proactively there are four options:

**1)PDLOGMSGS\_LAST24HOURS administrative view**

Example:

SELECT \* FROM SYSIBMADM.PDLOGMSGS\_LAST24HOURS  
ORDER BY TIMESTAMP DESC

**2)​PD\_GET\_LOG\_MSGS table function**

Example:

SELECT \*  
FROM TABLE ( PD\_GET\_LOG\_MSGS( CURRENT\_TIMESTAMP – 7 DAYS)) AS T  
WHERE INSTANCENAME = ‘DB2′ AND DBNAME = ‘SAMPLE’  
ORDER BY TIMESTAMP ASC

**3)Perl program or shell script to parse diaglog**

**4)db2diag utility**

Diaglog utility is the most flexible of all with wide options . Below are some of the common ones.

db2diag -A    –> To archive the diaglog

db2diag -time  –> To retrieve records for a particular time

db2diag -time 2011-09-09 -l severe,error  –> To retrieve records for a particular time. Only severe and error messages are returned here

db2diag -l severe –> To return only severe messages

db2diag -g db=SAMPLE,instance=aabrashk –> To return messages of SAMPLE database and instance aabrashk

db2diag -g -d SAMPLE –> To return messages related to SAMPLE database only

db2diag -g level=Severe,pid=952356 –> To return severe messages from process id 952356

db2diag -g ‘funcname=pdLog’ –> To return the messages related to the particular function name

db2diag -lastrecords 5  –> To return the last 5 records

system(“db2diag -readfile”); –> To read from perl script

db2diag -gi “level=severe” -H 3d –> To return the severe messages from last 3 days

db2diag -g msg:=0×87040055 –> to return records with a particular ZRC

db2diag -rc –>displays descriptions for db2 internal ZRC codes

db2diag -gi appid:=local –> To return the messages related to the appid local

CIRCULAR TO ARCHIVAL LOGGING:

db2 update db cfg for dbquiz using LOGRETAIN on  
  
*or*  
  
db2 update db cfg for dbquiz using USEREXIT on  
  
*or*  
  
db2 update db cfg for dbquiz using LOGARCHMETH1 <value>  
  
*or*  
  
db2 update db cfg for dbquiz using LOGARCHMETH2 <value>

**REPLACE AMOMMAND:**

REPLACE(A2.comments,'|',' ')

**Db2dart utility**

**——————-**

Did you ever run into one of the worst scenarios wherein your database is corrupt. Some tables might have become corrupt because someone messed up with active logs. To recover from this scenario , you could try an export followed by import . But the export utility would fail as the table pages were corrupt. We recovered a huge database recently with the following procedure:

1)Check the file system using fsck utility

2)Check and see what backups and log files are present .. Do we have all the archive logs in one place ? Making sure that you have backups and archive logs before this kind of activity always helps.If you run into any major issue , you can always recover using​ the backup image and archive logs

3)Check and see what kind of dependencies the table has . Does it have any referential integrity constraints ? When you load the data into the table , you got to make sure the parent-child relationships are still valid after the loads.

4)Collect all table and indexes DDL and store in a file . Do not forget the permissions on objects

5)Evaluate the row count to see what is missing towards the end. Once the load from db2dart is done you have to double check the count to see if you have got all the data or not

6) db2dart dbname /DDEL . This will generate a DEL file containing all records that it could recover from corrupted table.

It will prompt for tablename tableid start page and end page

You can give the end page a very big number to cover all

7)Create a temp table similar to actual and load the dump to this temp table

8)Rename the actual to actual\_copy and the temp table created in step 7 as actual. Before this step is performed it has to be verified that the table has no views , triggers , constraints on it . If there are any views on the table the rename will fail. You have to drop the views and recreate them after the rename is done

e.g RENAME SAMPLE.EMPLOYEE to SAMPLE.EMPLOYEE\_BACKUP

RENAME SAMPLE.EMPLOYEE\_COPY to SAMPLE.EMPLOYEE  (EMPLOYEE\_COPY is the temp table generated using the data from db2dart)

9)If there are views, triggers as mentioned in step 8 , they have to be dropped and recreated once the rename is done

10) create indexes and grant appropriate permissions on table using the file generated in step 4. Indexes has to be created with new names as the old indexes are attached to  backup table

11) Runstats utility has to be run to gather statistics for optimizer usage

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

select substr(stmt\_text,1,100) as Statement, query\_cost\_estimate, query\_card\_estimate, total\_sort\_time, sort\_overflows, stmt\_sys\_cpu\_time\_ms,

stmt\_elapsed\_time\_ms from table(snapshot\_statement('',-1)) as st

where sort\_overflows > 0 order by sort\_overflows desc fetch first 10 rows only

--

SELECT ELAPSED\_TIME\_MIN,SUBSTR(AUTHID,1,10) AS AUTH\_ID, AGENT\_ID,APPL\_STATUS,SUBSTR(STMT\_TEXT,1,20) AS SQL\_TEXT

FROM TABLE(SNAPSHOT\_STATEMENT('',-1)) as st WHERE ELAPSED\_TIME\_MIN > 0 ORDER BY ELAPSED\_TIME\_MIN DESC

--Find REORG STATUS

select \* from table(admin\_list\_hist()) as lh

where operation='G' and operationtype='F' and comment='REORG' and substr(start\_time,1,8) ='20080823'

--To find REORG CURRENT STATUS

select rtrim(a.tabschema)||'.'||a.tabname

from

(select \* from table(admin\_list\_hist()) as a1

where operation='G' and operationtype='N' and (comment='REORG START' or comment='REORG RESUME') ) as a

left outer join

(select \* from table(admin\_list\_hist()) as a2

where operation='G' and operationtype='N' and COMMENT in ('REORG Done','REORG STOP','REORG PAUSE')) as b

on (a.TABNAME = b.TABNAME and a.end\_time <= b.start\_time)

where

b.TABNAME is null;

--To Find The Paused Status

select rtrim(a.tabschema)||'.'||a.tabname

from

(select \* from table(admin\_list\_hist()) as a1

where operation='G' and operationtype='N' and (comment='REORG PAUSE' ) ) as a

left outer join

(select \* from table(admin\_list\_hist()) as a2

where operation='G' and operationtype='N' and COMMENT in ('REORG START','REORG RESUME')) as b

on (a.TABNAME = b.TABNAME and a.end\_time <= b.start\_time)

where b.TABNAME is null;

--Average time taken (in minutes) and the maximum time taken for full backups

SELECT AVG(TIMESTAMPDIFF(4,CHAR(TIMESTAMP(END\_TIME)- TIMESTAMP(START\_TIME) ))) AS AVG\_BTIME,

MAX(TIMESTAMPDIFF(4,CHAR(TIMESTAMP(END\_TIME)- TIMESTAMP(START\_TIME)))) AS MAX\_BTIME

FROM table(admin\_list\_hist())as a WHERE OPERATION = 'B' AND OPERATIONTYPE = 'F'

--View Lock

SELECT SMALLINT(AGENT\_ID) AS WAITING\_ID, SUBSTR(APPL\_NAME, 1,10) AS WAITING\_APP,

SUBSTR(AUTHID,1,10) AS WAITING\_USER,

SMALLINT(AGENT\_ID\_HOLDING\_LK) AS HOLDER\_ID,

LOCK\_MODE AS HELD, LOCK\_OBJECT\_TYPE AS TYPE,

LOCK\_MODE\_REQUESTED AS REQUEST

from TABLE(SNAPSHOT\_LOCKWAIT('',-1)) as lw

--1) List Backups

--This lists all the backups you have taken, the elapsed time (in minutes) to complete the backup, the number of tablespaces backed up, the type of backup (online, offline, full, incremental or delta), and the date and time completed.

select timestampdiff(4,char(timestamp(end\_time)-timestamp(start\_time))) as "Elapsed Time (min)",

substr(firstlog,1,13) as "Start Log", substr(lastlog,1,13) as "End Log",num\_tbsps as "Number Tbspcs",

case(operationType)

when 'F' then 'Offline Full'

when 'N' then 'Online Full'

when 'I' then 'Offline Incremental'

when 'O' then 'Online Incremental'

when 'D' then 'Offline Delta'

when 'E' then 'Online Delta'

else '?' end as Type,

date(timestamp(end\_time)) as "Day Completed",

time(timestamp(end\_time)) as "Time Completed"

from table(admin\_list\_hist()) as lh where operation = 'B';

--2) List logs archived

--This query lists the last 10 log files archived. You could start to get fancy here and see how much log you are archiving per given unit of time to ensure you have enough archive capacity should your workload suddenly double (but I'll save that query for another time…or leave it as an exercise for you).

select date(timestamp(end\_time)) as "Date",

time(timestamp(end\_time)) as "Time",

substr(firstlog,1,13) as LogFile

from table(admin\_list\_hist()) as lh

where operation = 'X' and end\_time is not null

order by timestamp(end\_time) desc

fetch first 10 rows only;

--3) List tables loaded

--This query lists the last 10 tables loaded and how long the loads ran for.

select timestampdiff(4,char(timestamp(end\_time)-timestamp(start\_time))) as "Elapsed Time (min)",

date(timestamp(end\_time)) as "Date",time(timestamp(end\_time)) as "Time",

case(operationType)

when 'R' then 'Replace'

when 'I' then 'Insert'

end as "Load Type",

substr(tabschema,1, 20) as Schema,

substr(tabname,1,30) as Table

from table(admin\_list\_hist()) as lh

where operation = 'L'

order by timestamp(end\_time) desc

fetch first 10 rows only;

--4) Find the dropped table and the DDL to recreate them

--This query will list the 10 most recent tables dropped, when they were dropped and the DDL that can be used to recreate the tables (just in case :-).

select date(timestamp(end\_time)) as "Date",

time(timestamp(end\_time)) as "Time",substr(tabschema,1, 13) as Schema,

substr(tabname,1,13) as Table,cmd\_text as "Recreate DDL"

from list\_history

where operation = 'D'

order by timestamp(end\_time) desc

fetch first 20 rows only;

--5) Instance info

select substr(inst\_name,1,10) as "Instance Name", num\_dbpartitions as "Number Partitions",

rtrim(char(inst\_ptr\_size)) ||'bit' as "Bitness", fixpack\_num as Fixpack

from table(env\_get\_inst\_info()) as ii;

--6) System info

select substr(os\_name,1,15) as "Operating System", substr(rtrim(os\_version) || '.' || rtrim(os\_release),1,20) as "OS Version",

substr(host\_name, 1,15) as "Hostname", total\_cpus, total\_memory as "Memory (Meg)"

from table(env\_get\_sys\_info()) as si

--7) Product info

select case(installed\_prod)

when 'ADCL' then 'DB2 Application Development Client'

when 'ADMCL' then 'DB2 Administration Client'

when 'CONEE' then 'DB2 Connect Enterprise Edition'

when 'CONPE' then 'DB2 Connect Personal Edition'

when 'CUBE' then 'DB2 Cube Views'

when 'DLM' then 'DB2 Data Links Manager'

when 'ESE' then 'DB2 Enterprise Server Edition'

when 'EXP' then 'DB2 Express Edition'

when 'GSE' then 'DB2 Spatial Extender'

when 'PE' then 'DB2 Personal Edition'

when 'QP' then 'DB2 Query Patroller'

when 'RTCL' then 'DB2 Run-Time Client'

when 'WM' then 'DB2 Warehouse Manager'

when 'WSE' then 'DB2 Workgroup Server Edition'

when 'WSUE' then 'DB2 Workgroup Server Unlimited Edition'

else '?' end as "Product",

case(is\_licensed)

when 1 then 'Yes'

when 0 then 'NO' end as "Licensed",

prod\_release

from table(env\_get\_prod\_info()) as pi ;

--8) List the registry variables in use for this database.

select substr(reg\_var\_name, 1, 20) as Variable, substr(reg\_var\_value, 1, 35) as Value,

case(level)

when 'I' then 'Instance'

when 'G' then 'Global'

when 'N' then 'DB Partition'

when 'E' then 'Environment' end as Level

from table(REG\_LIST\_VARIABLES()) as reg

order by Level, Variable;

--9) Administration using ADMIN\_CMD

call ADMIN\_CMD('runstats on table cudb27.employee and indexes all')

--Following commands can also be using this procedure:

--DESCRIBE

--EXPORT

--PRUNE HISTORY/LOGFILE

--REORG INDEXES/TABLE

--RUNSTATS

--UPDATE DATABASE CONFIGURATION

--11) Finding long running queries and long lock waits

select \* from SYSIBMADM.LOCKWAITS where authid <> 'db2inst1'"

-- Find slow running queries  
  
SELECT SUBSTR(ELAPSED\_TIME\_SEC,1,10) as ELAPSED\_TIME\_SEC, ACTIVITY\_TYPE, SUBSTR(STMT\_TEXT,1,500) as STMT\_TEXT FROM SYSIBMADM.MON\_CURRENT\_SQL where ELAPSED\_TIME\_SEC > 2 ORDER BY ELAPSED\_TIME\_SEC DESC FETCH FIRST 50 ROWS ONLY;  
  
-- stmt\_exec\_time is in milliseconds  
SELECT SUBSTR(STMT\_TEXT,1,1000) as STMT, num\_executions, stmt\_exec\_time FROM table(MON\_GET\_PKG\_CACHE\_STMT(NULL,NULL,NULL,-2)) where stmt\_exec\_time >5000 order by stmt\_exec\_time desc fetch first 50 rows only;

**INCREASE BUFFERPOOL SIZE**

CONNECT TO EMSSCDB4;

ALTER BUFFERPOOL IBMDEFAULTBP IMMEDIATE SIZE 10500;

CONNECT RESET;

**Long running queries > 4 minutes:**

SELECT ELAPSED\_TIME\_MIN,SUBSTR(AUTHID,1,10) AS AUTH\_ID, AGENT\_ID,APPL\_STATUS,STMT\_TEXT AS SQL\_TEXT FROM SYSIBMADM.LONG\_RUNNING\_SQL

WHERE ELAPSED\_TIME\_MIN > 0 ORDER BY ELAPSED\_TIME\_MIN DESC

SELECT 'db2 "force application(' || rtrim(char(AGENT\_ID)) || ')";'

--'')' || '";'

FROM SYSIBMADM.LONG\_RUNNING\_SQL

WHERE

STMT\_TEXT like 'SELECT AT.DOCUMENTTYPE , AT.EMDOCUMENTNAMECODE , ALTNM.FULLNAME , AT.RECEIPTDATE , CRC.COMMUNICATIONID , AT.EMDOCITEMTYPE%' and

ELAPSED\_TIME\_MIN > 4 ORDER BY ELAPSED\_TIME\_MIN DESC;

commit;

\

select NUM\_EXECUTIONS,(TOTAL\_EXEC\_TIME/NUM\_EXECUTIONS) as AVERAGEEXECTIME,STMT\_TEXT from table(SNAPSHOT\_DYN\_SQL('',-1)) as SNAPDYN where NUM\_EXECUTIONS > 0 order by AVERAGEEXECTIME desc

select

agent\_id,stmt\_start,stmt\_stop,(stmt\_stop-stmt\_start) as exectime,stmt\_text from table(snapshot\_statement('',-1)) as stmt where stmt\_start is not null or stmt\_stop is not null order by exectime desc

**Long running Query:**

WITH a AS (SELECT a.application\_handle, a.system\_auth\_id, a.application\_name, a.client\_wrkstnname, a.workload\_name, a.workload\_occurrence\_state, b.activity\_id, b.uow\_id, b.local\_start\_time, b.activity\_state, b.activity\_type, b.total\_cpu\_time, b.rows\_read, b.rows\_returned, b.query\_cost\_estimate, b.direct\_reads, b.direct\_writes

FROM TABLE(wlm\_get\_service\_class\_workload\_occurrences\_v97(NULL, NULL, NULL)) a

LEFT JOIN TABLE(wlm\_get\_workload\_occurrence\_activities\_v97(NULL, NULL)) b ON a.application\_handle = b.application\_handle),

b AS (SELECT t.\*

FROM TABLE(mon\_get\_activity\_details(FOR EACH ROW OF (SELECT application\_handle, uow\_id, activity\_id, NULL FROM a WHERE activity\_id > 0))) t),

c AS (SELECT x."activity\_id" activity\_id, x."uow\_id" uow\_id, x."stmt\_text" stmt\_text

FROM b,

XMLTABLE(XMLNAMESPACES(DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'), '$m/db2\_activity\_details' PASSING XMLPARSE(DOCUMENT b.details) AS "m" COLUMNS

"activity\_id" INTEGER PATH 'activity\_id',

"uow\_id" INTEGER PATH 'uow\_id',

"stmt\_text" VARCHAR(1024) PATH 'stmt\_text') x)

SELECT a.\*, TRANSLATE(c.stmt\_text, ' ', CHR(10)) stmt\_text

FROM a

LEFT JOIN c ON a.activity\_id = c.activity\_id AND a.uow\_id = c.uow\_id

WHERE stmt\_text IS NULL OR SUBSTR(stmt\_text, 1, 40) <> 'WITH a AS (SELECT a.application\_handle, '

ORDER BY application\_handle, a.activity\_id

select

case

when st.stmt\_stop is null then int(TIMESTAMPDIFF (4, char(st.snapshot\_timestamp - st.stmt\_start)))

else int(st.stmt\_elapsed\_time\_s/60)end as "Elapsed (min)",

case (ai.appl\_status)

when 3 then 'Executing'

when 4 then 'Wait on User'

when 5 then 'Lock Wait'

when 7 then 'Rolling Back'

else 'Other '||substr(char(appl\_status),1,2) end as "Status",

substr(ai.auth\_id,1,10) as "Auth ID",

substr(ap.inbound\_comm\_address,1,15) as "IP Addr",

substr(st.stmt\_text,1,100) as "SQL Text"

from TABLE(SNAPSHOT\_STATEMENT('',-1)) as st, TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai,

TABLE(SNAPSHOT\_APPL('',-1)) as ap

where st.agent\_id = ai.agent\_id and ai.agent\_id = ap.agent\_id

order by 1 desc;

--12) Finding lock wait chains

select substr(ai\_h.appl\_name,1,10) as "Hold App",

substr(ai\_h.auth\_id,1,10) as "Holder",

substr(ai\_w.appl\_name,1,10) as "Wait App",

substr(ai\_w.auth\_id,1,10) as "Waiter",

case(lw.lock\_mode)

when 0 then 'N'

when 1 then 'IS'

when 2 then 'IX'

when 3 then 'S'

when 4 then 'SIX'

WHEN 5 then 'X'

when 6 then 'IN'

when 7 then 'Z'

when 8 then 'U'

when 9 then 'NKS'

when 10 then 'NKX'

when 11 then 'WX'

when 12 then 'NKW'

else '?' end as "Hold Mode",

case(lw.lock\_object\_type) when 1 then 'Table' when 2 then 'Row' when 6 then 'Key'

else 'Other' || substr(char(lw.lock\_object\_type),1,2) end as "Obj Type",

substr(lw.table\_name,1,10) as "TabName", substr(lw.table\_schema,1,10) as "Schema",

timestampdiff(2,char(lw.snapshot\_timestamp - lw.lock\_wait\_start\_time)) as "waiting (s)"

from TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai\_h,

TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai\_w,

TABLE(SNAPSHOT\_LOCKWAIT('',-1)) as lw

where lw.agent\_id = ai\_w.agent\_id and lw.agent\_id\_holding\_lk = ai\_h.agent\_id

--12)

select 1 as row, lock\_list\_in\_use, appls\_cur\_cons as NumCons from TABLE(SNAPSHOT\_DATABASE ('',-1)) as DatabaseSnapshot;

select 1 as row, lock\_list\_in\_use, appls\_cur\_cons as NumCons from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap;

select \* from TABLE(SNAPSHOT\_DATABASE ('',-1)) as DatabaseSnapshot;

select \* from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap

call sysproc.get\_db\_config();

with dbcfg as( select 1 as row, float(locklist\*4096) as locklist,

float(maxlocks) as maxlocksfrom session.db\_config where dbconfig\_type = 1),

dbsnap as( select 1 as row, lock\_list\_in\_use, appls\_cur\_cons as NumCons

--from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap )

from TABLE(SNAPSHOT\_DATABASE ('',-1)) as dbsnap)

select dec((lock\_list\_in\_use/locklist)\*100,4,1) as "% Lock List",

dec((lock\_list\_in\_use/(locklist\*(maxlocks/100))\*100),4,1) as "% to Maxlock",

NumCons as "Number of Cons", lock\_list\_in\_use/numcons as "Avg Lock Mem Per Con (bytes)"

from dbcfg c, dbsnap s

where c.row = s.row;

--12) Performance (hit ratios, sorts, dynamic queries)

--Bufferpool hit ratio

with bpsnap as(select bp\_name, float(pool\_data\_p\_reads) as dp\_read, float(pool\_data\_l\_reads) as dl\_read, float(pool\_index\_p\_reads) as ip\_read, float(pool\_index\_l\_reads) as il\_read

from table (snapshot\_bp ('', -1) ) as snapshot\_bp)

select bp\_name, dec((1 -(dp\_read / dl\_read))\*100,4,1) as "Data Hit Ratio", dec((1 -(ip\_read / il\_read))\*100,4,1) as "Index Hit Ratio"from bpsnap;

--Sorts

with dbmcfg as( select 1 as row,float(sheapthres\*4096) as sheapthres FROM TABLE(GET\_DBM\_CONFIG()) AS DBMCFG where dbmconfig\_type = 1),

dbsnap as( select 1 as row, float(sort\_heap\_allocated\*4096) as sheap\_alloc, int(sort\_overflows) as overflows,int(active\_sorts) as active

from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap )

select dec((sheap\_alloc/sheapthres)\*100,4,1) as "% SHeap Alloc", overflows as "Sort Overflows", active as "Active Sorts"

from dbmcfg c, dbsnap s

where c.row = s.row;

--Top five most frequently executed SQL statements from the package cache

select int(num\_executions) as "Num Execs",

int(total\_exec\_time/num\_executions) as "Avg Time (sec)",

int(stmt\_sorts) as "Num Sorts",

int(stmt\_sorts/num\_executions) as "Sorts Per Stmt",

substr(stmt\_text,1,25) as "SQL Stmt"

FROM TABLE(SNAPSHOT\_DYN\_SQL('', -1)) as dynSnapTab

where num\_executions > 0

order by 1 desc

fetch first 5 rows only;

--Top 5 queries with the longest average execution time

select int(num\_executions) as "Num Execs",

int(total\_exec\_time/num\_executions) as "Avg Time (sec)",

int(stmt\_sorts) as "Num Sorts",

int(stmt\_sorts/num\_executions) as "Sorts Per Stmt",

substr(stmt\_text,1,25) as "SQL Stmt"

FROM TABLE(SNAPSHOT\_DYN\_SQL('', -1)) as dynSnapTab

where num\_executions > 0

order by 2 desc

fetch first 5 rows only;

--13) How to easily populate a table with random data

CREATE TABLE EMPLOYEE (EMP\_ID integer,DATE\_OF\_BIRTH date,SALARY integer,EMP\_NAME char(10));

delete from EMPLOYEE;

--select count(\*) from EMPLOYEE;

INSERT INTO EMPLOYEE (EMP\_ID, DATE\_OF\_BIRTH, SALARY, EMP\_NAME)

WITH EMP\_IDS(EMP\_ID) AS( VALUES(1)

UNION ALL

SELECT EMP\_ID+1 FROM EMP\_IDS WHERE EMP\_ID < 1000 )

SELECT EMP\_ID, CURRENT DATE - ((18 \* 365) + RAND()\*(47\*365)) DAYS, INTEGER(50000 + RAND()\*90000),

TRANSLATE ( CHAR(BIGINT(RAND() \* 1000000000 )), 'Diwakar', '1234567890' )FROM EMP\_IDS;

select \* from EMPLOYEE;

--create table tempcase like emrmrpt.t\_em\_caseheader

delete from tempcase;

INSERT INTO tempcase (appealindicator,caseid,casereference, casetypecode,concernroleid,ownerid,startdate,versionno,lastwritten)

WITH CASE\_IDS(caseid,concernroleid) AS( VALUES(1,1)

UNION ALL

SELECT caseid+1,concernroleid+1 FROM CASE\_IDS WHERE CASEID < 1000 )

SELECT '0', caseid, char(caseid),'CT5', concernroleid,'superuser',CURRENT DATE - ((18 \* 365) + RAND()\*(47\*365)) DAYS,1,CURRENT TIMESTAMP FROM CASE\_IDS;

--

select \* from randomid;

create table randomid (randomid decimal(10,2));

insert into randomid (randomid)

WITH RAND\_IDS(rand\_id) AS( VALUES(1)

UNION ALL

SELECT

DECIMAL(RAND()\*100000,7,2)

FROM RAND\_IDS WHERE rand\_id < 1000 )

SELECT rand\_id END FROM RAND\_IDS;

WITH RAND\_NUMS(rand\_id) AS( VALUES(1)

UNION ALL

SELECT DECIMAL(RAND()\*100000,7,2)

FROM RAND\_NUMS WHERE rand\_id < 100 )

SELECT rand\_id

END FROM RAND\_NUMS;

WITH RAND\_NUMS(rand\_id,num\_rows) AS( VALUES(1,1)

UNION ALL

SELECT INTEGER(RAND()\*10), num\_rows+1

FROM RAND\_NUMS WHERE NUM\_ROWS < 100 )

SELECT CASE rand\_id

WHEN 0 THEN 'Tampa'

WHEN 1 THEN 'Rome'

WHEN 2 THEN 'Denver'

WHEN 3 THEN 'Berlin'

WHEN 4 THEN 'Orlando'

WHEN 5 THEN 'Prague'

WHEN 6 THEN 'Las Vegas'

WHEN 7 THEN 'Nice'

WHEN 8 THEN 'San Diego'

WHEN 9 THEN 'Lisbon'

END FROM RAND\_NUMS;

SELECT SNAP\_DATE, APP\_NAME, AVG(LOCK\_WAIT\_TIME) AS "AVG WAIT(ms)", SUM(LOCK\_WAIT\_TIME) AS "TOT WAIT(ms)", SUM(DEADLOCKS) AS "TOT DL", AVG(DEADLOCKS) AS "AVG DL"

FROM LOCK\_SNAP

GROUP BY CUBE(SNAP\_DATE, APP\_NAME)

--To check which of the three things listed above is causing your page cleaners to fire run the following script.

with dbsnap as(select float(pool\_drty\_pg\_steal\_clns) as pg\_steal,

float(pool\_drty\_pg\_thrsh\_clns) as chg\_pg\_thresh, float(pool\_lsn\_gap\_clns) as softmax,

pool\_drty\_pg\_steal\_clns + pool\_drty\_pg\_thrsh\_clns + pool\_lsn\_gap\_clns as total\_clns

--from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap )

from TABLE(SNAPSHOT\_DATABASE ('',-1)) as dbsnap)

select dec((pg\_steal / total\_clns)\*100,4,1) as "% Steals", dec((chg\_pg\_thresh / total\_clns)\*100,4,1) as "% Thresh",

dec((softmax / total\_clns)\*100,4,1) as "% Softmax" from dbsnap;

--Speaking of aggression, here is a way to check if your prefetchers are being too aggressive.

with bpsnap as(select bp\_name, unread\_prefetch\_pages, pool\_async\_data\_reads + pool\_async\_index\_reads as Async,

pool\_data\_p\_reads + pool\_index\_p\_reads as TotReads

from table (snapshot\_bp ('', -1) ) as snapshot\_bp)

select bp\_name,

dec(100 \* (TotReads - Async) / TotReads,4,1) as "% Sync Reads", dec(100 \* unread\_prefetch\_pages / TotReads,4,1) as "% Async not read"

from bpsnap;

--Here is a script to monitor your log space utilization.

select int(total\_log\_used/1024/1024) as "Log Used (Meg)",

int(total\_log\_available/1024/1024) as "Log Space Free (Meg)",

int((float(total\_log\_used) / float(total\_log\_used+total\_log\_available))\*100) as "Pct Used",

int(tot\_log\_used\_top/1024/1024) as "Max Log Used (Meg)", int(sec\_log\_used\_top/1024/1024) as "Max Sec. Used (Meg)",

int(sec\_logs\_allocated) as "Secondaries"

from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap;

--from TABLE(SNAPSHOT\_DATABASE ('',-1)) as dbsnap

--Find the log hog

select case (ai.appl\_status)

when 3 then 'Executing'

when 4 then 'Wait on User'

when 5 then 'Lock Wait'

when 7 then 'Rolling Back'

else 'Other '||substr(char(appl\_status),1,2) end as "Status",

substr(ai.auth\_id,1,10) as "Authid",

substr(ai.appl\_name,1,15) as "Appl Name",

int(ap.UOW\_LOG\_SPACE\_USED/1024/1024) as "Log Used (M)",

int(ap.appl\_idle\_time/60) as "Idle for (min)",

ap.appl\_con\_time as "Connected Since"

from TABLE(SNAP\_GET\_DB ('',-1)) as db,

TABLE(SNAPSHOT\_APPL('',-1)) as ap,

TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai

where ai.agent\_id = db.APPL\_ID\_OLDEST\_XACT and ap.agent\_id = ai.agent\_id;

--Show the amount of free space available in each tablespaces.

select substr(tablespace\_name,1,30) as "Tablespace Name",

case (tablespace\_type)

when 0 then 'DMS'

else 'SMS' end as "Type",

case (tablespace\_state)

when 0 then 'Normal'

else 'Other ' || char(tablespace\_state) end as "State",

int((total\_pages\*page\_size)/1024/1024) as "Size (Meg)",

smallint((float(free\_pages) / float(total\_pages))\*100) as "% Free Space",

int((free\_pages\*page\_size) / 1024 / 1024) as "Meg Free Space"

from TABLE(SNAPSHOT\_TBS\_CFG('',-1)) as tbcfgsnap

--

--Database Health

call sysproc.get\_db\_config();

with bpsnap as

(

select bp\_name,dec((1 - (float(pool\_data\_p\_reads + pool\_index\_p\_reads)/float(pool\_data\_l\_reads + pool\_index\_l\_reads) ))\*100,3,1) as bphr from table (snapshot\_bp ('', -1) ) as snapshot\_bp

where pool\_data\_l\_reads > 0 or pool\_index\_l\_reads > 0

),

-- bufferpool with the best hit ratio

maxbphr as

(

select bp\_name,bphr from bpsnap where bphr = (select max(bphr) from bpsnap)

),

-- bufferpool with the worst hit ratio

minbphr as

(

select bp\_name,bphr from bpsnap where bphr = (select min(bphr) from bpsnap)

),

dbcfg as

( select float(50\*4096) as locklist from session.db\_config where dbconfig\_type = 1

--from sysibm.sysdummy1

),

-- get the lock wait chains

lock\_snap as

( select dec((lock\_list\_in\_use/locklist)\*100,3,1) as pctlocklist from TABLE(SNAP\_GET\_DB ('',-1)) as dbsnap, dbcfg

),

lock\_wait\_snap as

( select substr(ai\_h.auth\_id,1,10) as auth\_holder,substr(ai\_w.auth\_id,1,10) as auth\_waiter,timestampdiff(4,char(lw.snapshot\_timestamp - lw.lock\_wait\_start\_time)) as waiting

from TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai\_h, TABLE(SNAPSHOT\_APPL\_INFO('',-1)) as ai\_w, TABLE(SNAPSHOT\_LOCKWAIT('',-1)) as lw

where lw.agent\_id = ai\_w.agent\_id and lw.agent\_id\_holding\_lk = ai\_h.agent\_id

),

max\_lock\_wait as

( select auth\_holder, auth\_waiter, char(waiting) as wait\_time from lock\_wait\_snap where waiting = (select max(waiting) from lock\_wait\_snap)

),

final\_union as

(

select 1 as o\_by,'DBM Status' as Category,'-' as Element,case (db2\_status) when 0 then 'Active' else 'Quiesced' end as Value FROM TABLE(SNAPSHOT\_DBM(0)) AS DBM

union all

select 2 as o\_by,'Database Uptime' as Category,'-' as Element, char(date(db\_conn\_time)) as Value from TABLE(SNAP\_GET\_DB ('',-1)) as db

union all

select 3 as o\_by,'Best BP Hit Ratio' as category,bp\_name as Element,char(bphr) as Value from maxbphr

union all

select 4 as o\_by,'Worst BP Hit Ratio' as category,bp\_name as Element,char(bphr) as Value from minbphr

union all

select 5 as o\_by,'Percentage Lock List in Use' as category,'-' as Element,char(pctlocklist) as Value from lock\_snap

union all

select 6 as o\_by,'Log Space Used' as category,'Meg' as Element, char(total\_log\_used/1024/1024) as value from TABLE(SNAP\_GET\_DB ('',-1)) as db

union all

select 7 as o\_by,'Log Space Free' as category,'Meg' as Element, char(total\_log\_available/1024/1024) as value from TABLE(SNAP\_GET\_DB ('',-1)) as db

union all

select 8 as o\_by,'Holder of longest lock wait' as category,auth\_holder as Element,wait\_time || 'min' as value from max\_lock\_wait

union all

select 9 as o\_by,'Waiter of longest lock wait' as category,auth\_waiter as Element,wait\_time || 'min' as value from max\_lock\_wait

union all

select 10 as o\_by,'DB2' as category,'Fastest' as element,'Simple' as value from sysibm.sysdummy1

union all

select 11 as o\_by,'Oracle' as category, 'Slow' as element, 'Complex' as Value from sysibm.sysdummy1

union all

select 12 as o\_by, 'MS' as category, 'Slower' as element, 'Not Scalable' as value from sysibm.sysdummy1

)

-- finally select from the union all CTE in the order I want

select Category,Element, Value from final\_union order by o\_by;

**Below are the important db2 admin views(version 9)**

**REORG---->>SELECT SUBSTR(TABNAME, 1, 15) AS TAB\_NAME, SUBSTR(TABSCHEMA, 1, 15) AS TAB\_SCHEMA, REORG\_PHASE, SUBSTR(REORG\_TYPE, 1, 20) AS REORG\_TYPE,**

**REORG\_STATUS, REORG\_COMPLETION, DBPARTITIONNUM FROM SYSIBMADM.SNAPTAB\_REORG ORDER BY DBPARTITIONNUMLOCK WAIT--->>>SELECT AGENT\_ID, LOCK\_MODE, LOCK\_OBJECT\_TYPE, AGENT\_ID\_HOLDING\_LK,**

**LOCK\_MODE\_REQUESTED FROM SYSIBMADM.SNAPLOCKWAIT WHERE DBPARTITIONNUM = 0BPHIT RATIO ------>>>SELECT SUBSTR(DB\_NAME,1,8) AS DB\_NAME, SUBSTR(BP\_NAME,1,14) AS BP\_NAME, TOTAL\_HIT\_RATIO\_PERCENT, DATA\_HIT\_RATIO\_PERCENT,**

**INDEX\_HIT\_RATIO\_PERCENT, XDA\_HIT\_RATIO\_PERCENT, DBPARTITIONNUM FROM SYSIBMADM.BP\_HITRATIO ORDER BY DBPARTITIONNUMTOP DYNAMIC SQL ------>>>SELECT NUM\_EXECUTIONS, AVERAGE\_EXECUTION\_TIME\_S, STMT\_SORTS,**

**SORTS\_PER\_EXECUTION, SUBSTR(STMT\_TEXT,1,60) AS STMT\_TEXT FROM SYSIBMADM.TOP\_DYNAMIC\_SQL ORDER BY NUM\_EXECUTIONS DESC FETCH FIRST 5 ROWS ONLYUTILITY PROGRESS ------->>SELECT UTILITY\_ID, PROGRESS\_TOTAL\_UNITS, PROGRESS\_COMPLETED\_UNITS,**

**DBPARTITIONNUM FROM SYSIBMADM.SNAPUTIL\_PROGRESSMONITOR LOG UTILIZATION ----------->>SELECT \* FROM SYSIBMADM.LOG\_UTILIZATION**

BACKUP AND RESTORE WITH ROLLFORWARD COMMANDS:

1)Checking for return code after the backup and alerting incase of backup failure

RC=$?

If [$RC != 0 ] then echo "backup failure" | mail inaganti@us.ibm.com

2)Changing the backup mechanism to **include logs in the backup.**This will help in faster recovery in cases when the current active log is corrupt.I tested this on dev server on a SAMPLE database and it worked perfectly fine.I believe this would be a good change for all the future recovery scenarios.Listed below are the steps.

db2 "backup db SAMPLE online use TSM **include logs**"

db2adutl extract taken at 20080801104054

db2 "restore database SAMPLE taken at 20080801104054 **logtarget /container2/srini/logs**"

db2 "rollforward db SAMPLE to end of logs and complete **overflow log path (/container2/srini/logs**)”

3)Verifying the backup image using the db2adutl utility and also dbckbkp if possible.Also alerting the DBA if the verification fails for some reason

4)Maintaining the latest backup on disk would help as extract from TSM takes longer time

**Extracting Backup image file form TSM:**

**db2adutl query full database <databasename>  
  
then, find desired backup in list, it has a timestamp in its name.  
  
db2adutl extract full taken at <timestamp> database <databasename>  
  
similar goes for logs.**

**To remove tablespace from backup pending state:**

**db2 "backup db empscdb2 tablespace EMPTSTAB2 to /dev/null without prompting"**

**CHECK PENDING STATE**

**To remove from check pending state to normal command**

**set integrity for id.caseheader immediate checked**

**LOBS export**

**export to '/primlogs/backup/jan2409/task.ixf' of ixf lobs to /primlogs/backup/jan2409 lobfile task**

**modified by lobsinfile messages task.out**

**select \* from SCHEMANAME.task where status = 'WS7' and date(lastwritten) < '07-01-2008' ;**

**Foreign key disabling:**

**alter table EMSSCWU2.ADJUSTMENTINSTRUCTION alter foreign key SQL080310194412561 NOT ENFORCED;**

SELECT 'SET CONSTRAINTS FOR SCHEMANAME.' || TABNAME || ' FOREIGN KEY,CHECK,SUMMARY,GENERATED COLUMN IMMEDIATE UNCHECKED;' FROM SYSCAT.TABLES WHERE STATUS <> 'N'

Db2bind applications :

/SQLLIB/bnd/db2 connect to curamdb3 user udb3 using f7YUPiTj

/SQLLIB/bnd/db2 bind @db2ubind.lst blocking all grant public

/SQLLIB/bnd/db2 bind @db2cli.lst blocking all grant public

/SQLLIB/bnd/db2 bind db2schema.bnd blocking all grant public

**Kasi,**

**RESTORE COMMAND**

**db2 "RESTORE database empscdb1 from /primlogs1 taken at 20080531092016 without rolling forward"**

**nohup db2 "RESTORE database empscdb1 from /primlogs1 taken at 20080531092016 without rolling forward" &**

**nohup db2 "RESTORE database empscdb1 from /primlogs1 taken at 20080531092016 without rolling forward" &**

**REDIRECTED RESTORE very very important steps:**

This was tested in 9.73.99.69 server

====================================

db2 "RESTORE database empscdb1 from /primlogs1 taken at 20080531092016 without rolling forward"

nohup db2 "RESTORE database empscdb1 from /primlogs1 taken at 20080531092016 without rolling forward" &

SET TABLESPACE CONTAINERS FOR 0 USING (PATH "/container61/EMPSCDB2/NODE0000/SQL00001/SQLT0000.0");

SET TABLESPACE CONTAINERS FOR 1 USING (PATH "/container61/EMPSCDB2/NODE0000/SQL00001/SQLT0001.0");

SET TABLESPACE CONTAINERS FOR 2 USING (PATH "/container61/EMPSCDB2/NODE0000/SQL00001/SQLT0002.0");

other tablespaces containers ..... here

RESTORE DATABASE testdb CONTINUE;

/home/db2inst1/db2inst1/NODE0000/SQL00002/SQLT0000.0

/home/db2inst1/db2inst1/NODE0000/SQL00002/SQLT0001.0

/home/db2inst1/db2inst1/NODE0000/SQL00002/SQLT0002.0

====================

1. Take backup of curam database

db2 backup db curam to /home/db2inst1/db2dbbkp

2. List history

db2 list history backup all for database curam

Here is the timestamp of database backup

20070217011652

3. Find out containers name from source database

db2 list tablespaces

Above command will tell you container details for curam database tablespaces.

4. List containers for each tablespace

db2 list tablespace containers for 0

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0000.0

db2 list tablespace containers for 1

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0001.0

db2 list tablespace containers for 2

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0002.0

db2 list tablespace containers for 3

/home/db2inst1/curam/a

db2 list tablespace containers for 4

/home/db2inst1/curam/b

db2 list tablespace containers for 5

/home/db2inst1/db2inst1/NODE0000/SQL00001/SYSTOOLSPACE

db2 list tablespace containers for 6

/home/db2inst1/curam/c0

db2 list tablespace containers for 7

/home/db2inst1/curam/d

db2 list tablespace containers for 8

/home/db2inst1/curam/e

make directory

4. Restore database to new database [curamtst]

RESTORE database curam user db2inst1 using db2inst1 taken at 20070216111246 to /home/db2inst1/curamtest into curamtest redirect;

OR

RESTORE database curam user db2inst1 using db2inst1 from /home/db2inst1/db2dbbkp to /home/db2inst1/curamtest into curamtest redirect;

5. Force application

db2 force application all;

6. Set tablespace containers to new database

SET TABLESPACE CONTAINERS FOR 0 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0000.0");

SET TABLESPACE CONTAINERS FOR 1 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0001.0");

SET TABLESPACE CONTAINERS FOR 2 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0002.0");

SET TABLESPACE CONTAINERS FOR 3 USING (PATH "/home/db2inst1/curamtest/curam/a");

SET TABLESPACE CONTAINERS FOR 4 USING (PATH "/home/db2inst1/curamtest/curam/b");

SET TABLESPACE CONTAINERS FOR 5 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SYSTOOLSPACE");

SET TABLESPACE CONTAINERS FOR 6 USING (PATH "/home/db2inst1/curamtest/curam/c0");

SET TABLESPACE CONTAINERS FOR 7 USING (PATH "/home/db2inst1/curamtest/curam/d");

SET TABLESPACE CONTAINERS FOR 8 USING (PATH "/home/db2inst1/curamtest/curam/e");

7. Continue database restore

RESTORE DATABASE curam CONTINUE;

8. The database [curamtst] will be in rollforward pending status since LOGRETAIN=RECOVERY, USEREXIT=ON. The rollforward pending status must be removed on target database before accessing it.

db2 rollforward db curamtst user db2inst1 using db2inst1 complete

9. Connect to target database

db2 connect to curamtst

**TRANSLATE FUNCTION:**

**SELECT zipcode, '' as zip\_code FROM SCHEMANAME.tablename where length(rtrim(ltrim(TRANSLATE (upper(task\_name), '', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ')))) < 5**

**CASE STATEMENTS:**

)SELECT pub\_name,   
  CASE WHEN state is NULL or state = ''  
       THEN 'Not supplied'  
       ELSE state  
  END  
FROM publishers

2)update titles  
   set price =  
   case  
   when type = "business"  
       then price \* 0.75  
   when pub\_id = "0736"  
       then price \* 0.9  
   end  
where pub\_id = "0736" OR  
type = "business"

**force applications of a particular database on an instance with multiple databases**

If you want to force applications of a particular database on an instance with multiple databases use the following code

for j  in `db2 list applications show detail | grep -i SAMPLEDB | awk '{print $3}'`

do

db2 "force application($j)"

done

**To find values for sequence :**

db2 "VALUES PREVVAL FOR CAPTIVA.ASYNCHCORRDOCDETAILSID\_SEQ"

**To terminate rollforward pending status:**

db2 rollforward –d databasename complete

**To find the rollforward status:**

**Db2pd –d databasename –rec status**

**db2pd -d empscdb1 -reorg**

**DB2 INSTALLATION ON LINUX**

1. Check for the available space in the file system

**Command$ df –h**

Filesystem Size Used Avail Use% Mounted on

/dev/sda5 15G 2.6G 11G 19% /

/dev/sda1 99M 15M 79M 16% /boot

none 16G 0 16G 0% /dev/shm

/dev/sda3 4.9G 131M 4.5G 3% /var

/dev/mapper/W0AA0155\_dg-storage

145G 92M 138G 1% /db2

nas1:/shares/Volume1/storage

2.0T 1.9T 75G 97% /storage

1. Locate the installer directory and do “**ls –la”**

**Command$ ls –la**

total 40

drwxr-xr-x 4 1009 1009 4096 Aug 9 2007 .

drwxr-xr-x 3 1009 1009 4096 Aug 9 2007 ..

**drwxr-xr-x 6 1009 1009 4096 Aug 9 2007 db2**

**-rwxr-xr-x 1 1009 1009 5090 Apr 24 2007 db2\_install**

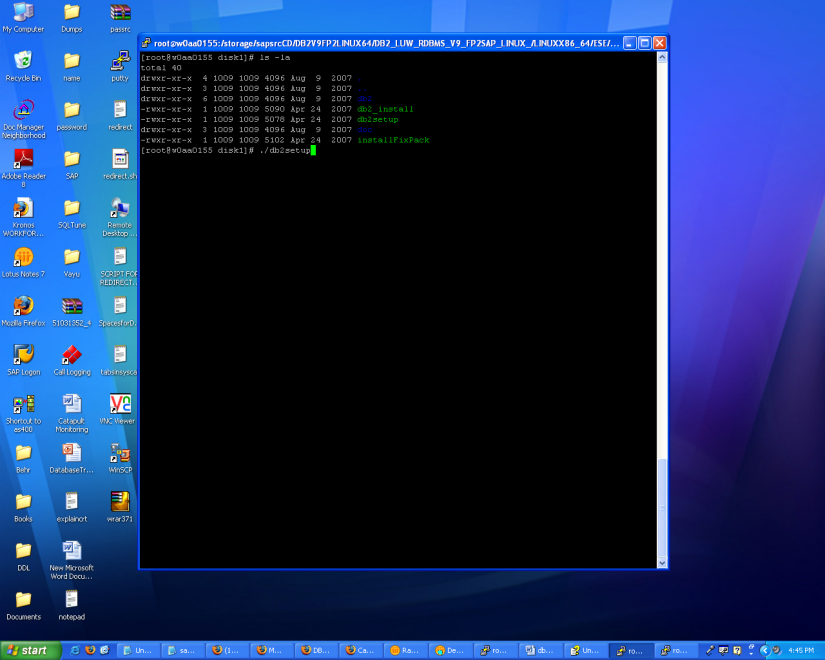
**-rwxr-xr-x 1 1009 1009 5078 Apr 24 2007 db2setup**

**drwxr-xr-x 3 1009 1009 4096 Aug 9 2007 doc**

**-rwxr-xr-x 1 1009 1009 5102 Apr 24 2007 installFixPack**

1. Now to start the installation **./db2setup** will start the set up in GUI.

**Command$ ./db2setup**



**DB2INSTALL THRU TEXT PROMPT**

**(All instructions from /storage/sapsrcCD/DB2FP9LINUX64/LINUXX86\_64/DBSW)**

<http://elibrary.fultus.com/technical/index.jsp?topic=/com.fultus.linux.howtos/howtos/DB2-HOWTO/gentoo.html>

**1.** Type in the following command and wait until the prompt returns by specifying the version you want

(Text after **DB2.ESE** is in **font 2.** Maximize it to look for the messages.)

**Command$ ./db2\_install**

Specify one or more of the following keywords,

separated by spaces, to install DB2 products.

Keyword Product Description

DB2.ADMCL DB2 Administration Client for LINUXAMD64

DB2.ESE DB2 Enterprise Server Edition for LINUXAMD64

DB2.ADCL DB2 Application Development Client for LINUXAMD64

Enter "help" to redisplay product names.

Enter "quit" to exit.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**DB2.ESE**

IBM\_db2msen81...

Preparing... ##################################################

IBM\_db2msen81 ##################################################

IBM\_db2cliv81...

Preparing... ##################################################

IBM\_db2cliv81 ##################################################

IBM\_db2icw81...

Preparing... ##################################################

IBM\_db2icw81 ##################################################

IBM\_db2ldap81...

Preparing... ##################################################

IBM\_db2ldap81 ##################################################

IBM\_db2wmsa81...

Preparing... ##################################################

IBM\_db2wmsa81 ##################################################

IBM\_db2cucs81...

Preparing... ##################################################

IBM\_db2cucs81 ##################################################

IBM\_db2conv81...

Preparing... ##################################################

IBM\_db2conv81 ##################################################

IBM\_db2secl81...

Preparing... ##################################################

IBM\_db2secl81 ##################################################

IBM\_db2sesm81...

Preparing... ##################################################

IBM\_db2sesm81 ##################################################

IBM\_db2icuc81...

Preparing... ##################################################

IBM\_db2icuc81 ##################################################

IBM\_db2icut81...

Preparing... ##################################################

IBM\_db2icut81 ##################################################

IBM\_db2jhen81...

Preparing... ##################################################

IBM\_db2jhen81 ##################################################

IBM\_db2repl81...

Preparing... ##################################################

IBM\_db2repl81 ##################################################

IBM\_db2sp81...

Preparing... ##################################################

IBM\_db2sp81 ##################################################

IBM\_db2chen81...

Preparing... ##################################################

IBM\_db2chen81 ##################################################

IBM\_db2adt81...

Preparing... ##################################################

IBM\_db2adt81 ##################################################

IBM\_db2adts81...

Preparing... ##################################################

IBM\_db2adts81 ##################################################

IBM\_db2xmls81...

Preparing... ##################################################

IBM\_db2xmls81 ##################################################

IBM\_db2cj81...

Preparing... ##################################################

IBM\_db2cj81 ##################################################

IBM\_db2dc81...

Preparing... ##################################################

IBM\_db2dc81 ##################################################

IBM\_db2dwcm81...

Preparing... ##################################################

IBM\_db2dwcm81 ##################################################

IBM\_db2fs81...

Preparing... ##################################################

IBM\_db2fs81 ##################################################

IBM\_db2inst81...

Preparing... ##################################################

IBM\_db2inst81 ##################################################

IBM\_db2wbdb81...

Preparing... ##################################################

IBM\_db2wbdb81 ##################################################

IBM\_db2xml81...

Preparing... ##################################################

IBM\_db2xml81 ##################################################

IBM\_db2jdbc81...

Preparing... ##################################################

IBM\_db2jdbc81 ##################################################

IBM\_db2ca81...

Preparing... ##################################################

IBM\_db2ca81 ##################################################

IBM\_db2rte81...

Preparing... ##################################################

IBM\_db2rte81 ##################################################

IBM\_db2conn81...

Preparing... ##################################################

IBM\_db2conn81 ##################################################

IBM\_db2crte81...

Preparing... ##################################################

IBM\_db2crte81 ##################################################

IBM\_db2das81...

Preparing... ##################################################

IBM\_db2das81 ##################################################

IBM\_db2dj81...

Preparing... ##################################################

IBM\_db2dj81 ##################################################

IBM\_db2engn81...

Preparing... ##################################################

IBM\_db2engn81 ##################################################

IBM\_db2smpl81...

Preparing... ##################################################

IBM\_db2smpl81 ##################################################

IBM\_db2djx81...

Preparing... ##################################################

IBM\_db2djx81 ##################################################

IBM\_db2inx81...

Preparing... ##################################################

IBM\_db2inx81 ##################################################

IBM\_db2pext81...

Preparing... ##################################################

IBM\_db2pext81 ##################################################

IBM\_db2essg81...

Preparing... ##################################################

IBM\_db2essg81 ##################################################

IBM\_db2cc81...

Preparing... ##################################################

IBM\_db2cc81 ##################################################

IBM\_db2icc81...

Preparing... ##################################################

IBM\_db2icc81 ##################################################

IBM\_db2icms81...

Preparing... ##################################################

IBM\_db2icms81 ##################################################

The installation logfile can be found in /tmp/db2\_install\_log.7427.

db2\_install program completed successfully.

**2.** Now create the required groups using the following commands

**Command$ groupadd -g 901 db2iadm1**

**Command$ groupadd -g 902 db2fadm1**

**Command$ groupadd -g 903 dasadm1**

**3.**  Now create the required users using the following commands.

**Command$ useradd -u 1001 -g db2iadm1 -m -d /db2/db2inst1 db2inst1**

**Command$ useradd -u 1002 -g db2fadm1 -m -d /db2/db2fadm1 db2fadm1**

**Command$ useradd -u 1003 -g dasadm1 -m -d /db2/dasadm1 dasadm1**

**4.** After creating the users in their respective groups.

Add the following changes to the lines with ‘**+**’ file **/opt/IBM/db2/V8.1/instance/db2iutil** file

--- db2iutil.orig 2005-01-17 07:05:58.000000000 -0500

+++ db2iutil 2005-01-17 07:06:32.000000000 -0500

@@ -327,7 +327,7 @@

fi

# Get output of the "df" command

**- output\_df="`df -k ${dirname2?} | tail -1`"**

**+ output\_df="`df -k ${dirname2?} | tail -n 1`"**

# On some platforms, the filesystem is on the previous line causing us

# one less token for awk to find.

free\_space\_in\_fs=`echo ${output\_df?} | awk '{if ($4 !~ /%/) {print $4} else

{print $3}}'`

@@ -382,7 +382,7 @@

rm -f ${TMPFILE3?}

# Get the name of the filesystem where dir $dirname2 resides.

**- df -k ${dirname2?} | tail +2 > ${TMPFILE3?}**

**+ df -k ${dirname2?} | tail -n +2 > ${TMPFILE3?}**

# There must be only one line in TMPFILE3 file

lcount=`wc -l ${TMPFILE3?} | awk '{print $1}'`

@@ -394,7 +394,7 @@

chk\_fsystype 22

stop\_prog 1

fi

**- fsname=`awk '{print $NF}' ${TMPFILE3?} | tail -1`**

**+ fsname=`awk '{print $NF}' ${TMPFILE3?} | tail -n 1`**

rm -f ${TMPFILE3?}

foundit=${FALSE?}

@@ -879,7 +879,7 @@

${DB2VER\_V2?})

# Dir of DB2 V2 instances

if [ -f ${DB2V2ILIST?} ]; then

**- tail +2 ${DB2V2ILIST?} > ${TMPFILE3?}**

**+ tail -n +2 ${DB2V2ILIST?} > ${TMPFILE3?}**

if [ -s ${TMPFILE3?} ]; then

for iname in `cat ${TMPFILE3?}`; do

db2ilist=${db2ilist?}" ${iname?}"

@@ -891,7 +891,7 @@

${DB2VER\_DJ?})

# Dir of DB2 V2 instances

if [ -f ${DJV211ILIST?} ]; then

**- tail +2 ${DJV211ILIST?} > ${TMPFILE3?}**

**+ tail -n +2 ${DJV211ILIST?} > ${TMPFILE3?}**

if [ -s ${TMPFILE3?} ]; then

for iname in `cat ${TMPFILE3?}`; do

db2ilist=${db2ilist?}" ${iname?}"

**5.** Create the DB2 Administration Server

**Command$ ./dascrt -u dasadm1**

**SQL4409W The DB2 Administration Server is already active.**

**DBI1070I Program dascrt completed successfully.**

1. Create the instance

**Command$ ./db2icrt –u db2inst1 db2inst1**

1. Now install the licesne

**Command$ db2licm -a db2ese.lic**

DBI1402I License added successfully.

DBI1426I This product is now licensed for use as specified in the License Acceptance and License Information documents pertaining to the licensed copy of this product. USE OF THE PRODUCT CONSTITUTES ACCEPTANCE OF THE TERMS OF THE IBM LICENSE ACCEPTANCE AND LICENSE INFORMATION DOCUMENTS, LOCATED IN THE FOLLOWING

DIRECTORY: /opt/IBM/db2/V8.1/license/en\_US.iso88591

1. **Command$ db2licm -l**

Product Name = "DB2 Enterprise Server Edition"

Product Password = "DB2ESE"

Version Information = "8.1"

Expiry Date = "Permanent"

Registered Connect User Policy = "Disabled"

Number Of Entitled Users = "5"

Enforcement Policy = "Soft Stop"

Number of processors = "4"

Number of licensed processors = "1"

Annotation = "80022f7f"

Other information = ""

1. Now start the instance and create a sample database

**Command$ db2start**

04-11-2008 20:02:27 0 0 SQL1063N DB2START processing was successful.

SQL1063N DB2START processing was successful.

1. To create a sample datebase

**Command$ db2sampl**

1. List all the DBs in the instance

**Command$ db2 list db directory**

System Database Directory

Number of entries in the directory = 1

Database 1 entry:

Database alias = SAMPLE

Database name = SAMPLE

Local database directory = /db2/db2inst1

Database release level = a.00

Comment =

Directory entry type = Indirect

Catalog database partition number = 0

1. Connect to the database

**Command$ db2 connect to sample**

Database Connection Information

Database server = DB2/LINUX 8.1.6

SQL authorization ID = DB2INST1

Local database alias = SAMPLE

**ISSUES WHILE INSTALLING**

**Command$ ls -la**

total 40

drwxr-xr-x 4 1009 1009 4096 Aug 9 2007 .

drwxr-xr-x 3 1009 1009 4096 Aug 9 2007 ..

drwxr-xr-x 6 1009 1009 4096 Aug 9 2007 db2

-rwxr-xr-x 1 1009 1009 5090 Apr 24 2007 db2\_install

-rwxr-xr-x 1 1009 1009 5078 Apr 24 2007 db2setup

drwxr-xr-x 3 1009 1009 4096 Aug 9 2007 doc

-rwxr-xr-x 1 1009 1009 5102 Apr 24 2007 installFixPack

**Command$ ./db2setup**

ERROR:

The following library files could not be loaded by db2langdir in /storage/sapsrcCD/DB2V9FP2LINUX64/DB2\_LUW\_RDBMS\_V9\_FP2SAP\_LINUX\_/LINUXX86\_64/ESE/disk1/db2/linuxamd64/install/../bin

-> libstdc++.so.5

-> libstdc++.so.5

Check the following web site for the up-to-date system requirements

of IBM DB2 Version 9.1:

http://www.ibm.com/software/data/db2/udb/sysreqs.html

Aborting the current installation ...

**Command$ ./db2\_install**  
ERROR:

The following library files could not be loaded by db2langdir in /storage/sapsrcCD/DB2V9FP2LINUX64/DB2\_LUW\_RDBMS\_V9\_FP2SAP\_LINUX\_/LINUXX86\_64/ESE/disk1/db2/linuxamd64/install/../bin

-> libstdc++.so.5

-> libstdc++.so.5

Check the following web site for the up-to-date system requirements

of IBM DB2 Version 9.1:

http://www.ibm.com/software/data/db2/udb/sysreqs.html

Aborting the current installation ...

((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((())))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))

**Command$ ./db2setup**

DBI1190I db2setup is preparing the DB2 Setup wizard which will

guide you through the program setup process. Please

wait.

/storage/sapsrcCD/DB2FP9LINUX64/LINUXX86\_64/DBSW/db2/linux/install/db2jinst: line 131: 7154 Segmentation fault $JAVA\_PATH/$JAVA\_INTERPRETER $JAVA\_OPTIONS -cp $JAVA\_CLASSPATH $DB2SetupRun "$@" 2>/tmp/db2setup.err.running

**Command$ ./db2\_install**

Specify one or more of the following keywords,

separated by spaces, to install DB2 products.

Keyword Product Description

DB2.ADMCL DB2 Administration Client for LINUXAMD64

DB2.ESE DB2 Enterprise Server Edition for LINUXAMD64

DB2.ADCL DB2 Application Development Client for LINUXAMD64

Enter "help" to redisplay product names.

Enter "quit" to exit.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**SOLUTIONS TO THE ABOVE PROBLEM**

If you encounter a problem with **-> libstdc++.so.5, -> libstdc++.so.5**  as above.

Following is the solution.

<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamrtt.doc_6.0/ITCAMfRTT_ProblemDetermination18.htm>

1. Run up2date -i compat-libstdc++ to fetch the latest version and install the following two files: /usr/lib/libstdc++.so.5 and /usr/lib/libstdc++.so.5.0.7.
2. Copy all the missing shared libraries (for example, libstdc++.so.5) to the directory where the installation program resides.
3. Change directory to the directory where the installation program is.
4. Run the following command: export LD\_LIBRARY\_PATH=.
5. Run the installation program again.

Also make the appropriate changes to user’s **.bash\_profile** by adding

**export LD\_LIBRARY\_PATH=/usr/lib**

**Looping Sample:**

. /home/db2inst1/sqllib/db2profile  
PATH1=/home/db2inst1/raghtest;  
db2 connect to sample;

while true ; do

db2 "insert into test values(200,'John','Xu')";

db2 "insert into test values(300,'Bob','Will')";

db2 "insert into test values(400,'Peter','Klau')";

done

**Redirected Restore :**

You have to do redirected restore ..Below are all the steps..Note that you got to place all these in a single script and execute them..Please understand them and if u have any doubts get back to me

db2 restore db PROD1 dbpath on /db2/PROD2/ INTO PROD2 redirect without prompting;  
db2 "set tablespace containers for 3 using (PATH "/db2/PROD2/db2inst1/NODE0000/SQL00001/TEMPSPACE2")";

db2 "set tablespace containers for 4 using (PATH "/db2/PROD2/db2inst1/NODE0000/SQL00001/USERSPACE2")";

db2 "set tablespace containers for 8 using (FILE "newpath" filesz,"newpath1" filesz,"newpath2" filesz...like this for all the containers)";

db2 "set tablespace containers for 9 using (FILE "newpath" filesz,"newpath1" filesz,"newpath2" filesz...like this for all the containers)";

db2 restore db prod1 continue;

**For deleting all the tables in a schema:**

for i in `db2 "select tabname from syscat.tables where tabschema='V7userco'`

do

db2 "drop table V7userco.$i";

done

**knowing if a select privilege exists for a user on a particular table**   Inbox

db2 "select grantee,selectauth,tabschema,tabname from syscat.tabauth where

tabname='SYSDUMMY1'"

**---dumpfile is used to capture any violations in the column definition**

for example if a column is defined as numeric and a value like 'gopi' is being inserted, that particular row will go into dumpfile.Another example would be NULL value for a column defined as not null

----Exception table is used to capture errors caused due to unique violations

Any duplicate rows with same primary key will be put in exception table

Let me know if you need any additional info

**INSERT COMMAND :**

* **INSERT** INTO T2 (intcol1,identcol2)
* **SELECT** intcol1, identcol2

**FROM** T1

**Foreign KEY CREATION:**

ALTER TABLE "EMDSCWU1"."ACTIVITY"

ADD CONSTRAINT "SQL080310234829680" FOREIGN KEY

("ORGANISATIONID")

REFERENCES "EMDSCWU1"."ORGANISATION"

("ORGANISATIONID")

ON DELETE NO ACTION

ON UPDATE NO ACTION

ENFORCED

ENABLE QUERY OPTIMIZATION;

**container monitoring script using snapshot table function**

SELECT  
substr(tablespace\_name,1,18) as TBSPC\_Name,  
substr(Container\_name,1,70) as Cont\_Name,

(CASE  
 WHEN container\_type = 0 THEN 'SMS Directory'  
 WHEN container\_type = 6 THEN 'DMS File'  
 ELSE 'DMS Device'  
 END) as Container\_Type,  
 total\_pages,  
 usable\_pages,  
 (total\_pages - usable\_pages) as free\_pages   
 FROM table (snapshot\_container (' ', -1) ) as snapshot\_container;

Monitoring commands:

Db2 list application show detail | grep –I exec | wc –l

Netstat –m

Vmstat –m

**Store Procedure Execution :**

**db2 -td# -vf sp.sql**

**TO see the Recovery File**

db2 list history backup all for empscdb1

**To get the advice of db2 to configure the parameters of the db cfg and dbm cfg**

db2 autoconfigure apply none

**For EVENT MONITORS**

**Step 1**

db2 select evmonname from syscat.eventmonitors

**step 2**

db2 "select evmonname,event\_mon\_state(evmonname) from syscat.eventmonitors"

**step 3**

C:\Documents and Settings\Administrator>

db2 "select evmonname,event\_mon\_state(evmonname) from syscat.eventmonitors"

EVMONNAME

2

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

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

DB2DETAILDEADLOCK

1

1 record(s) selected.

**Step 4**

db2 "create event monitor gopidlmon for deadlocks write to file '/home/udb4/gopi' buffersize 8 blocked"

**Step 5**

db2 set event monitor gopidlmon1 STATE=1

**step 6**

db2evmon -db dbname -evm evmonitorname

C:\Documents and Settings\Administrator>db2 "create event monitor gopidlmon for

statements write to file '/home/udb4' buffersize 8 blocked"

DB20000I The SQL command completed successfully.

DB20000I The SQL command completed successfully

**to see the output:**

db2evmon -db emsscdb4 -evm gopiStatementmon

db2evmon -db emdscdb2 -evm gopiStatementmon

**To DROP Event Monitor :**

DB2DETAILDEADLOCK

SET EVENT MONITOR DB2DETAILDEADLOCK STATE 0

**STATEMENT EVENT MONITOR;**

create event monitor gopistatementmon for statements write to table buffersize 8 blocked

set event monitor gopistatementmon STATE=1

set event monitor gopistatementmon STATE=0

select STOP\_TIME - START\_TIME AS TIME,S.\* from STMT\_gopistatementmon S

ORDER BY TIME DESC

drop event monitor gopistatementmon;

**EXPORT COMMAND :**

C:\Documents and Settings\Administrator>db2 "export to dept.del of del select \*

from "administrator.department""

**LOAD COMMAND**

method N will not work in del format

C:\Documents and Settings\Administrator>db2 "load from dept.ixf of ixf method n

(deptno,deptname,mgrno,admrdept,location,test) insert into administrator.departm

ent"

C:\Documents and Settings\Administrator>db2 "load from dept.del of del method p

(1,2,3,5,6,7) insert into administrator.departm

ent"

**C:\Documents and Settings\Administrator>db2pd -db sample -applications**

**db2 get snapshot for application agentid apllhandleno**

This will display the system name and operation performed by the user by using the paramets displayed like rows read operation perfoemed etc

**db2 get snapshot for applications on emdscdb2**

this will also show all the parametrs of all the applications

**db2 list utilities show detail**

This will diaplay the results of LOAD and what is the current situation of load

**To Kill through Agent Id if it is not easy to killa process**

db2pd -d emdscdb2 –agents

db2pd -d empscdb1 –agents

db2 list applicat

kill -9 1151092

db2stop

db2start

**To select “describe table” statements for tables in a particular schema**

db2 "SELECT 'DESCRIBE TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ';' FROM syscat.tables WHERE type = 'T' and tabschema='SCHEMANAME' ORDER BY tabschema, tabname" > DescribeTableStage.sql

**Truncating a Table:**

ALTER TABLE SCHEMANAME.CASEDATA\_IVR ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;

db2 “alter able

Ex:

db2 “Alter table schemaname.temp\_client activate not logged initially with empty table”

**Db2move command :**

**To move a table from one database to another database with data:**

db2move emdscdb1 export -sn schemaname -tn gopi(the table will be still in this database)

db2move emdscdb2 import -io replace\_create(this will recreate the table in the destination)

**Db2move databasename export -sn schemaname -tn CASEDATA\_IVR\_HUGE**

**Ex:**

db2move emdscdb1 export -sn schemaname -tn gopi

**db2move emdscdb2 import -sn schemaname -tn casedata\_ivr\_huge1**

**SNAPSHOT table Functions :**

db2 "select \* from table (snapshot\_bp(' ',-1)) as SNAPBP"  
db2 "select \* from table (SNAPSHOT\_DYN\_SQL(' ',-1)) as SNAPDYNSQL"  
db2 "select \* from table (SNAPSHOT\_TABLE(' ',-1)) as SNAPTABLE"  
db2 "select \* from table (snapshot\_database(' ',-1)) as SNAPDATA"  
db2 "select \* from table (SNAPSHOT\_LOCKWAIT(' ',-1)) as SNAPLOCKWAIT"  
db2 "select \* from table (SNAPSHOT\_LOCK (' ',-1)) as SNAPlock"

db2 "select \* from table (SNAPSHOT\_TBS (' ',-1)) as SNAPtbs"  
  
db2 "select (TOTAL\_EXEC\_TIME/NUM\_EXECUTIONS) as averageexec,STMT\_TEXT from table (SNAPSHOT\_DYN\_SQL(' ',-1)) as snapdyn where NUM\_EXECUTIONS > 0 order by averageexec desc" > snapdyn.out

**LOCK command**

db2 “call sysproc.am\_get\_lock\_chns(15,?)”

*Value of output parameters*

*--------------------------*

*Parameter Name : LOCK\_CHAINS*

*Parameter Value : >db2bp.exe (Agent ID: 15) (Auth ID: YASIR )*

*<db2bp.exe (Agent ID: 15) (Auth ID: YASIR )*

*<db2bp.exe (Agent ID: 11) (Auth ID: YASIR )*

*<db2bp.exe (Agent ID: 7) (Auth ID: YASIR )*

*Return Status = 0*

This confirms that there are more members in the lock chain:

15 --> 11 --> 7

Appl handle 15 is waiting on 11, which in turn is waiting on 7.

**Dev server 10.50.5.244 udb4 4504**

**DB2 stored proc ddl**

**To get the DDL of the desired stored procedure from the command line interface:**

db2 -x "select text from syscat.routines where routinename='INT001PROC2'"

**Event Monitor :**

db2 "select evmonname,event\_mon\_state(evmonname) from syscat.eventmonitors"

The above statement will list all the event monitors and their states in your database...Gopi could u run this on your database and send me the output??

db2 “create event monitor dlmon1 for statements with details write to file ‘/home/iudb21/gopidlmon1’ buffersize 8 blocked”

db2 “create event monitor dlmon1 for statements write to file ‘/home/iudb21/gopidlmon1’ buffersize 8 blocked”

set event monitor dlmon1 STATE=1

set event monitor gopidlmon1 STATE=1

command to view event monitor output

db2evmon -db 'databasename' -evm 'eventmonitorname'

**ALTER TABLESPACE Comamnd :**

ALTER TABLESPACE INDEX\_TS  
     RESIZE (ALL 100 M)

Example 6: Add three new containers. Extend the first container, and resize the second.

ALTER TABLESPACE TS0

ADD (FILE 'cont2' 2000, FILE 'cont3' 2000)

ADD (FILE 'cont4' 2000)

EXTEND (FILE 'cont0' 100)

RESIZE (FILE 'cont1' 3000)

**Another Method:**

Alter tablespace tbname

EXTEND (ALL 10000)

ALTER TABLESPACE <TS\_NAME>       
ADD (FILE '<path>' <number\_of\_pages>)

ALTER TABLESPACE STAGETS1

EXTEND (FILE '/container7/EMDSCDB21/STAGETS1/cont-1' 10000,

FILE '/container9/EMDSCDB21/STAGETS1/cont-2'10000,

FILE '/container10/EMDSCDB21/STAGETS1/cont-3'10000)

**Alter command for data type**

ALTER TABLE DISASTER.ADDRESS ALTER COLUMN CITY SET DATA TYPE VARCHAR ( 20 ) ;

CONNECT RESET;

**Alter command for foreign key drop**

ALTER TABLE DISASTER.DSTR\_CLIENT DROP FOREIGN KEY DSTR\_CLIENT\_FK1 ;

CONNECT RESET;

**Snapshot commands :**

db2 "select evmonname,event\_mon\_state(evmonname) from syscat.eventmonitors"

db2 -x "select \* from table (SNAPSHOT\_DYN\_SQL('TPIDB',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

db2 "select \* from table (SNAPSHOT\_DYN\_SQL('empscdb1',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

**Db2look Command for all databases :**

db2look –d emdscdb3 -i cudb50 -w jprGuZxj -e -p -o db2lookDEVD.txt

db2look –d **emsscdb5** -e -o db2lookCITC.txt

**CRT fron end address**

<http://10.50.6.115:9080/CaseReviewerTool/userHome.do?method=displaySelf>

**Alter table “:**

**Alter table tablename set columnname=’’ where errorcode=‘E01’**

db2 set current explain mode explain

**For single select statement :**

db2expln -d empscdb1  -u janartha Brn2bwld -f 1.sql -z \; -g -o test3.txt

**for the whole file :**

select \* from

db2batch -d empscdb2 –f explain.sql -a empscin1/4jqAit2d

**To create Database :**

Db2 create db databasename

Db2 drop db databasename

**Unix/Linux shell commands**

**To compress a file**

gzip filename

**To decompress a file**

Gzip –d filename.gz

**To find execution time:**

SELECT job\_run\_date,

timestampdiff(4, char(timestamp(max(sql\_start\_time))-timestamp(min(sql\_start\_time)))) Execution\_time

FROM schemaname.JOBTIMECONTROL

WHERE job\_name='CASEMGMT'

GROUP BY job\_run\_date

;

**Runstats Command :**

For RUNSTATS on all tables;

SELECT 'RUNSTATS ON TABLE  ' || rtrim(tabschema) || '.' || char(tabname,40) || ' AND DETAILED INDEXES ALL;' FROM syscat.tables WHERE type = 'T' ORDER BY tabschema, tabname;

#### Generate Runstats

#### For all schema

SELECT 'RUNSTATS ON TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ' AND DETAILED INDEXES ALL;' FROM syscat.tables WHERE type = 'T' ORDER BY tabschema, tabname;

#### For specific schema

SELECT 'RUNSTATS ON TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ' AND DETAILED INDEXES ALL;' FROM syscat.tables WHERE type = 'T' and tabschema='SCHEMANAME' ORDER BY tabschema, tabname;

For REORG on all tables in a database

SELECT 'REORG TABLE  ' || rtrim(tabschema) || '.' || char(tabname,40) || ' ;' FROM syscat.tables WHERE type = 'T' and tabschema not like 'SYS%' ;

For REORG on all tables in a database

SELECT 'REORG TABLE  ' || rtrim(tabschema) || '.' || char(tabname,40) || ' ;' FROM syscat.tables WHERE type = 'T' and tabschema='SCHEMANAME';

**Rollforward Command**

db2 rollforward db sample to 1998-04-03-14.21.56.245378 using local time and stop  
      overflow log path (/logs)

db2 "runstats on table schemaname.ag and detailed indexes all"

**Catalog commands :**

db2 "catalog tcpip node UAT remote fssdb2q01vu.fssa.state.in.us server 4580"

db2 "catalog db EMUSCDB3 as EMUSCDB3 at node UAT authentication server"

atdb02 4536

db2 catalog tcpip node PRSSL remote fssdb2p01vu server 5524 security ssl

db2 catalog db EMPSCDB1 as SLPSCDB1 at node PRSSL authentication server"

db2 catalog tcpip node SSLCP remote fssdb2p01vu server 5545 security ssl

db2 catalog db DBNAME as SLPCTDB1 at node SSLCP authentication server"

db2 catalog tcpip node SSLCR remote fssdb2p01vu server 5546 security ssl

db2 catalog db EMPCRDB1 as SLPCRDB1 at node SSLCR authentication server"

secret server

<https://passvault.in.gov/secretserver/login.aspx>

password vault

**tablespace backup or backup tablespace**

db2 "backup db empscdb2 tablespace (EMPTSTAB2) to /dev/null"

**Connection to Instance in UAT :**

Db2 attach to instancename user username using passwordname

**IN SSH :**

db2 connect to tcuram user iudb21 using U18ivTWj

**Export and Import and Load commands :**

Extract the ddl of the table using generate ddl from control center and create it in the destination database using the following command

Db2 –tvf gopitest.sgl

Db2 “export to gopitest.del of del select \* from administrator.department”

Db2 “ import from gopitest.del of del insert into administrator.department”

**For Extracting table structures from system catalog tables :**

He can get the structure using catalog tables but not the entire create table statement. He might have modified the following types of query to format the output. Please try to understand these queries and let me know .

db2 "select colname,typename,length,scale,default from syscat.columns where tabname='EMPLOYEE'"

db2 "select colname,typename,length,scale,default from syscat.columns where tabschema not like '%SYS'"

To get a stored procedure,view,trigger,function DDL use the following kind of query::

db2 -x "select text from syscat.routines where routinename='TOTAL\_RAISE'"

db2 -x "select text from syscat.views where viewname="VIEWNAME"

db2 -x "select text from syscat.triggers where routinename="TRIGGERNAME"

**To alter a table which has a drop constraint:**

Alter table tablename drop constraint on drop;

Drop table tablename;

Create table tablename;

Alter table tablename add constraint on drop;

**SIT Commands:**

**db2 "select distinct caseid from schemaname.casedata\_ivr" > caseidsfromRECONFeb11.txt**

**Runstats Command :**

**RUNSTATS ON TABLE db2user.employee**

1. Collect basic statistics on all indexes only:

RUNSTATS ON TABLE db2user.employee FOR INDEXES ALL

**Sql Querys :**

**INSERT INTO SCHEMANAMEMR1.CLIENT\_ADDRESSELEMENTDATA**

**(SELECT C.ADDRESSID, NEXTVAL FOR SCHEMANAMEMR1.ADDRESSELEMENTID\_SEQ,'TOWNSHIP', C1.RA\_TWP**

**FROM SCHEMANAMEMR1.CLIENT\_ADDRESSDATA C, SCHEMANAMEMR1.CLIENT C1**

**WHERE C1.CASEID=C.CASEID AND**

**C.CONCERNROLEID=C1.CONCERNROLEID AND NOT EXISTS (SELECT ADDRESSID FROM ID.ADDRESSELEMENT A WHERE A.ADDRESSID=C.ADDRESSID**

**AND A.ELEMENTTYPE='TOWNSHIP'));**

**ALTER TABLE SCHEMANAMEMR1.CLIENT\_ADDRESSDATA ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;**

**SELECT COUNT(\*) FROM(**

**SELECT CASEID FROM (SELECT CASEID,TYPEOFASSISTANCE,AGCATEGORY,CATEGORYSEQ,COUNT(\*) FROM SCHEMANAMEMR1.AG GROUP BY CASEID,TYPEOFASSISTANCE,AGCATEGORY,CATEGORYSEQ**

**HAVING COUNT(\*) > 1)F ) A;**

**UPDATE SCHEMANAME.AG**

**SET (FLAG,ERRORCODE)=('Y','E37')**

**WHERE CASEID IN (SELECT CASEID FROM (SELECT CASEID,TYPEOFASSISTANCE,AGCATEGORY,CATEGORYSEQ,COUNT(\*) FROM SCHEMANAMEMR1.AG GROUP BY CASEID,TYPEOFASSISTANCE,AGCATEGORY,CATEGORYSEQ**

**HAVING COUNT(\*) > 1)F);**

**INSERT INTO SCHEMANAMEMR1.AG\_EXCP**

**(SELECT CASEID,TYPEOFASSISTANCE,AGCATEGORY,CATEGORYSEQ,AGSTATUS,AGINDIVIDUALSTATUS,AGINDIVIDUALPARTICIPANTCODE,AGBEGINDATE,CURRMONTHBENEFITAMOUNT,NEXTMONTHBENEFITAMOUNT,EBTAVAILABLEDATE,AGEFFECTIVEDATE,REDETERMINATION\_DATE,AGPAYEERID,AR\_FIRSTNAME,AR\_MIDDLENAME,AR\_LASTNAME,AR\_LINE1ADDRESS,AR\_LINE2ADDRESS,AR\_CITY,AR\_STATE,AR\_ZIP,FLAG,ERRORCODE,RUNID,CURRENT TIMESTAMP,'NON-UNIQUE AG DATA' FROM SCHEMANAMEMR1.AG A WHERE FLAG ='Y');**

**DELETE FROM SCHEMANAMEMR1.AG A WHERE FLAG ='Y' ;**

**Db2advis(design advisor)**

**This db2 inbuilt utility is used to tune queries..It recommends indexes if needed.**

1)Put this query in a input file called testtune.sql

vi testtune.sql

SELECT DISTINCT deptno,location,mgrno from department where deptname = 'COMP' and location = 'ghg' order by MGRNO;

2)Execute the following command

**This one works perfect:**

**db2advis -d EMPSCDB1 -i 1.sql -n janartha –o advis.txt**

**This one worked super perfect without any error**

**db2advis -d EMPSCDB1 -i 1.sql -n SCHEMANAME -l -1 -o caseadvis.txt**

**In correct order which worked:**

**db2advis -d empscdb1 -i 1.sql -n SCHEMANAME**

It would give an output like the following:

Using user id as default schema name. Use -n option to specify schema

execution started at timestamp 2008-02-19-01.04.20.472500

found [1] SQL statements from the input file

Recommending indexes...

total disk space needed for initial set [ 0.013] MB

total disk space constrained to [ -1.000] MB

Trying variations of the solution set.

Optimization finished.

**1 indexes in current solution**

**[ 12.8764] timerons (without recommendations)**

**[ 12.8682] timerons (with current solution)**

**[0.06%] improvement**

--

--

-- **LIST OF RECOMMENDED INDEXES**

**-- ===========================**

**-- index[1], 0.013MB**

**CREATE INDEX "DB2INST1"."IDX802190604220000" ON "DB2INST1"."DEPARTMENT" ("LOCATION" ASC, "DEPTNAME" ASC, "MGRNO" ASC, "DEPTNO" ASC) ALLOW REVERSE SCANS ;**

**COMMIT WORK ;**

**RUNSTATS ON TABLE "DB2INST1"."DEPARTMENT" FOR INDEX "DB2INST1"."IDX802190604220000" ;**

**COMMIT WORK ;**

--

--

-- RECOMMENDED EXISTING INDEXES

-- ============================

--

--

-- UNUSED EXISTING INDEXES

-- ============================

-- ===========================

--

7 solutions were evaluated by the advisor

DB2 Workload Performance Advisor tool is finished.

LIST OF RECOMMENDED INDEXES suggests to add new indexes to improve the given query.You can create the index test the query for improvement and then drop the index is it is not needed.

Timerons:It is the cost of the query ..Higer the cost means longer the time it would take..So as a DBA our goal is to reduce the cost as much as possible



**Db2expln utility:**

**Db2expln can be used to analyse accesspath for a particular query**

db2expln -d empscdb1 user janartha using Brn2bwld -f 1.sql -z \; -g -o new3.txt

db2expln -d tcuram -f a1.sql -z \; -g -o new5.txt

you will have the output like this

SQL Statement:

SELECT DISTINCT deptno, location, mgrno

from department

where deptname ='COMP'and location ='ghg'

order by MGRNO

Section Code Page = 1208

Estimated Cost = 12.876366

Estimated Cardinality = 0.040000

Access Table Name = DB2INST1.DEPARTMENT ID = 2,5

| #Columns = 2

| Skip Deleted Keys

| Skip Deleted Rows

| Relation Scan

| | Prefetch: Eligible

| Lock Intents

| | Table: Intent Share

| | Row : Next Key Share

| Sargable Predicate(s)

| | #Predicates = 2

| | Insert Into Sorted Temp Table ID = t1

| | | #Columns = 2

| | | #Sort Key Columns = 2

| | | | Key 1: MGRNO (Ascending)

| | | | Key 2: DEPTNO (Ascending)

| | | Sortheap Allocation Parameters:

| | | | #Rows = 1

| | | | Row Width = 16

| | | Piped

| | | Duplicate Elimination

Sorted Temp Table Completion ID = t1

Access Temp Table ID = t1

| #Columns = 2

| Relation Scan

| | Prefetch: Eligible

| Sargable Predicate(s)

| | Return Data to Application

| | | #Columns = 3

Return Data Completion

End of section

Optimizer Plan:

RETURN

( 1)

|

TBSCAN

( 2)

|

SORT

( 3)

|

TBSCAN

( 4)

|

Table:

DB2INST1

DEPARTMENT



db2 set current explain mode explain

db2 –tvf mer.sql

db2exfmt -1 –d tcuram –o output1.txt

more output.txt

output.txt has a access plan for the query..As a DBA we need to analyse that accesspath and take decisions

**A sample DDL for a tablespace**

**Crate this in .sql file and then do**

**Db2 –tvf filename.sql**

CREATE REGULAR TABLESPACE TS IN DATABASE PARTITION GROUP NG\_ALL PAGESIZE 4096 MANAGED BY SYSTEM

USING ('/db2fs/db2inst1/cnolmdb/TS0440')

EXTENTSIZE 8

PREFETCHSIZE 24

BUFFERPOOL IBMDEFAULTBP

OVERHEAD 15.000000

TRANSFERRATE 0.100000

NO FILE SYSTEM CACHING

DROPPED TABLE RECOVERY ON;

CREATE REGULAR TABLESPACE TS PAGESIZE 4096 MANAGED BY SYSTEM

USING ('/db2fs/db2inst1/test/TS01')

EXTENTSIZE 8

PREFETCHSIZE 24

BUFFERPOOL IBMDEFAULTBP

OVERHEAD 15.000000

TRANSFERRATE 0.100000

NO FILE SYSTEM CACHING

DROPPED TABLE RECOVERY ON;

**A sample DDL for create Table :**

CREATE TABLE "QUEST "."QCOBJLIST\_STMT045" (

"NAME" VARCHAR(18) NOT NULL ,

"SEQNO" SMALLINT NOT NULL ,

"ACTION" CHAR(1) NOT NULL ,

"RESULT\_OBJECT" SMALLINT NOT NULL ,

"COPY" CHAR(1) NOT NULL ,

"SEARCH\_OBJECT" SMALLINT NOT NULL ,

"SEARCH\_FIELD1" VARCHAR(128) NOT NULL ,

"SEARCH\_FIELD2" VARCHAR(128) NOT NULL ,

"PARTLEVEL" VARCHAR(256) NOT NULL WITH DEFAULT '' ,

"RI" CHAR(1) NOT NULL ,

"AUX\_OBJECTS" CHAR(1) NOT NULL ,

"RESERVED" VARCHAR(50) NOT NULL WITH DEFAULT '' )

IN "USERSPACE1" ;

CREATE REGULAR TABLESPACE ARCHIVETASK IN DATABASE PARTITION GROUP IBMDEFAULTGROUP PAGESIZE 4096 MANAGED BY DATABASE

USING (FILE '/container1/EMPSCDB1/ARCHIVETASK/cont-1'3000000,

FILE '/container1/EMPSCDB1/ARCHIVETASK/cont-2'3000000,

FILE '/container100/EMPSCDB1/ARCHIVETASK/cont-3'3000000,

FILE '/container100/EMPSCDB1/ARCHIVETASK/cont-4'3000000)

EXTENTSIZE 8

PREFETCHSIZE 32

BUFFERPOOL IBMDEFAULTBP

OVERHEAD 12.500000

TRANSFERRATE 0.130000

DROPPED TABLE RECOVERY ON;

CREATE TABLE table1(

"FNAME" CHAR(18) NOT NULL ,

"LNAME" CHAR(10) NOT NULL '' )

IN "TS" ;

**db2 describe table syscat.tables**

To list the property structure of a database

To get all the table’s properties from syscat.tables

db2 "select tabname from syscat.tables where tabschema not like 'SYS%'"

**db2 describe table table1**

Column Type Type

name schema name Length Scale Nulls

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

FNAME SYSIBM CHARACTER 18 0 No

LNAME SYSIBM CHARACTER 10 0 No

2 record(s) selected.

**To move a Table space from one database to another :**

**db2move TEST export -ts tablespacename**

**db2move sample import**

**Example to create an Index for a Table :**

CREATE INDEX "QUEST "."TB\_KEYS" ON "QUEST "."SM\_TABLE\_HISTORY"

("MACHINE\_ID" ASC,

"INSTANCE\_ID" ASC,

"DATABASE\_ID" ASC,

"TABLESPACE\_ID" ASC,

"TABLE\_ID" ASC) PCTFREE 10 CLUSTER

ALLOW REVERSE SCANS;

CREATE INDEX “I\_KEY1" ON DB2INST1.table1"

("LNAME" ASC

) PCTFREE 10 CLUSTER ;

A table should have only one clustered Index

**Steps to move a table from one database to another database :**

**After Creating an Index we should run runstats for that table and Index** **db2**

**Db2"runstats on table schema.tablename and indexes all"**

**Command : to seek the tablespace DDL for a particular database :**

**QCTEMPTS**

db2look -d CNOLMDB -e -l | grep -i -A20 -B20 tablespace

db2look -d tcuram -e -l | grep -i -A20 -B20 tablespace > out.ddl.txt

db2look -d tcuram -i iudb21 -w U18ivTWj -e -p -o db2look1.sql

db2look -d tcuram -i iudb21 -w U18ivTWj -e -l | grep -i -A20 -B20 tablespace > out.ddl.txt

db2look -d tcuram -e -p -o db2look1.sql

Memory Size:

row Id 4 bytes

Page no 3 bytes ( 24 bits of memory)

2 to the power 24 pages can only be there in this kind of tablespace

If pagesize is 4K then tablespace is 64 GB size

**To convert a instance from 32 bit to 64 bit :**

Db2icrt –w 64

To see the space available in file system and containers :

ulimit -a

**db2inst1@szvmds024:PT:db2inst1> df –h**

Filesystem Size Used Avail Use% Mounted on

/dev/dasde1 3.4G 2.1G 1.2G 63% /

tmpfs 1002M 4.0K 1002M 1% /dev/shm

/dev/dasda1 51M 18M 31M 37% /boot

/dev/dm-1 248M 157M 79M 67% /home

/dev/dm-2 5.0G 131M 4.6G 3% /db2fs

/dev/dm-3 5.0G 2.1G 2.7G 44% /db2logfs

/dev/dasdi1 5.0G 20K 4.7G 1% /hadr/loaddir

**Top command**

To see the CPU usage by the users

**To see the CPU’s usage and to monitor the system**

**“sar –o <filename> 60 10 >/dev/null 2>&1 &” will sample the CPU activity every minute for 10 minutes and save the output in a file.  If you are running into issues, reduce the time-frame for the data collection.**

In Db2 –tvf filename.sql

Run Db2 echo date

At before and end of the select statement

After runninfg the db2advis command do runsstats on that table with index

To run all the sqls in a single file

**Db2batch -f filename.sql –d dbname**

**Db2 set current explain mode explain**

**Db2exfmt**

Following are the snapshot commands you would use for monitoring at various levels..Please store it in your document ..Also run it in your system and

1)calculate bufferpool hit ratio

2)find what kind of tablespaces do you use(SMS/DMS)

3)Find the available and used logspace on your database

4)Find top 10 sql on your database

5)Find highly read tables in your database

6)find how many bufferpools you have in your database

7)Calculate package cache overflow %

8)calculate catalog cache overflow %

9)Calculate sort overflow %

10)find the associations between tablespaces and bufferpools

11)Find the containers being used for your tablespaces

12)Find the high watermark for agents(maximum number of agents that your database has used in the recent times)

13)Check for the highwatermark on statement monitor heap

db2 get snapshot for dbm

db2 get snapshot for db on databasename

db2 get snapshot for tables on databasename

db2 get snapshot for tablespaces on databasename

db2 get snapshot for locks on emdscdb1

db2 get snapshot for bufferpools on databasename

db2 get snapshot for applications on databasename

db2 get snapshot for dynamic sql on empscdb1

**Stored Procedure Example :**

**CREATE PROCEDURE SCHEMANAMEMR1.loadAlternateNamesForCase( IN caseVar BIGINT, OUT errorCode INTEGER)**

**LANGUAGE SQL**

**P1: BEGIN**

**DECLARE SQLCODE INTEGER DEFAULT 0;**

**DECLARE SQLSTATE CHAR(5) DEFAULT ' ';**

**DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING**

**SET errorCode = SQLCODE;**

**SET errorCode = 0;**

**MERGE INTO**

**ID.ALTERNATENAME T1**

**USING (SELECT CONCERNROLEID,N1ST,NMI,NLST,NSUF FROM SCHEMANAMEMR1.CLIENT WHERE CASEID = caseVar ) T2**

**ON**

**T1.CONCERNROLEID=T2.CONCERNROLEID**

**WHEN MATCHED**

**THEN UPDATE**

**SET**

**T1.FIRSTFORENAME=T2.N1ST,**

**T1.OTHERFORENAME=T2.NMI,**

**T1.SURNAME=T2.NLST,**

**T1.NAMESUFFIX=T2.NSUF,**

**T1.FULLNAME=T2.N1ST||T2.NLST**

**WHEN NOT MATCHED**

**THEN INSERT(ALTERNATENAMEID,CONCERNROLEID,FIRSTFORENAME,OTHERFORENAME,SURNAME,NAMESUFFIX,NAMETYPE,NAMESTATUS,FULLNAME,INITIALS,VERSIONNO)**

**VALUES (NEXTVAL FOR ID.ALTERNATENAMEID\_SEQ,T2.CONCERNROLEID,T2.N1ST,T2.NMI,T2.NLST,T2.NSUF,'AT1','RST1',T2.N1ST||T2.NLST,T2.NMI,1);**

**END P1@**

**DROP PROCEDURE SCHEMANAMEMR1.loadAddressesForCase@**

**CREATE PROCEDURE SCHEMANAMEMR1.loadAddressesForCase ( IN caseVar BIGINT, OUT errorCode INTEGER)**

**LANGUAGE SQL**

**P1: BEGIN**

**DECLARE SQLCODE INTEGER DEFAULT 0;**

**DECLARE SQLSTATE CHAR(5) DEFAULT ' ';**

**DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING**

**SET errorCode = SQLCODE;**

**SET errorCode = 0;**

**------------------------------------------------------------------------**

**-- SQL Stored Procedure**

**--@AUTHOR: mahdevup**

**--@this procedure loads the ADDRESS TABLE**

**--dependencies - none**

**------------------------------------------------------------------------**

**MERGE INTO**

**ID.ADDRESS T1**

**USING**

**(SELECT ADDRESSID,ADDRESSDATA FROM SCHEMANAMEMR1.CLIENT\_ADDRESSDATA WHERE CASEID = caseVar) T2**

**ON**

**T1.ADDRESSID=T2.ADDRESSID**

**WHEN MATCHED**

**THEN UPDATE**

**SET**

**T1.ADDRESSDATA=T2.ADDRESSDATA,**

**T1.LASTWRITTEN=CURRENT\_TIMESTAMP**

**WHEN NOT MATCHED**

**THEN INSERT**

**(T1.ADDRESSID, T1.COUNTRYCODE, T1.MODIFIABLEIND, T1.ADDRESSDATA,T1.ADDRESSLAYOUTTYPE, T1.VERSIONNO, T1.LASTWRITTEN)**

**VALUES(T2.ADDRESSID,'US','1',T2.ADDRESSDATA,'US',1 ,CURRENT\_TIMESTAMP);**

**END P1@**

**DROP PROCEDURE SCHEMANAMEMR1.loadAddressElementsForCase@**

**To calculate average execution time:**

 Total executions  /   num of executions    = average execution time

**Snap shot table function :**

db2 -x "select \* from table (SNAPSHOT\_DYN\_SQL('TPIDB',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

db2 "select \* from table (SNAPSHOT\_BP('sample',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

db2 "select \* from table (SNAPSHOT\_BP('TPIDB',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

 C:\Documents and Settings\Administrator>db2 update monitor switches using buffer

pool on lock on statement on table on timestamp on

**To see the number of applications executing :**

db2 "GET SNAPSHOT FOR DATABASE  
ON empscdb1" | grep -i "Appls."

 GET SNAPSHOT FOR DATABASE

ON sample | grep -i 'Appls. executing in db manager currently'

 or

db2 “SELECT appls\_in\_db2 FROM table(snapshot\_database('empscdb1', -1)) as snapshot\_database” > appsnap.out

NOTE

 GET SNAPSHOT FOR DATABASE

ON sample | grep -i 'Appls. executing in db manager currently'

 or

SELECT appls\_in\_db2

FROM table(snapshot\_database('sample', -1)) as snapshot\_database

**To see Locks**

**U ll see lock wait**

 Db2 get snapshot for application agent id apllicationhandle  
db2 "get snapshot for application agentid 169"

db2 "get snapshot for application agentid 386"

It will show Waiting for lock

**It will show like this :**

Locks held by application = 386

Lock waits since connect = 5

And will display the lock handle of the other lock

Get the application snapshot for the other lock

 db2 -x "select \* from table (SNAPSHOT\_BP('TPIDB',-1)) AS TABLESNAP order by TOTAL\_EXEC\_TIME desc fetch first 5 rows only" | more

**Db2 get monitor switches**

**Db2 get dbm monitor switches**

**Db2 update monitor switches using LOCK on/off**

**Db2 update dbm cfg using monitorswitchname on/off**

**Db2 reset monitor all**

**What kind of monitor switches exist??**

**What is the difference between dbm level switches and session level??**

**How do you reset counters in a snapshot??**

**Which switch is on by default and what is that used for??**

**LOCKS section :**

When you type db2 “list applications all”

You ll see like this on locks section:

Lock timeout (seconds) = 90

Locks held by application = 1338

Lock waits since connect = 6

Time application waited on locks (ms) = Not Collected

Deadlocks detected = Not Collected

Lock escalations = 30

Exclusive lock escalations = 0

Number of Lock Timeouts since connected = 5

Total time UOW waited on locks (ms) = Not Collected

**ISOLATION LEVELS**

SELECT deptno, location

FROM department

WHERE location LIKE 'B%'

The example in Figure 5.24 demonstrates the locks that will he held on the different rows in the department table when the cursor is first opened. Figure 5.25 demonstrates which locks will be held during the second fetch.

**Figure 5.25. Lock types on each row at first fetch of row (B01, B0).**

Deptno Location UR/NC CS RS RR

A00 A S

B01 B0 S S S

C01 B1 S S

D01 D S

E01 E S

**Figure 5.26. Lock types on each row at second fetch of row (C01, B1).**

Deptno Location UR/NC CS RS RR

A00 A S

B01 B0 S S

C01 B1 S S S

D01 D S

E01 E S

As you can see in the example, the locking behavior on each row is the same with RR, RS, and UR/NC during the first and second fetch. The row that is locked does vary with CS because when a row is fetched, the lock on the previous row is released. You should use the isolation level that gives you the data integrity you need in your queries but also causes the minimal amount of contention. Cursor Stability and Read Stability are most often used to achieve these goals

**HADR SETUP :**

To set up HADR for the command line environment, complete the following steps:

1. Set the required database configuration parameters.

If archive logging is not already turned on, then update the LOGRETAIN

parameter using the following command:

**db2 update database configuration for**

*sample* **using LOGRETAIN**

recovery

Also, set the LOGINDEXBUILD parameter, so that index creation, re-creation,

or reorganization operations are logged, using the following command:

**db2 update database configuration for**

*sample* **using LOGINDEXBUILD ON**

2. Back up your database on the primary and move a copy to the standby server

using the following command:

**db2 backup database**

*sample* **to** */usr/db2/backup*

3. Restore the database on your standby system.

In our example, we restore to the SAMPLE database on the POLONIUM

server using the following command:

**db2 restore database**

*sample* **from /usr/db2/backup taken at**

*20061011141321*

**replace history file**

4. Configure databases for ACR. This step is optional but highly recommended.

To configure ACR, update the ALTERNATE SERVER database configuration

parameter on both the primary and standby servers with the following steps:

a. On the primary server, LEAD in our test case, set the standby server as

the alternate server using the following command:

**db2 update alternate server for database sample using hostname**

*polonium*

**port** *50001*

b. On the standby server, POLONIUM in our test case, set the primary server

as the alternate server using the following command:

**db2 update alternate server for database sample using hostname**

*lead*

**port** *50001*

5. Update the following fields in the services file on the primary system LEAD for

HADR communication:

– Service name: DB2\_HADR\_1

– Port number: 55001

– Service name: DB2\_HADR\_2

– Port number: 55002

Chapter 3. HADR setup

**67**

6. Update the following fields in the services file on the standby system

POLONIUM for HADR communication:

– Service name: DB2\_HADR\_1

– Port number: 55001

– Service name: DB2\_HADR\_2

– Port number: 55002

7. Update HADR database configuration parameters on the primary database

with the following commands. In our test case, it is LEAD.

**db2 update db cfg for**

*emdscdb2* **using** *HADR\_LOCAL\_HOST LEAD*

**db2 update db cfg for**

*sample* **using** *HADR\_LOCAL\_SVC DB2\_HADR\_1*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_HOST POLONIUM*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_SVC DB2\_HADR\_2*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_INST DB2INST1*

**db2 update db cfg for**

*sample* **using** *HADR\_SYNCMODE SYNC*

**db2 update db cfg for**

*sample* **using** *HADR\_TIMEOUT 3*

**db2 connect to**

*sample*

**db2 quiesce database immediate force connections**

**db2 unquiesce database**

**db2 connect reset**

8. Update HADR database configuration parameters on the standby database,

POLONIUM in our test case, using the following commands:

**db2 update db cfg for**

*sample* **using** *HADR\_LOCAL\_HOST POLONIUM*

**db2 update db cfg for**

*sample* **using** *HADR\_LOCAL\_SVC DB2\_HADR\_2*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_HOST LEAD*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_SVC DB2\_HADR\_1*

**db2 update db cfg for**

*sample* **using** *HADR\_REMOTE\_INST db2inst1*

**db2 update db cfg for**

*sample* **using** *HADR\_SYNCMODE SYNC*

**db2 update db cfg for**

*sample* **using** *HADR\_TIMEOUT 3*

9. Start HADR on the standby database on POLONIUM in our test case, using

the following commands:

**db2 deactivate database**

*sample*

**db2 start hadr on database**

*sample* **as standby**

10.Start HADR on the primary database on LEAD in our test case, using the

following commands:

**db2 deactivate database**

*sample*

**db2 start hadr on database**

*sample* **as primary**

Pleaqse clieck link below for more details on HADR and there is a sample PDF document also on this.

<http://www.ibm.com/developerworks/data/bestpractices/hadr/>

db2 “load client from "/home/iudb21/interfaces/data/abc.del" of del modified by coldel| dateformat="YYYYMMDD" timeformat="HHMM" USEDEFAULTS savecount 1

500 messages "/home/iudb21/db2load/mesgint001.db2load" replace into schemaname.casedata\_ivr(CASEID,RID,N1ST,NMI,NLST,NSUF,SSN,DOB,P\_ACODE,P\_NUMBER,REDET\_DATE,CASESTATUS,RA\_STNBR,

RA\_UNIT,RA\_DIR,RA\_STRR,RA\_SFX,RA\_QUAD,RA\_APT,RA\_LINE2,RA\_CITY,RA\_STATE,RA\_ZIP,RA\_CNTY,RA\_TWP,TANF\_PENDVR\_SSN,TANF\_PENDVR\_NONFINANCIAL,TANF\_PENDVR\_FINANCIAL,TANF\_PENDVR\_RESOURCE,A

PPOINTMENTDATE,APPOINTMENTSTARTTIME,ACTIVITYCODE,TANF\_CAT,TANF\_SEQ,TANF\_AGSTATUS,TANF\_AGENDDATE,TANF\_AR\_FNAME,TANF\_AR\_MNAME,TANF\_AR\_LNAME,TANF\_AR\_LINE1,TANF\_AR\_LINE2,TANF\_AR\_CITY

,TANF\_AR\_STATE,TANF\_AR\_ZIP,TANF\_AGINDSTATUS,TANF\_AGINDPARTCODE,MA1\_CAT,MA1\_SEQ,MA1\_AGSTATUS,MA1\_AGENDDATE,MA1\_AR\_FNAME,MA1\_AR\_MNAME,MA1\_AR\_LNAME,MA1\_AR\_LINE1,MA1\_AR\_LINE2,MA1\_AR\_

CITY,MA1\_AR\_STATE,MA1\_AR\_ZIP,MA1\_AGINDSTATUS,MA1\_AGINDPARTCODE,MA2\_CAT,MA2\_SEQ,MA2\_AGSTATUS,MA2\_AGENDDATE,MA2\_AR\_FNAME,MA2\_AR\_MNAME,MA2\_AR\_LNAME,MA2\_AR\_LINE1,MA2\_AR\_LINE2,MA2\_AR\_

CITY,MA2\_AR\_STATE,MA2\_AR\_ZIP,MA2\_AGINDSTATUS,MA2\_AGINDPARTCODE,FS\_CAT,FS\_SEQ,FS\_AGSTATUS,FS\_AGENDDATE,FS\_AR\_FNAME,FS\_AR\_MNAME,FS\_AR\_LNAME,FS\_AR\_LINE1,FS\_AR\_LINE2,FS\_AR\_CITY,FS\_AR

\_STATE,FS\_AR\_ZIP,FS\_AGINDSTATUS,FS\_AGINDPARTCODE,GENDER,LINKSWITCH,RIDALIAS,CONFFLAG,FS\_IMPACTCODE,TANF\_IMPACTCODE,FS\_CERTIFICATION\_BEGINDATE,FS\_CURRENTMONTHBENEFITAMOUNT,FS\_NEXT

MONTHBENEFITAMOUNT,FS\_EBTAVAILABLEDATE,TANF\_CERTIFICATION\_BEGINDATE,TANF\_CURRENTMONTHBENEFITAMOUNT,TANF\_NEXTMONTHBENEFITAMOUNT,TANF\_EBTAVAILABLEDATE,MA1\_AGEFFECTIVEDATE,MA2\_AGEFF

ECTIVEDATE,MA1\_PENDVR\_SSN,MA1\_PENDVR\_NONFINANCIAL,MA1\_PENDVR\_FINANCIAL,MA1\_PENDVR\_RESOURCE,MA2\_PENDVR\_SSN,MA2\_PENDVR\_NONFINANCIAL,MA2\_PENDVR\_FINANCIAL,MA2\_PENDVR\_RESOURCE,FS\_PEND

VR\_SSN,FS\_PENDVR\_NONFINANCIAL,FS\_PENDVR\_FINANCIAL,FS\_PENDVR\_RESOURCE,CURAM\_APPL\_NO,FS\_AGPAYEERID,TANF\_AGPAYEERID,MA1\_AGPAYEERID,MA2\_AGPAYEERID,APPOINTMENTINTERVIEWMETHOD,APPOINTM

ENTENDTIME,APPOINTMENTOFFICELOCATION) FOR EXCEPTION SCHEMANAME.CASEDATAIVR\_EXCP NONRECOVERABLE indexing mode rebuild”

REFM\_INT001\_Daily.del

MAIL sent to:

Luann Mengwasser/Jefferson City/IBM, Anthony Chen/Houston/IBM, Allan Moso/Fort Wayne/Contr/IBM, Bharathi Nagarajan/Indianapolis/Contr/IBM, Pradeep Muhdeva/Indianapolis/Contr/IBM, Diwaker Ubnare/Indianapolis/Contr/IBM, Sankar K Singha/Fort Lauderdale/IBM, dadapa@rcrtechnology.com, tcox@rcrtechnology.com, dforsythe@rcrtechnology.com, mpaynter@rcrtechnology.com, Matthew Rager/Fort Wayne/Contr/IBM, kavitha@rcrtechnology.com, sbuddhe@rcrtechnology.com

"TIMESTAMP" TIMESTAMP ,

"MESG" CLOB(32768) LOGGED NOT COMPACT )

Curamdata

CREATE TABLE "EMCASEREVIEW"."CURAMDATA\_EXCP" (

"CASENUMBER" BIGINT NOT NULL ,

"CASENAMERID" BIGINT ,

"CASENAME" CHAR(56) ,

"DESCRIPTION" CHAR(150) ,

"TASKID" BIGINT NOT NULL ,

"TASKCREATIONDATE" TIMESTAMP ,

"TASKCLOSEDDATE" TIMESTAMP NOT NULL ,

"TYPEOFASSISTANCE" CHAR(10) ,

"CAT" CHAR(10) NOT NULL ,

"SEQ" CHAR(2) NOT NULL ,

"AGSTATUS" CHAR(10) ,

"CASEACTIONTYPECODE" CHAR(4) ,

"MAXDATEFLAG" CHAR(1) ,

"TIMESTAMP" TIMESTAMP ,

"MESG" CLOB(32768) LOGGED NOT COMPACT )

IN "CRTDATA32KTS1" ;

CREATE TABLE "EMCASEREVIEW"."ICESDATA\_EXCP" (

"CASENUMBER" BIGINT NOT NULL ,

"CAT" CHAR(4) NOT NULL ,

"SEQ" CHAR(2) NOT NULL ,

"SIGNEDDATE" DATE ,

"TRANSACTIONDATE" DATE ,

"AGSTATUS" CHAR(1) ,

"LIVINGARRANGEMENTCODE" CHAR(2) ,

"RSN\_STATUS1" SMALLINT ,

"RSN\_STATUS2" SMALLINT ,

"RSN\_STATUS3" SMALLINT ,

"RSN\_STATUS4" SMALLINT ,

"RSN\_STATUS5" SMALLINT ,

"COUNTYTOWNSHIPCODE" CHAR(2) ,

"ADR\_ZIP" CHAR(5) ,

"SW\_EXPEDITED\_FS" CHAR(1) ,

"CLAIM\_IND" CHAR(1) ,

"APPEAL\_HEARING\_IND" CHAR(1) ,

"TIMESTAMP" TIMESTAMP ,

"MESG" CLOB(32768) LOGGED NOT COMPACT )

IN "CRTDATA32KTS1" ;

# DBA Activity

## DBA Checklists

### Daily Procedures

* Verify that all instances are up and running
* Verify that all databases are active or consistent
* Look for new Administration Notification Log and/or DB2DIAG.LOG entries
* Check that the previous night's backups were successful on all servers
* Verify database logs have been archived successfully
* Check that no DBM and/or DB configuration parameters have changed
* Check important performance measures based on your workload
* Check for automatic actions DB2 has taken on your behalf
* Ensure there is enough free memory on both database and server

### Weekly Procedures

* Look for new objects
* Look for new or changed applications
* Look for tables and/or indexes needing REORG
* Look for tables and indexes needing RUNSTATS
* Look for the 10 most active tables
* Archive all alert logs and DB2DIAG.LOG files
* Check for software updates

### Monthly Procedures

* Look for indicators of exceptional growth
* Project future performance based on projected growth

## DBA Procedure

### DB2 Installation & Configuration

* + Install DB2 first time – Steps and prerequisites
  + Upgrade DB2 version
  + Apply fixes/patches

## Instance Management

### Create Instance

You can have DB2 UDB create the DB2 instance automatically while installing DB2 UDB. You also can create the DB2 instance manually after the installation is completed. On Linux or AIX, the DB2 instance can be created by executing **db2setup** program used to install DB2 manually through the command line by

Issuing the **db2icrt** command, or by using the Control Center provided by DB2.

* **Create Instance using db2setup/db2isetup utilities**

Using the ./db2setup utility provides an easy way to create a DB2 instance. As root perform the following:

1. Launch the **db2setup** utility.

2. Check the **Create a new DB2 instance or set up an existing DB2 instance** option.

3. This screen will allow you to configure the DB2 administration server and user used as a repository for the GUI administration tools provided with DB2 such as the Control Center. The default value for this user is *dasusr1* with a default home directory of */home/dasusr1*

4. Click on the **Instance setup** option and choose the **Create DB2 instance - 32 bit** option.

5. For a single partition instance choose the first option.

6. On the Set User Information for the DB2 Instance Owner screen, you need to identify a system user who will be the instance owner. If you choose a new user, then specify the name of the user and his password. The default values are user *db2inst1* and group *db2grp1.* You also have to specify the home directory for this user i.e. /home/db2inst1. By default any databases created under this instance will be created in this directory unless otherwise specified.

Both the user and the home directory will be created by the installer.

7. The Set User Information for the Fenced User screen allows you to specify its username and password. The default user is db2fenc1 assigned to group db2fgrp1 in home directory /home/db2fenc1.

8. The tools Catalog screen is meant for preparing the DB2 Tools catalog on the server. Choose the **Do not prepare the DB2 tools catalog on this computer** if you do not need the tools catalog installed.

9. Finally, set the administrator contact information and click **Finish**. As part of the instance creation, the installer will create all three users identified mainly as db2inst1, db2fenc1, and dasadm1. If you do not want to use the default user IDs, you can create the user IDs and groups ahead of time and use the IDs during creating the instance. The installer will also add the following entry to the /etc/services file in order to allow communication from DB2 clients:

db2c\_db2inst1 50000

Where *db2c\_db2inst1* indicates the service name and *50000* indicates the port number. Subsequent instances may be created on the same server simply by invoking the ***/opt/IBM/db2/V8.1/instance/db2isetup***utility and going through the above steps.

* **Create Instance and Database using DB2 command**

The DB2 instance can also be created manually by following the following steps.

1. Log on to the Linux system as root.

2. Create the necessary groups for DB2 Instance owner, administration server, and Fenced ID using the following commands:

**groupadd db2grp1**

**groupadd db2fenc1**

**groupadd dasadm1**

3. Create the DB2 Instance user ID, administration server user ID, and Fenced ID and assign them to their respective groups using the following commands

**useradd -g** db2grp1 **-d** /home/db2inst1 db2inst1 **-p** *my\_password*

**useradd -g** db2fenc1 **-d** /home/db2fenc1 db2fenc1 **-p** *my\_password*

**useradd -g** dasadm1 **-d** /home/dasusr1 dasusr1 **-p** *my\_password*

4. Issue the command:

**db2icrt -u db2fenc1 db2inst1**

5. Edit the /etc/services file and add the following entries:

db2c\_db2inst1 50000/tcp #DB2 port for remote clients

db2idb2inst1 50001/tcp #interrupt ports for DB2 1.x clients

6. Log on as the instance owner and update the Database Manager Configuration (dbm cfg) file to reflect the service name in the /etc/services file

**update dbm cfg using SVCENAME db2c\_db2inst1**

7. Set up the default communication protocol:

**db2set -i db2inst1 -i DB2COMM=TCPIP**

8. Set the instance to auto-start with the system if desired:

**db2set -i** db2inst1 **DB2AUTOSTART=TRUE**

At this point the server is ready to create the database. To simply database connectivity test, you can create a sample database in four easy steps.

### Drop Instance

### Update/Configure Instance

## Database Management

Create Database Drop Database, Configure Database

### Create Database

A DB2 database can either be created by the Control Center or by using the command line. In order to create a DB2 database manually you may follow the following steps:

1. Log on to the Linux/unix system as the instance owner db2inst1.

2. Since DB2 allows for one instance to have multiple databases, it is always recommended to attach to the desired instance before the create database command is issued:

**db2 attach to** *instance\_name*

where the instance name in our case is db2inst1.

3. Issue the create database command. The simplest create database command can take the form:

db2 “create database *my\_database* on */db\_path*”

Example:

CREATE DATABASE EMDSCDB3 ON '/container2/EMDSCDB3' USING CODESET UTF-8 TERRITORY US COLLATE USING SYSTEM PAGESIZE 32 K CATALOG TABLESPACE MANAGED BY SYSTEM USING ('/container3/EMDSCDB3/catalogts') EXTENTSIZE 16 PREFETCHSIZE 32 USER TABLESPACE MANAGED BY SYSTEM USING ( '/container4/EMDSCDB3/usersts/cont-1','/container6/EMDSCDB3/usersts/cont-2', '/container6/EMDSCDB3/usersts/cont-3') EXTENTSIZE 16 PREFETCHSIZE 48 TEMPORARY TABLESPACE MANAGED BY SYSTEM USING ( '/container5/EMDSCDB3/tempts/cont-1','/container6/EMDSCDB3/tempts/cont-2' ) EXTENTSIZE 16 PREFETCHSIZE 32 WITH 'DEV-CURAM ORGUSER Database';

This command will create a database and the following three table spaces:

– **SYSCATSPACE** to store system catalog tables

– **USERSPACE1** to store user defined objects

– **TEMPSPACE1** to store temporary objects

These table spaces can be viewed by issuing the command:

**db2 list tablespaces**

### Create database Using REDIRECTED RESTORE

* Take backup of curam database

db2 backup db curam to /home/db2inst1/db2dbbkp

* List history

db2 list history backup all for database empscdb1

Here is the timestamp of database backup

20070217011652

* Find out containers name from source database

db2 list tablespaces

Above command will tell you container details for curam database tablespaces.

* List containers for each tablespace

db2 list tablespace containers for 0

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0000.0

db2 list tablespace containers for 1

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0001.0

db2 list tablespace containers for 2

/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0002.0

db2 list tablespace containers for 3

/home/db2inst1/curam/a

db2 list tablespace containers for 4

/home/db2inst1/curam/b

db2 list tablespace containers for 5

/home/db2inst1/db2inst1/NODE0000/SQL00001/SYSTOOLSPACE

db2 list tablespace containers for 6

/home/db2inst1/curam/c0

db2 list tablespace containers for 7

/home/db2inst1/curam/d

db2 list tablespace containers for 8

/home/db2inst1/curam/e

make directory

* Restore database to new database [curamtst]

RESTORE database curam user db2inst1 using db2inst1 taken at 20070216111246 to /home/db2inst1/curamtest into curamtest redirect;

OR

RESTORE database curam user db2inst1 using db2inst1 from /home/db2inst1/db2dbbkp to /home/db2inst1/curamtest into curamtest redirect;

* Force application

db2 force application all;

* Set tablespace containers to new database

SET TABLESPACE CONTAINERS FOR 0 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0000.0");

SET TABLESPACE CONTAINERS FOR 1 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0001.0");

SET TABLESPACE CONTAINERS FOR 2 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SQLT0002.0");

SET TABLESPACE CONTAINERS FOR 3 USING (PATH "/home/db2inst1/curamtest/curam/a");

SET TABLESPACE CONTAINERS FOR 4 USING (PATH "/home/db2inst1/curamtest/curam/b");

SET TABLESPACE CONTAINERS FOR 5 USING (PATH "/home/db2inst1/curamtest/db2inst1/NODE0000/SQL00001/SYSTOOLSPACE");

SET TABLESPACE CONTAINERS FOR 6 USING (PATH "/home/db2inst1/curamtest/curam/c0");

SET TABLESPACE CONTAINERS FOR 7 USING (PATH "/home/db2inst1/curamtest/curam/d");

SET TABLESPACE CONTAINERS FOR 8 USING (PATH "/home/db2inst1/curamtest/curam/e");

* Continue database restore

RESTORE DATABASE curam CONTINUE;

* The database [curamtst] will be in rollforward pending status since LOGRETAIN=RECOVERY, USEREXIT=ON. The rollforward pending status must be removed on target database before accessing it.

db2 rollforward db curamtst user db2inst1 using db2inst1 complete

* Connect to target database

db2 connect to curamtst

--

RESTORE database curamtst user db2inst1 using db2inst1 from /home/db2inst1/db2dbbkp taken at 20070315215750 to /home/db2inst2/curamtst into curamtst redirect;

SET TABLESPACE CONTAINERS FOR 0 USING (PATH "/home/db2inst2/curamtst/db2inst2/NODE0000/SQL00001/SQLT0000.0");

SET TABLESPACE CONTAINERS FOR 1 USING (PATH "/home/db2inst2/curamtst/db2inst2/NODE0000/SQL00001/SQLT0001.0");

SET TABLESPACE CONTAINERS FOR 2 USING (PATH "/home/db2inst2/curamtst/db2inst2/NODE0000/SQL00001/SQLT0002.0");

SET TABLESPACE CONTAINERS FOR 3 USING (PATH "/home/db2inst2/curamtst/curamtst/curamtst\_l");

SET TABLESPACE CONTAINERS FOR 4 USING (PATH "/home/db2inst2/curamtst/curamtst/curamtst\_t");

SET TABLESPACE CONTAINERS FOR 5 USING (PATH "/home/db2inst2/curamtst/db2inst2/NODE0000/SQL00001/SYSTOOLSPACE");

SET TABLESPACE CONTAINERS FOR 6 USING (PATH "/home/db2inst2/curamtst/curamtst/elabts");

SET TABLESPACE CONTAINERS FOR 7 USING (PATH "/home/db2inst2/curamtst/curamtst/indx01");

SET TABLESPACE CONTAINERS FOR 8 USING (PATH "/home/db2inst2/curamtst/curamtst/ts1");

SET TABLESPACE CONTAINERS FOR 9 USING (PATH "/home/db2inst2/curamtst/captiva/ts2");

SET TABLESPACE CONTAINERS FOR 10 USING (PATH "/home/db2inst2/curamtst/captiva/indx02");

SET TABLESPACE CONTAINERS FOR 11 USING (PATH "/home/db2inst2/curamtst/db2inst2/NODE0000/SQL00001/SYSTOOLSTMPSPACE");

RESTORE DATABASE curamtst CONTINUE;

rollforward db curamtst user db2inst2 using db2inst2 complete;

connect to curamtst user db2inst2 using db2inst2;

### Copy everything to new database using db2move

* Create database using following

db2 "create database testdb on /home/db2inst1/testdb autoconfigure apply none"

* Use db2move to export source database tables and data

db2move curamtst export

* Import all tables data using db2move import

db2move testdb import -io REPLACE\_CREATE

### Renaming database

The database name can be changed using db2relocatedb utility. If you would like to change the database name test123 from instance udb9 that resides on path /home/udb9

* Create the following configuration file:

DB\_NAME=test123,testabc

DB\_PATH=/home/udb9

INSTANCE=udb9

NODENUM=0

* and Save the configuration file as relocate.cfg
* Use the following command to make the changes to the database files:

db2relocatedb -f relocate.cfg

### Dropping Database

The databases can be dropped using drop database <database name> command.

### Adding/Changing Database Comments

The databases comments can be changed using following command:

db2 change database TCAPTIVA comment with \"SIT-CAPTIVA Database\"

## Creating DB2 Database Users

In DB2 UDB, users are created at the operating system level using operating system commands and utilities.

Steps for creating a new database user called db2usr and grant him select, insert, and update privileges on table accounts on an AIX environment, perform the following steps:

1. Log on to the AIX server as root and create a group:

**mkgroup** id=995 accttab

2. Create a user and assign him to group accttab:

**mkuser id=**1001 **pgrp=**accttab **groups=**accttab **home=**/home/db2user db2user

3. Edit the .profile file for user db2usr and add the db2profile path to it, and execute the .profile in order to reflect the changes:

. /db2/home/db2inst1/sqllib/db2profile

. ./.profile

4. Log on to the AIX server as the instance owner or any authorized user and connect to the database:

**su -** db2inst1

**db2 connect to** sample

5. Grant the desired privileges to the group:

**db2 “grant select, insert, update on accounts to group accttab”**

6. Log on as user db2user, connect to database sample, and issue a SQL statement against table accounts:

**su -** db2user

**db2 connect to sample**

**db2 “select \* from** db2inst1.staff”

## Table space Management

## Table Management

## Backup and Recovery

### Database Backup and Recovery

* **With Rolling Forward**

db2 "RESTORE database tcuram user udb21 using JZCvxQdU use TSM taken at 20070826120219 to /container1/TCURAM into tcuram"

db2 list tablespaces show detail

db2 rollforward database tcuram to end of logs and complete

* **With out rollforward**

db2 "RESTORE database tcuram user udb21 using JZCvxQdU use TSM taken at 20070826120219 to /container1/TCURAM into tcuram WITHOUT ROLLING FORWARD"

db2 list tablespaces show detail

### Tablespace Backup and Recovery

1. **Take tablespace backup**

db2 "backup database emdscdb1 tablespace (STAGETS3) online to /container1/DBBACKUPS"

1. List history

db2 "list history backup all for emdscdb1"

1. Restore tablespace from backup

db2 "restore database emdscdb1 tablespace (STAGETS3) from /container1/DBBACKUPS/temp taken at 20080127043944"

db2 "rollforward database emdscdb1 to end of logs and stop"

1. List tablespaces and verify the status

db2 list tablespaces show detail

### Dropped Table Recovery

DB2 allows you to recover a database's data using a database restore operation. In fact, you could use the restore operation in tandem with the rollforward operation to restore the database to a point before the table was dropped, but restoring the entire database can be very time consuming. More importantly, during this database-level operation, you can't access your database. On the flipside, with the table recovery feature, your table-level restore effort is more directed, allowing you to use the database in question.

* **Enable Dropped Table Recovery**

Inside DB2, the database is divided into table spaces. Database tables are assigned to these table spaces. In order for a dropped table to be recoverable, you have to turn on the DROPPED TABLE RECOVERY option on the table space that houses it. This option is enabled by default for newly created data table spaces if you used the command line to create your table space.

To make sure a given name space has table recovery enabled, query the DROP\_RECOVERY column of the SYSCAT.TABLESPACES catalog table:

db2 => select TBSPACE, DROP\_RECOVERY from SYSCAT.TABLESPACES

To enable the DROPPED TABLE RECOVERY option after you've created a table space, use the following syntax:

ALTER TABLESPACE <tablespace-name> DROPPED TABLE RECOVERY ON

* **Backup**

Before you can recover a database table, you must have backed up its housing database.

a) DB2 uses database logs to recover the changes that occurred to the database after a backup. For recovery to take place, you need to enable a logging type called archival logging. Enter the following command:

db2 update db cfg using LOGRETAIN recovery immediate

1. Disconnect all applications from the database for the changes to take effect.
2. Take backup of the database: **db2 backup db emdscdb1**

*Backup successful. The timestamp for this backup image is:   
20040320123838*

Be sure to keep the timestamp of the backup operation in a safe place.

1. Table to be recovered (just for reference)

Assume that you already have a table. The DDL for table is as:

create table schemaname.RECOVERME(NAME varchar(55) not null primary key);

Sample of record inserted into this table:

insert into schemaname.RECOVERME values('John Thomas');

insert into schemaname.RECOVERME values('Diwakar Ubnare');

* **Recovery Steps**

You may or may not know the table name that is dropped. Assume that you do not know the name of the table.

1. Find out what table is accidentally dropped with the following command:

**db2 "list history dropped table all for emdscdb1"**

This gives information table that is dropped, tablespace name, backupID and also DDL (Data Definition Language) necessary to reconstruct the table you dropped.

1. Restore database from the backup image using the RESTORE backup command. This can be either a database-level restore or a tablespace-level restore. When deciding which one to choose, understand that tablespace-level restore requires exclusive access only to the tablespace and not the entire database during the recovery. The following syntax restores your database tablespace (USERSPACE1):

**restore database emdscdb1 tablespace (STAGETS2) from "/container1/DBBACKUPS" taken at 20080127025839 without rolling forward without prompting**

1. At this point in time, if you tried to query for data in the dropped table, DB2 will inform you that tables do not exist. You need to rollforward to the end of your logs to get back to the point of your last data transaction.

**rollforward database emdscdb1 to end of logs and complete recover dropped table 00000000006b67e200080004 to "/container1/DBBACKUPS"**

If the rollforward operation completes successfully, you will get confirmation.

1. Created the table from DDL – in list history

**CREATE TABLE SCHEMANAME.RECOVERME( "NAME" VARCHAR(55) NOT NULL ) IN "USERSPACE1" ;**

1. In your ROLLFORWARD operation, you specified the predicate of:

and complete recover dropped table 00000000006b67e200080004 to "/container1/DBBACKUPS"

It created subdirectories under export directory (i.e., /container1/DBBACKUPS). The subdirectories are named NODE*nnnn* (where *nnnn* represents the database partition or node number), and the data files created house your dropped table data in a subdirectory called data

1. **IMPORT the tables contents**

The table already reconstructed and now you can import data from backup directory.

**IMPORT FROM "/container1/DBBACKUPS/NODE0000/data" OF DEL METHOD P (1) MESSAGES "/container1/DBBACKUPS/importlog.msg" INSERT INTO SCHEMANAME.RECOVERME (NAME)**

## Data Movement

### Export

### Import

### Load

## Comparison

### Truncate and Delete

### Difference between SMS and DMS tablespaces

### Difference between IMPORT and Truncate with ALTER

|  |  |  |
| --- | --- | --- |
| **Difference Between ALTER and IMPORT to Truncate/Empty Table** | | |
| **Description** | **Truncate** | **Import with NULL** |
| Syntax/command | db2 import from /dev/null of del replace into | alter table <table\_name> activate not logged initially with empty table |
| Logging | Import logs the activity. The IMPORT can be handled by roll-forward recovery | It does not log the activity. It can not handle roll-forward recovery |
| Privileges | IMPORT requires one of the following: SYSADM DBADM CONTROL on the table or view INSERT, SELECT, and DELETE on the table or view | ALTER needs at least one of the following: *ALTER on the table to be altered CONTROL on the table to be altered ALTERIN on the schema of the table SYSADM DBADM* |
| Performance | No performance difference | No performance difference |

UNIX

ALTER TABLE SCHEMANAME.CASE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;

db2 "IMPORT FROM '/dev/null' of del replace into schemaname.case"

db2 "LOAD CLIENT FROM '/dev/null' of del replace into schemaname.case"

Windows

db2 "IMPORT FROM NULL of del replace into schemaname.case"

## Authentication and Privileges

## Monitoring

### Operating System Monitoring

**CPU, Memory and Paging Space Utilization**

Capture the CPU, memory and other operating system usage information on UNIX or Linux use iostat, vmstat commands.

vmstat 300 28800 > vmstat.out

iostat -tx 300 28800 > iostat.out

Note: Above command will capture a snapshot every 5 minutes (300 seconds) for 8 hours (28,800 seconds).

* Disk Utilization
* Network Utilization
  + 1. **DB2 Tools**

DB2 has a number of tools that can be used to monitor the activity of the databases and instances.

* The Health Monitor / Health Center
* Snapshot Monitors / SQL Snapshot Functions
* Event Monitors

There are also other tools and logs available that provide information about the databases and instances including:

* The administration notification log [db2inst1.nfy]
* DB2DIAG.LOG
* Memory Visualizer

Note: The Administration notification log is a separate file on Linux and UNIX and incorporated into the Event Log on Windows.

**Event Monitoring**

Define Event Monitor using File

It can have one or more elements to monitor. The 'write to file' path must exist.

create event monitor my\_mon for database, tables, tablespaces, bufferpools, connections, statements, transactions,deadlocks,deadlock with details

write to file '/myhome/my\_mon\_dir' nonblocked append

Start:

set event monitor my\_mon state 1;

Stop:

set event monitor Q3MON state 0;

Query State:

select evmonname,event\_mon\_state(evmonname) as state from syscat.eventmonitors ;

Report :

db2evmon -db mydb  -evm my\_mon

Event Monitoring - using Pipes

To save disk space, event monitor records can be written to a pipe. This pipe should be read continuously as long as the event monitor is active.

1) Create a pipe, say /home/inst1/evmon\_pipe

2) Create event monitor say evmon1 using -  WRITE TO '/home/inst1/evmon\_pipe'  (The example given here is Unix pipe. DB2 supports windows pipes also). Remember, the event monitor cannot be NONBLOCKED.

3) Start the db2evmon as a continuous running process

             /home/db2inst1>nohup db2evmon -db databasename -evm evmon1 > /home/db2inst1/evmon1.out &

4) Activate the event monitor

The data should now be available in the file evmon1.out. Probably, you can have a window open with 'tail -f  /home/db2inst1/evmon1.out'

## Optimization and Tuning

### Explain & Index Advisor Tables

/home/db2inst1/sqllib/misc/EXPLAIN.DDL

Execute command: db2 –tvf EXPLAIN.DDL

## Create DDL Scripts

### Generate DDLs to Add ADD/DROP RESTRICT ON DROP

Add drop restrict

SELECT 'ALTER TABLE ' || rtrim(tabschema) || '.' || rtrim(TABNAME) || ' ADD RESTRICT ON DROP '|| ';' FROM SYSCAT.TABLES where tabschema = 'CAPTIVA' and type='T';

Remove drop restrict

SELECT 'ALTER TABLE ' || rtrim(tabschema) || '.' || rtrim(TABNAME) || ' DROP RESTRICT ON DROP '|| ';' FROM SYSCAT.TABLES where tabschema = 'CAPTIVA' and type='T'

alter table "SCHEMANAME"."WORKFLOWDEADLINE" drop RESTRICT ON DROP;

alter table "SCHEMANAME"."WORKFLOWDEADLINE" add RESTRICT ON DROP;

### Grant Privileges

select 'GRANT SELECT,UPDATE,DELETE,INSERT ON' || ' '|| rtrim(tabschema) || '.' || char(tabname,40) || 'TO GDBDEV'||';' FROM syscat.tables WHERE type = 'T' and tabschema='EMRMPLN' ORDER BY tabschema, tabname;

select 'GRANT SELECT ON' || ' '|| rtrim(tabschema) || '.' || char(tabname,40) || 'TO emprmldu,emprmalu'||';' FROM syscat.tables WHERE tabschema in ('SYSIBM','SYSCAT') ORDER BY tabschema, tabname;

### select 'delete from ' || ' '|| rtrim(tabschema) || '.' || char(tabname,40) || ';'

### FROM syscat.tables WHERE TABNAME LIKE 'SIB%' AND tabschema in ('SIBUS') ORDER BY tabschema, tabname;

### commit;

### select 'create alias emdscwu4.'|| char(tabname,40) ||' for '|| rtrim(tabschema) || '.' || char(tabname,40) ||';'

### FROM syscat.tables WHERE type = 'T' and tabschema in ('SCHEMANAME','EMDSCWU1','CUDB21','ADOBE') ORDER BY tabschema,

### tabname;

### commit;

### Grant privileges on VIEW

select 'GRANT SELECT ON' || ' '|| rtrim(viewschema) || '.' || char(viewname,40) || 'TO emprmldu,emprmalu'||';' FROM syscat.views WHERE viewschema='SCHEMANAME' ORDER BY viewschema, viewname;

### Create Table DDL

db2look -e -d empscdb1 -t casedata\_ivr -i db2inst1 -w db2inst1 o

db2look -d curamtst -a -o curamtst.ddl -e -x –l

Note: First statement is to create ddl for single table, second is for entire database.

**For entire schema – DB2INST3**

db2look -d curamtst -e -z db2inst3 -x -o sch\_struc\_db2inst3.out -u db2inst2

### DROP tables

SELECT 'DROP TABLE ' || rtrim(tabschema) || '.' || rtrim(tabname) || ';' AS Tablename FROM syscat.tables WHERE tabschema = 'ID' ORDER BY tabschema,tabname

### Drop Indexes

SELECT 'DROP INDEX DB2INST3.' || rtrim(indname)|| ';' FROM SYSCAT.INDEXES where tabschema='DB2INST3'

#### Drop triggers

SELECT 'DROP TRIGGER ' || rtrim(trigschema) || '.' || rtrim(trigname) || ';' AS Tablename FROM syscat.triggers WHERE trigschema = 'ID' ORDER BY trigschema,trigname

select \* from syscat.triggers where trigschema='ID'

#### Truncate table

select 'ALTER TABLE SCHEMANAME'||'.'||char(tabname,40)||' ' ||'ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE;' FROM syscat.tables WHERE type = 'T' and tabschema='SCHEMANAME' ORDER BY tabschema, tabname;

#### Drop sequences

SELECT 'DROP SEQUENCE ' || rtrim(seqschema) || '.' || rtrim(seqname) || ';' AS Tablename FROM syscat.sequences WHERE seqschema = 'EMSSCSM2' ORDER BY seqschema,seqname

#### Delete table

select 'DELETE FROM ' || rtrim(tabschema) || '.' || rtrim(tabname) || ';'

from syscat.tables

where tabschema='SCHEMANAME' and tabname like 'T%'

#### Generate Runstats

#### For all schema

SELECT 'RUNSTATS ON TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ' AND DETAILED INDEXES ALL;' FROM syscat.tables WHERE type = 'T' ORDER BY tabschema, tabname;

#### For specific schema

SELECT 'RUNSTATS ON TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ' AND DETAILED INDEXES ALL;' FROM syscat.tables WHERE type = 'T' and tabschema='SCHEMANAME' ORDER BY tabschema, tabname;

**With Distribution option**

SELECT ‘RUNSTATS ON TABLE ‘ || ‘MYSCH’ || ‘.’ || TABNAME ||  ‘ WITH DISTRIBUTION AND DETAILED INDEXES ALL;’  
FROM SYSCAT.TABLES  
WHERE TABSCHEMA = ‘MYSCH’  
AND TYPE = ‘T’;

#### Using db2look

db2look –d <database name> -m > output.out

### Create DDLs To Calculate Table Count

select 'select '''||rtrim(creator)||'.'||rtrim(name)|| ''' ,count(\*) from '||rtrim(creator)||'.'||rtrim(name) ||’;' from sysibm.systables where creator='EMRMPLN' group by creator,name;

SELECT ('SELECT COUNT(\*) ' ||'FROM ' || tabschema || '.' || tabname || ';')AS #rows FROM syscat.tables WHERE tabschema = 'SCHEMANAME' ORDER BY tabschema,tabname

### REORG CHECK

SELECT 'CALL SYSPROC.REORGCHK\_TB\_STATS (' || 'T' || ',' || rtrim(tabschema) || '.' || char(tabname,40) || ')' || ';'

FROM syscat.tables WHERE type = 'T' and tabschema in ('SCHEMANAME')

SELECT 'CALL SYSPROC.REORGCHK\_IX\_STATS (' || 'T' || ',' || rtrim(tabschema) || '.' || char(tabname,40) || ')' || ';'

FROM syscat.tables WHERE type = 'T' and tabschema in ('SCHEMANAME');

Note: Include “’” in T and tablename.

### Generate REORG

SELECT 'REORG TABLE ' || rtrim(tabschema) || '.' || char(tabname,40) || ' ;' FROM syscat.tables WHERE type = 'T' and tabschema in ('SCHEMANAME') ORDER BY tabschema, tabname;

db2 reorg indexes all for table homer.employee allow write access cleanup only

### Generate ADD/DELETE Constraints

#### Drop Foreign Constraints

SELECT 'ALTER TABLE schema.' || SUBSTR(TABNAME,1,50),' DROP FOREIGN KEY ' || CONSTNAME || ';'

FROM SYSCAT.REFERENCES where tabschema='DB2INST3' and tabname='ACTIVITY';

For generating the enable/disable for the tables:

– enable\_constraint

select ‘alter table ‘ || current server || ‘.’ || tabname ||  ‘ alter foreign key ‘ || constname || ‘ ENFORCED;’  
from syscat.references WHERE TABSCHEMA = CURRENT SERVER;

– Disable constraint

select ‘alter table ‘ || current server || ‘.’ || tabname ||  ‘ alter foreign key ‘ || constname || ‘ NOT ENFORCED;’  
from syscat.references WHERE TABSCHEMA = CURRENT SERVER;

#### Add Foreign Constraints

SELECT 'ALTER TABLE ' || rtrim(tabschema) || '.' || rtrim(TABNAME) || ' ADD CONSTRAINT ' || rtrim(CONSTNAME) || ' FOREIGN KEY (' || rtrim(FK\_COLNAMES) || ')' || ' REFERENCES ' || rtrim(tabschema) || '.' || rtrim(REFTABNAME) || '('||rtrim(PK\_COLNAMES)||')'

FROM SYSCAT.REFERENCES where tabschema = 'DB2INST3' and tabname='ACTIVITY';

Note: Remove “tabname =” from where condition if wish to generate the DDL for entire schema.

Another version:

SELECT   
'ALTER TABLE schema.' || SUBSTR(TABNAME,1,50), 'ADD CONSTRAINT ' || CONSTNAME, 'FOREIGN KEY (' || substr(FK\_COLNAMES,1,50) || ')',   
'REFERENCES schema.' || SUBSTR(REFTABNAME,1,17), ' ON DELETE ' ||

case deleterule  
when 'A' then 'NO ACTION'   
when 'C' then 'CASCADE'   
when 'N' then 'SET NULL'   
when 'R' then 'RESTRICT'  
end,   
' ON UPDATE ' ||   
case updaterule  
when 'A' then 'NO ACTION'   
when 'R' then 'RESTRICT'  
end   
FROM SYSCAT.REFERENCES where tabschema = 'schema' ;

### For Identity Resets:

SELECT ‘SELECT ”ALTER TABLE ‘ || A.TABNAME || ‘ ALTER ‘ || A.COLNAME || ‘ RESTART WITH ” ||  ‘ || ‘RTRIM(CHAR(MAX(’ || A.COLNAME || ‘)+1)) ‘ ||  
‘ FROM ‘ || A.TABNAME ||’;’  
FROM SYSCAT.COLUMNS A  
WHERE A.TABSCHEMA = CURRENT SCHEMA  
AND A.IDENTITY = ‘Y’  
ORDER BY A.TABNAME;

### Get Table Column

with tab(tsch,tname) as

(

values('CAPTIVA','ENVELOPE')

),

star(cno,concatcol) as

(

select colno,varchar(colname,1000) from syscat.columns,tab where colno=0 and tabname=tname and tabschema=tsch

union all

select colno,concatcol||','||colname from star s,syscat.columns c, tab where c.colno=cno+1 and tabname=tname and tabschema=tsch)

select 'SELECT '||concatcol||' FROM '||rtrim(tsch)||'.'||rtrim(tname) from star,tab where cno=(select max(cno) from star)

OR

with tab(tsch,tname) as

(

values('SYSCAT','SEQUENCES')

),

star(cno,cname) as

(

select -100,'SELECT' from sysibm.sysdummy1

UNION ALL

select colno,(case when colno>0 then ',' else '' end)||colname from syscat.columns,tab where tabschema=tsch and tabname=tname

union all

select 100000,' FROM '||rtrim(tsch)||'.'||rtrim(tname) from tab

)

select cname from star order by cno

Note: Replace schema and table name. In above query, SCHEMANAME is schema and ACTIVITY is table name.

### Find duplicate index on table

SELECT INDEX1.INDSCHEMA,

INDEX1.INDNAME,

INDEX1.DEFINER,

INDEX1.TABSCHEMA,

INDEX1.TABNAME,

INDEX1.COLNAMES,

INDEX1.COLCOUNT,

INDEX1.INDEXTYPE,

INDEX2.INDSCHEMA,

INDEX2.INDNAME,

INDEX2.DEFINER,

INDEX2.TABSCHEMA,

INDEX2.TABNAME,

INDEX2.COLNAMES,

INDEX2.COLCOUNT,

INDEX2.INDEXTYPE

FROM "SYSCAT"."INDEXES" as INDEX1,

"SYSCAT"."INDEXES" as INDEX2

WHERE INDEX1.TABSCHEMA = INDEX1.TABSCHEMA AND

INDEX1.TABNAME = INDEX1.TABNAME AND

NOT ( INDEX1.INDSCHEMA = INDEX1.INDSCHEMA AND

INDEX1.INDNAME = INDEX1.INDNAME AND

INDEX1.DEFINER = INDEX1.DEFINER ) AND

INDEX1.COLCOUNT <= INDEX2.COLCOUNT AND

INDEX1.INDSCHEMA || INDEX1.INDNAME || INDEX1.DEFINER <

INDEX1.INDSCHEMA || INDEX1.INDNAME || INDEX1.DEFINER

-- AND POSSTR(INDEX2.COLNAMES, INDEX1.COLNAMES) >= 0

## Database Sizing Scripts

* **Find Table Row Length**

select sum(length) from syscat.columns where tabname='ACTIVITY' and tabschema='STAGEELAB'

select substr(tabname,1,50),sum(length) from syscat.columns where tabschema='SCHEMANAME' group by tabname;

* **Calculate the size and maximum capacity of a database**

Steps:

* 1. Connect to the database for which you want to know the size and maximum capacity
  2. Execute the procedure GET\_DBSIZE\_INFO
  3. Close the connection

Syntax:

db2 call GET\_DBSIZE\_INFO (?,?,?,refresh-window)

Example:

Get the database size and capacity using a refresh window of 0 minutes:

1) db2 connect to curamtst

2) db2 "call GET\_DBSIZE\_INFO (?,?,?,0)"

3) db2 terminate

Note: This procedure available in DB2 UDB Version 8.2 onward.

DATABASECAPACITY would represent the maximum size your DB can attain considering the free space you have left on the disk where you're DB resides.  
  
DATABASESIZE would represent the current size of your database, not taking the logs into account.

**OR**

db2 "select \* from systools.STMG\_DBSIZE\_INFO"

* **Find Table Size for database**

select char(tabname,30), char(tabschema,25),npages from syscat.tables where tabname not like 'SYS%'

select substr(tabname,1,40),substr(tabschema,1,15),npages from syscat.tables where tabname not like 'SYS%'

* **Calculate Table Size**

Example: The table contains 15 million rows, and 4 columns: integer, char(10), TimeStamp, varchar(75).

Add up the maximum row length:

Integer = 4 Char(10) = 10 timestamp = 10 varchar(75) = 75 total = 99

If any of these are nullable that you will need to add 1 for each column that is nullable.

Determine what pagesize you want (minus overhead) and devide by max length to get rows per page (round down):

4096 = 4005 / 99 = 40

8192 = 8101 / 99 = 81

16384 = 16,293 / 99 = 164

32768 = 32677 / 99 = 330

Divide 15 million by records per page (round up) = number of pages needed

4096 = 375,000

8192 = 185,186

16384 = 91,464

32768 = 45,456

Multiply number of pages need by page size = number of byes needed

4096 \* 375,000 = 1,536,000,000

8192 \* 185,186 = 1,517,043,712

16384 \* 91,464 = 1,498,546,176

32768 \* 45,456 = 1,489,403,904

## General Database Administration SQLs

* **Determine Indexes stored in a tablespace**

SELECT DISTINCT TBCREATOR, TBNAME, name FROM SYSIBM.SYSINDEXES WHERE TBSPACEID = 6

* **Find Table Creation Details**

select tabname,tabschema,definer, create\_time,colcount,tableid,tbspace from syscat.tables

where tabschema='DB2INST1' and tabname in ('DOCUMENT','ENVELOPE') order by create\_time;

* **How to concatename INTEGER column with CHAR columns**

select nlst||varchar(char(caseid)) from stageelab.casedata\_flat1

* **Deleting duplicate record from table**

Assume you have a table tab1(a int,b char(10)) with the following records:

A           B   
----------- ------------------------------   
          1 Test1   
          1 Test1   
          2 Test2   
          2 Test2   
          3 Test3   
          4 Test4   
          5 Test5   
          5 Test5   
          4 Test4   
          5 Test5

10 record(s) selected.

If you want to delete, all duplicate records from the table, the following steps may be used:   
  
1) create view del\_tmp1 as select CASEID, RUNID, row\_number() over(partition by CASEID, RUNID)  as row# from schemaname.gopi   
  
select \* from del\_tmp1 ;  
  
  
A           B                              ROW#   
----------- ------------------------------ --------------------   
          1 Test1                                             1   
          1 Test1                                             2   
          2 Test2                                             1   
          2 Test2                                             2   
          3 Test3                                             1   
          4 Test4                                             1   
          4 Test4                                             2   
          5 Test5                                             1   
          5 Test5                                             2   
          5 Test5                                             3   
  
10 record(s) selected.   
  
  
2) delete from del\_tmp1 where row# >= 2   
  
This will eliminate duplicate rows from the base table tmp1.   
  
select \* from tmp1   
  
A           B   
----------- ------------------------------   
          1 Test1   
          2 Test2   
          3 Test3   
          4 Test4   
          5 Test5   
  
  5 record(s) selected.

* **How to delete 3rd row from table**

DELETE FROM TESTELAB.TEST1 WHERE SSN =

(SELECT SSN FROM

(SELECT ROW\_NUMBER () OVER () AS NUMBER, SSN FROM TESTELAB.TEST1)

AS TEMP WHERE NUMBER = 3);

* **Bulk Delete Rows from Table**

ALTER TABLE <tablename> ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE

OR

IMPORT FROM /dev/null of del replace into <tablename>  
  
Note: cannot be issued from a SQL Interface like JDBC.

* **Find creator/owner for table**

SELECT CREATOR||'.'||NAME FROM SYSIBM.SYSTABLES WHERE NAME = ‘ADVISE\_INSTANCE’ AND TYPE = 'T'

* **Find Total Number of tables in database**

Select count(\*) from sysibm.systables where type='T' and creator <> 'SYSIBM'

* **Find bufferpool for tablespace**

SELECT tbspace AS TABLESPACE, bpname AS BUFFERPOOL, npages AS NUM\_PAGES, estore as ESTORE FROM syscat.tablespaces, syscat.bufferpools WHERE syscat.tablespaces.bufferpoolid=syscat.bufferpools.bufferpoolid ORDER BY tbspace

OR

Select substr(bs.bpname,1,20) as BPNAME,bs.npages,bs.pagesize,substr(ts.tbspace,1,20) as TBSPACE from syscat.bufferpools bs join syscat.tablespaces ts on bs.bufferpoolid = ts. bufferpoolid;

db2 "select varchar(bpname,20) as bpname,npages,pagesize from syscat.bufferpools"  
  
BPNAME NPAGES PAGESIZE  
-------------------- ----------- -----------  
IBMDEFAULTBP 1000 4096  
BKDB\_BP 10000 8192  
BKDB\_DATABP 10000 8192  
TEMPSP2\_BP 2000 8192  
HISTBK\_BP 2000 8192  
UTILBK\_BP 2000 8192  
BKDBREPL\_BP 3000 8192

* **Find column count for all tables**

SELECT RTRIM(tabschema)||'.'||tabname, colcount FROM syscat.tables WHERE tabschema = 'DB2INST1'

* **List Tablespace Details**

Select TBSPACE,CREATE\_TIME,TBSPACEID,TBSPACETYPE,EXTENTSIZE,PREFETCHSIZE,PAGESIZE,DROP\_RECOVERY from sysibm.systablespaces

* **List Tablespace pages**

select tablespace\_id,tablespace\_name,total\_pages from table(snapshot\_container('curamtst',-2)) as s

* **Tablespace Details**

Select tbspace as "TableSpaceName", tbspaceid as "Identifier",

CASE tbspacetype

WHEN 'S' THEN 'System Managed Space'

WHEN 'D' THEN 'Database Managed Space'

END as "Table Space Type",

CASE datatype

WHEN 'A' THEN 'Permanent Data'

WHEN 'L' THEN 'Long Data'

WHEN 'T' THEN 'Temporary Table'

END as "Type of Data Stored"

from syscat.tablespaces

* **List tables**

select substr(tabschema,1,8) as "Qualified Name", substr(tabname,1,50) as "Table name",

CASE type

WHEN 'A' THEN 'Alias'

WHEN 'H' THEN 'Hierarchy Table'

WHEN 'N' THEN 'Nickname'

WHEN 'S' THEN 'Summary Table'

WHEN 'T' THEN 'Table'

WHEN 'U' THEN 'Typed Table'

WHEN 'V' THEN 'View'

WHEN 'W' THEN 'Typed View'

END as type,

CASE status

WHEN 'N' THEN 'Normal'

WHEN ' ' THEN 'Check Pending'

WHEN 'X' THEN 'Inoperative'

END as Status

from syscat.tables where tabschema ='SCHEMANAME’;

OR

## Table Functions

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Description** | **Query** |
| ENV\_GET\_SYS\_INFO | Get information about the **System** | SELECT OS\_NAME, OS\_VERSION, OS\_RELEASE,HOST\_NAME, TOTAL\_CPUS,CONFIGURED\_CPUS, TOTAL\_MEMORY MEMORY\_MB  FROM TABLE(SYSPROC.ENV\_GET\_SYS\_INFO()) AS SYSTEMINFO |
| ENV\_GET\_PROD\_INFO | Get information about installed **DB2 products** | SELECT INSTALLED\_PROD, IS\_LICENSED, PROD\_RELEASE FROM TABLE(SYSPROC.ENV\_GET\_PROD\_INFO()) AS PRODUCTINFO |
| ENV\_GET\_INST\_INFO | Get information about current **instance** | SELECT \* FROM TABLE(SYSPROC.ENV\_GET\_INST\_INFO()) AS INSTANCEINFO |
| DB\_PARTITIONS | Returns the contents of the **db2nodes.cfg** file in table form | select \* from TABLE(DB\_PARTITIONS ()) as dbpartitions |
| REG\_LIST\_VARIABLES | Returns the **DB2 registry** settings that are currently being used by the database partition | SELECT DBPARTITIONNUM,  SUBSTR(REG\_VAR\_NAME, 1, 25) AS "REG\_VAR\_NAME",  SUBSTR(REG\_VAR\_VALUE, 1, 15) AS "REG\_VAR\_VALUE",  IS\_AGGREGATE,  SUBSTR(AGGREGATE\_NAME, 1, 15) AS "AGGREGATE\_NAME",  LEVEL FROM TABLE(SYSPROC.REG\_LIST\_VARIABLES()) AS REGISTRYINFO |
| ADMIN\_LIST\_HIST | List information from the history file that is associated with the currently connected database partition | select eid, operation, start\_time from table(sysproc.admin\_list\_hist()) as listhistory |
| **Snapshot Function** | | |
| SNAP\_GET\_DB | Returns snapshot information from the database and detail\_log logical data groups. | select \* from TABLE(SNAP\_GET\_DB ('EMPSCDB1',-1)) as dbsnap |
| SNAPSHOT\_TBREORG | Table reorganization information | select \* from TABLE(SYSPROC.SNAPSHOT\_TBREORG ('EMPSCDB1',-1)) as REORGSnapshot  where table\_schema in ('SCHEMANAME','SCHEMANAME') order by table\_schema; |
| SNAP\_GET\_CONTAINER | Container info | select \* from table(sysproc.snap\_get\_container('curamtst',-1)) as contsnap |
| SNAP\_GET\_STO\_PATHS | Get database storage path | select \* from table(sysproc.SNAP\_GET\_STO\_PATHS('curamtst',-1)) as dbstorage |
| SNAP\_GET\_TAB | Get table snapshot | select \* from table(sysproc.SNAP\_GET\_TAB('EMPSCDB1',-1)) as tabsnap |
| SNAP\_GET\_TBSP | Get tablespace info | select \* from table(sysproc.SNAP\_GET\_TBSP('curamtst',-1)) as TSSNAP |
| SNAP\_GET\_TBSP\_PART | Get Tablespace Part info | select \* from table(sysproc.SNAP\_GET\_TBSP\_PART('curamtst',-1)) as TSPARTSNAP |
| SNAPSHOT\_AGENT | Get agent snapshot | select \* from table(sysproc.snapshot\_agent('curamtst',-1)) as agentinfo |
| SNAPSHOT\_APPL | General application information for each application that is connected to the database on the partition | select \* from table(SNAPSHOT\_APPL('EMPSCDB1',-1)) as applsnap;  select agent\_id, agent\_usr\_cpu\_time\_s, agent\_usr\_cpu\_time\_ms, appl\_con\_time,conn\_complete\_time,inbound\_comm\_address  from table(SNAPSHOT\_APPL('CURAMDB9',-1)) as applsnap; |
| SNAPSHOT\_APPL\_INFO | General application identification information for each application that is connected to the database on the partition | select agent\_id, client\_pid,appl\_name,appl\_id,auth\_id,client\_nname,client\_prdid,db\_name,execution\_id,corr\_token,tpmon\_client\_userid,tpmon\_client\_wkstn,tpmon\_client\_app  from table(SNAPSHOT\_APPL\_INFO('CURAMDB9',-1)) as applinfosnap; |
| SNAPSHOT\_BP | Buffer pool activity counters for the specified database.  Requires the buffer pool monitor switch | select substr(bp\_name,1,20) as BP\_NAME, int (( 1 - (decimal(pool\_data\_p\_reads) / nullif(pool\_data\_l\_reads,0) )) \* 100) as data\_hit\_ratio,int (( 1 - (decimal(pool\_index\_p\_reads) / nullif(pool\_index\_l\_reads,0) )) \* 100) as index\_hit\_ratio,int (( 1 - (decimal(pool\_data\_p\_reads + pool\_index\_p\_reads) / nullif( (pool\_data\_l\_reads + pool\_index\_l\_reads),0) )) \* 100) as BP\_hit\_ratio,int (( 1 - (decimal(pool\_async\_data\_reads + pool\_async\_index\_reads) / nullif( (pool\_async\_data\_reads + pool\_async\_index\_reads + direct\_reads),0) )) \* 100) as Async\_read\_pct,int (( 1 - (decimal(direct\_writes) / nullif(direct\_reads,0) )) \* 100) as Direct\_RW\_Ratio  from table (snapshot\_bp ('CURAMDB9', -1) ) as snapshot\_bp |
| SNAPSHOT\_CONTAINER |  | select \* from TABLE(SYSPROC.SNAPSHOT\_CONTAINER('EMPSCDB1',-1)) as ContainerSnapshot; |
| SNAPSHOT\_DATABASE | Database Information | select \* from TABLE(SYSPROC.SNAPSHOT\_DATABASE ('EMPSCDB1',-1)) as DatabaseSnapshot; |
| SNAPSHOT\_DBM | Database Manager Information | select \* from TABLE(SYSPROC.SNAPSHOT\_DBM (0)) as DBMSnapshot; |
| SNAPSHOT\_DYN\_SQL | Point-in-time statement information from the SQL  statement cache for the database | select \* from TABLE(SYSPROC.SNAPSHOT\_DYN\_SQL('EMPSCDB1',-1)) as DynSQLSnapshot; |
| SNAPSHOT\_FCM |  |  |
| SNAPSHOT\_FCMNODE |  |  |
| SNAPSHOT\_LOCK | Lock information at the database level, and application  level for each application connected to the database.  Requires the lock monitor switch | select \* from table(SNAPSHOT\_LOCK('EMPSCDB1',-1)) AS LOCKINFO |
| SNAPSHOT\_LOCKWAIT |  | select \* from table(SNAPSHOT\_LOCKWAIT('EMPSCDB1',-1)) AS LOCKINFO |
| SNAPSHOT\_QUIESCERS |  |  |
| SNAPSHOT\_RANGES |  |  |
| SNAPSHOT\_STATEMENT | Application information regarding statements for the  applications connected to the database on the partition.  This includes the most recent SQL statement executed (if  the statement monitor switch is set) | select \* from TABLE(SYSPROC.SNAPSHOT\_STATEMENT('EMPSCDB1',-1)) as StatementSnapshot; |
| SNAPSHOT\_SUBSECT |  | select \* from table(SNAPSHOT\_SUBSECT('EMPSCDB1',-1)) AS SUBSECTINFO; |
| SNAPSHOT\_SWITCHES | Returns information about the database snapshot switch state | select \* from table(SYSPROC.SNAPSHOT\_SWITCHES(-1)) as switchsnap |
| SNAPSHOT\_TABLE | Table activity information for each table that was  accessed by an application connected to the database.  Requires the table monitor switch | select \* from table(SYSPROC.SNAPSHOT\_TABLE('CURAMDB9',-1)) as tabsnap |
| SNAPSHOT\_TBREORG | Displays table reorganization information | select \* from table(SYSPROC.SNAPSHOT\_TBREORG('CURAMDB9',-1)) as tbreorg |
| SNAPSHOT\_TBS | Information about table space activity at the database level, the application level for each application connected  to the database, and the table space level for each table space that has been accessed by an application  connected to the database. Requires the buffer pool  monitor switch | select \* from table(SYSPROC.SNAPSHOT\_TBS('CURAMDB9',-1)) as tbssnap |
| SNAPSHOT\_TBS\_CFG | Returns configuration information from a table space snapshot | select \* from table(SYSPROC.SNAPSHOT\_TBS\_CFG('CURAMDB9',-1)) as tbscfg;  OR  select substr(tablespace\_name,1,120) as TBSPC\_NAME, used\_pages, free\_pages from table (snapshot\_tbs\_cfg ('EMPSCDB1', -1) ) as snapshot\_tbs\_cfg |

* **Retrieve 5 table names, which have most read and write activity on database CURAMTST**

select snapshot\_timestamp, table\_name, rows\_written, rows\_read, rows\_written + rows\_read as rows\_accessed from table (SNAPSHOT\_TABLE('CURAMTST', -1)) as T

order by rows\_accessed desc fetch first 5 rows only

* **Find the SQL statements that are taking the most time in the database**

SELECT stmt\_text, total\_exec\_time, num\_executions

FROM TABLE(SNAPSHOT\_DYN\_SQL('CURAMTST', -1)) as dynSnapTab

ORDER BY total\_exec\_time desc FETCH FIRST 5 ROW ONLY

* **Find 5 SQL statements with the worst average execution time**

SELECT CASE

WHEN num\_executions = 0 THEN 0

ELSE (total\_exec\_time / num\_executions)

END avgExecTime, num\_executions,stmt\_text

FROM TABLE(SNAPSHOT\_DYN\_SQL('CURAMTST', -1)) as dynSnapTab

ORDER BY avgExecTime desc FETCH FIRST 5 ROWS ONLY

* **Create a table based on the snapshot query and periodically stores the results of the query in the table**

create table table\_snap\_hist as

(select snapshot\_timestamp, table\_name, rows\_written, rows\_read,

rows\_written + rows\_read as rows\_accessed

from table (SNAPSHOT\_TABLE('CURAMTST', -1))as T) definition only;

insert into table\_snap\_hist

select snapshot\_timestamp, table\_name, rows\_written, rows\_read,

rows\_written + rows\_read as rows\_accessed

from table (SNAPSHOT\_TABLE('CURAMTST', -1))as T

order by rows\_accessed desc fetch first 5 rows only;

## Scalar Functions and Procedure

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Description** | **Query** |
| APPLICATION\_ID | Get Application ID | select application\_id() as appl\_id from sysibm.sysdummy1 |
| REORGCHK\_IX\_STATS | Returns a result set containing statistics that indicate whether or not one or more indexes need to be reorganized | call sysproc.reorgchk\_ix\_stats('s',stageelab.client\_indx) |
| REORGCHK\_TB\_STATS | Returns a result set containing statistics that indicate whether or not one or more tables need to be reorganized | call sysproc.reorgchk\_tb\_stats('t','stageelab.client') |

**Additional tables in DB2 UDB v8.2**

SYSCATV82.SNAPCONT

SYSCATV82.SNAPTBSPACEPART

SYSCATV82.SNAPCONT

SYSCATV82.SNAPTAB

SYSCATV82.SNAPDYNSQL

SYSCATV82.SNAPDB

SYSCATV82.SNAPTBSPACE

SYSCATV82.SNAPSTOPATHS

SYSCATV82.SNAPDETAILLOG

## Database Monitoring Scripts

### Database

* **Examine overall database usage pattern**

select db\_name, SNAPSHOT\_TIMESTAMP, rows\_read, rows\_selected,lock\_waits,lock\_wait\_time, lock\_wait\_time/nullif(lock\_waits,0) as avg\_wt\_time,deadlocks,lock\_escals,total\_sorts,total\_sort\_time,total\_sort\_time/nullif(total\_sorts,0) as avg\_sort\_time,sort\_overflows,sort\_overflows/nullif(total\_sorts,0) as pct\_ovflow\_sorts from table (snapshot\_database ('curamtst', -1) ) as snapshot\_database

* **Examine how many rows were read, written, and the number of overflow records accessed**

select substr(table\_schema,1,8) as Schema,substr(table\_name,1,30) as Table\_Name,rows\_read,rows\_written, overflow\_accesses from table (snapshot\_table ('EMPSCDB1', -1) ) as snapshot\_table;

select substr(table\_schema,1,8) as Schema,substr(table\_name,1,30) as Table\_Name,rows\_read,rows\_written, overflow\_accesses

from table (snapshot\_table ('EMPSCDB1', -1) ) as snapshot\_table

where table\_schema='SCHEMANAME'

and table\_name in

('ACTIVITY','AG','AGRID','CASE','CASEAUDIT','CASEDATAIVR\_EXCP',

'CASEDATA\_IVR','CASEDATA\_IVR1','CLIENT','CLIENT\_ADDRESSDATA',

'CLIENT\_ADDRESSELEMENTDATA','CLIENT\_PHONENUMBERDATA','PENDINGVR',

('ACTIVITY','ADDRESS','ADDRESSELEMENT','ALTERNATENAME','CASEHEADER',

'CASEPARTICIPANTROLE','CASERELATIONSHIP',

'CASEUSERROLE','CONCERN','CONCERNROLE','CONCERNROLEADDRESS',

'CONCERNROLEALTERNATEID','CONCERNROLEPHONENUMBER','EMAGAUTHREP',

'EMAGCLIENTLINK','EMASSISTANCEGROUP','EMPENDINGVR','PERSON','PHONENUMBER');

**Check Status of INT001**

select date(dateinitiated), tmstmpcompleted, dateinitiated, ((DAYS(tmstmpcompleted) - DAYS(dateinitiated)) \* 86400 + (MIDNIGHT\_SECONDS(tmstmpcompleted) - MIDNIGHT\_SECONDS(dateinitiated)))/60 TimeTakeninMin from schemaname.logtable where interfacename='INT001'

--and date(dateinitiated) ='01-26-2008'

and completionstatus='SUCCESSFUL';

* **Find top 10 read tables based on the number of rows read**

select substr(table\_schema,1,10) as tbschema,substr(table\_name,1,30) as tbname, rows\_read,rows\_written,overflow\_accesses,page\_reorgs from table (SNAPSHOT\_TABLE(' ',-1)) as snapshot\_table order by rows\_read desc fetch first 10 rows only

* **Find top 10 updated tables based on the number of rows written**

select substr(table\_schema,1,10) as tbschema, substr(table\_name,1,30) as tbname, rows\_read, rows\_written, overflow\_accesses,page\_reorgs from table (SNAPSHOT\_TABLE('EMPSCDB1 ',-1)) as snapshot\_table order by rows\_written desc fetch first 10 rows only

Note: Above tables are also likely candidates for atleast a runstats, if not a reorg and a runstats.

* **Look for indicators of exceptional growth**

select substr(tablespace\_name,1,120) as TBSPC\_NAME, used\_pages, free\_pages from table (snapshot\_tbs\_cfg ('curamtst', -1) ) as snapshot\_tbs\_cfg

* **Look for new or changed applications**

To look for changes applications you can see what SQL is running on your system over time, and look for new SQL that has not been run previously. To do this you can create a table as follows:

**create table SQLstmts ( stmt varchar(200), tstamp timestamp not null with default)**

You can then retrieve the SQL statements from the current package cache and insert them into a table for analysis using the following statement:

**insert into SQlstmts (stmt) select substr(stmt\_text,1,200) as SQL\_Stmt from table (snapshot\_dyn\_sql ('CURAMTST', -1) ) as snapshot\_dyn\_sql**

You can then examine this table for any SQL statements that have not been executed previously using the statement:

**select distinct stmt, count(stmt),tstamp from sqlstmts group by stmt, tstamp**

In the output of this statement, any statement with a count of 1, and the timestamp column showing the current date is one that has not been run previously.

* **Identify Runstat Time**

SELECT char(tabname,40),tabschema FROM syscat.tables WHERE type ='T' AND stats\_time is null and tabschema='DB2INST1';

SELECT char(indname,40),indschema FROM syscat.indexes WHERE stats\_time is null and indschema ='DB2INST1';

* **To list runstat times (oldest first)**

SELECT char(tabname,40), char(tabschema,30),stats\_time FROM syscat.tables WHERE type = 'T' ORDER by stats\_time

select substr(name,1,30),substr(creator,1,10),stats\_time from sysibm.systables where stats\_time < ((current timestamp) - 7 days) or stats\_time is null

select substr(name,1,30),substr(creator,1,10),stats\_time from sysibm.sysindexes where stats\_time < ((current timestamp) - 7 days) or stats\_time is null

* **To check pending status**

select tabschema,tabname,status,const\_checked from syscat.tables where tabname='CASEDATA\_IVR' and tabschema='STAGEELAB';

select tabschema,tabname,status,substr(const\_checked,1,1) as Foreign\_keys, substr(const\_checked,2,1) as Check\_Constraints,substr(const\_checked,5,1) as Summary\_Tables from syscat.tables where tabname='CASEDATA\_IVR' and tabschema='STAGEELAB';

### Tablespace Information

* **Get Tablespace Information**

select substr(tablespace\_name,1,20) Tablespace\_name, s.TABLESPACE\_ID,case t.TBSPACETYPE when 'S' then 'System Managed'

when 'D' then 'Database managed'

end as TBSManaged, s.total\_pages Total\_pages, s.used\_pages Used\_pages,

(s.total\_pages-s.USABLE\_PAGES ) Overhead\_pgs, s.NUM\_CONTAINERS No\_of\_containers, s.free\_pages Free\_pages,

case t.DATATYPE when 'A' then ' ALL typ perm '

when 'L' then ' Large '

when 'T' then ' System Temp'

when 'U' then ' User Temp '

end as TBStype, case s.TABLESPACE\_STATE

when 0 then ' Normal '

when 1 then ' Quiesced: SHARE '

when 2 then ' Quiesced: UPDATE '

when 4 then ' Quiesced: EXCLUSIVE '

when 8 then ' Load pending '

when 16 then ' Delete pending '

when 32 then ' Backup pending '

when 64 then ' Roll forward in progress '

when 128 then ' Roll forward pending '

when 256 then ' Restore pending '

when 512 then ' Disable pending '

when 1024 then ' Reorg in progress '

when 2048 then ' Backup in progress '

when 4096 then ' Storage must be defined '

when 8192 then ' Restore in progress '

when 16384 then ' Offline and not accessible '

when 32768 then ' Drop pending '

when 33554432 then ' Storage may be defined '

when 67108864 then ' Storage Definition is in (final) state '

when 134217728 then ' Storage Definition was changed prior to rollforward '

when 268435456 then ' DMS rebalancer is active '

when 536870912 then ' TBS deletion in progress '

when 1073741824 then ' TBS creation in progress ' end TABLESPACE\_STATE

from table(snapshot\_tbs\_cfg('CURAMINT',-1)) as s, sysibm.systablespaces t where s.tablespace\_name = t.tbspace ;

OR

select substr(tablespace\_name,1,20) tbspace,TABLESPACE\_ID,substr(container\_name,1,50) container\_name,total\_pages from table(snapshot\_container('CURAMINT',-1)) as cn;

* **Create the table named tablespaceinfo to store information from the table space snapshots for analysis.**

CREATE TABLE "DB2INST1"."TABLESPACEINFO" (

"TIMESTMP" TIMESTAMP,

"TABLESPACE\_NAME" CHAR(128),

"PCT\_FREE" INTEGER,

"TYPE" CHAR(5),

"CONTENTS" CHAR(5),

"TOTAL\_PAGES" INTEGER,

"USABLE\_PAGES" INTEGER,

"USED\_PAGES" INTEGER,

"FREE\_PAGES" INTEGER,

"PAGE\_SIZE" INTEGER

)

IN "USERSPACE1";

* **Insert the snapshot info into the tablespaceinfo table to be stored for analysis**

insert into tablespaceinfo

select

current timestamp,

substr(tablespace\_name,1,120) as TBSPC\_NAME,

(case

-- We can calculate pct free for DMS table spaces only as total\_pages is set to 0 for SMS by this stmt...

-- Therefore, check if DMS, and then calculate pct\_free as 1- (used/total) \* 100%

when tablespace\_type = 0 then (int( (1- (decimal(used\_pages)/decimal(total\_pages))) \* 100) )

-- For SMS set pct\_free to 100... Could set to any numeric value.

else 100

end) as pct\_free,

(case

-- Display the table space type, i.e. DMS or SMS as a string, not the numeric value in the info.

when tablespace\_type = 0 then 'DMS'

when tablespace\_type = 1 then 'SMS'

-- Only 0 and 1 are VALID, therefore return an error for anything else.

else 'Error'

end) as Managed\_By,

(case

-- Display the type of data that can stored in the table space, i.e. TEMP, LARGE/LOB OR ALL,not the numeric value in the info.

when tbs\_contents\_type = 2 then 'TEMP'

when tbs\_contents\_type = 1 then 'LARGE'

when tbs\_contents\_type = 0 then 'ALL' end) as Data\_Type,

-- Also return the total\_pages using the heading ALLOCATED PAGES,

total\_pages as allocated\_pages,usable\_pages, used\_pages, free\_pages, page\_size

from table (snapshot\_tbs\_cfg ('CURAMTST', -1) ) as snapshot\_tbs\_cfg

order by pct\_free;

* **Query TABLESPACE INFO Table**

select tablespace\_name, date(timestmp) as dte, pct\_free from tablespaceinfo

group by tablespace\_name, pct\_free, timestmp ;

select \* from tablespaceinfo

group by tablespace\_name, pct\_free, timestmp ;

### Tablespace Container information script

* **Get container info**

select substr(tablespace\_name,1,20) tbspace,TABLESPACE\_ID,substr(container\_name,1,50) container\_name,total\_pages from table(snapshot\_container('CURAMINT',-1)) as cn;

**OR**

select distinct ' SET TABLESPACE CONTAINERS FOR '||rtrim(char(t.TABLESPACE\_ID))|| case t.TABLESPACE\_TYPE when 1 then ' USING ( PATH ' when 0 then ' USING ( FILE ' end ||

case t.NUM\_CONTAINERS when 1 then

case t.TABLESPACE\_TYPE when 1 then substr(container\_name,1,length(container\_name))

when 0 then substr(container\_name,1,length(container\_name)) ||' '||rtrim(char(c.usable\_pages))

end ||' ) ;'

when 2 then substr(container\_name,1,length(container\_name)-6)||'01.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'02.DBF '||char(c.usable\_pages)||');'

when 3 then substr(container\_name,1,length(container\_name)-6)||'01.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'02.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'03.DBF '||char(c.usable\_pages)||');'

when 4 then substr(container\_name,1,length(container\_name)-6)||'01.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'02.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'03.DBF '||char(c.usable\_pages)||',FILE '

||substr(container\_name,1,length(container\_name)-6)||'04.DBF '||char(c.usable\_pages)||');'

end from table (snapshot\_container (' ', -1) ) as c, table(snapshot\_tbs\_cfg(' ',-1)) as t

where c.tablespace\_name = t.tablespace\_name;

* **Report the containers, their size and type associated with each tablespace**

select substr(tablespace\_name,1,12) as TBSPC\_Name, substr(Container\_name,1,67) as Cont\_Name,

(case

when container\_type = 0 then 'SMS Directory'

when container\_type = 6 then 'DMS File'

else 'DMS Device'

end) as Container\_Type,

usable\_pages

from table (snapshot\_container (' ', -1) ) as snapshot\_container;

Note: Group by tablespace\_name to get all containers for a table space together as the containers will be unique in this report. Set the database name to NULL to get info on currently connected database.

### Bufferpool - Tablespace Information script

Below information can be used to help size the bufferpools more appropriately.

* **Report buffer pool name and size along with the size and table space assigned to the bufferpool**

select substr(b.bpname,1,12) as BufferPool, b.npages as BP\_Pages, substr(t.tbspace,1,12) as TableSpace,

usable\_pages as TBSPC\_Pages

from table (snapshot\_tbs\_cfg ('CURAMTST', -1) ) as snapshot\_tbs\_cfg , syscat.tablespaces t, syscat.bufferpools b

where t.bufferpoolid = b.bufferpoolid and t.tbspace = tablespace\_name

group by b.bpname, t.tbspace, usable\_pages, npages;

Note: Group by bpname first to get all table spaces for a bufferpool together as the table spaces will be unique in this report.

* **Bufferpool Hit Ratio**

select substr(bp\_name,1,20) as BP\_NAME, int (( 1 - (decimal(pool\_data\_p\_reads) / nullif(pool\_data\_l\_reads,0) )) \* 100) as data\_hit\_ratio,int (( 1 - (decimal(pool\_index\_p\_reads) / nullif(pool\_index\_l\_reads,0) )) \* 100) as index\_hit\_ratio,int (( 1 - (decimal(pool\_data\_p\_reads + pool\_index\_p\_reads) / nullif( (pool\_data\_l\_reads + pool\_index\_l\_reads),0) )) \* 100) as BP\_hit\_ratio,int (( 1 - (decimal(pool\_async\_data\_reads + pool\_async\_index\_reads) / nullif( (pool\_async\_data\_reads + pool\_async\_index\_reads + direct\_reads),0) )) \* 100) as Async\_read\_pct,int (( 1 - (decimal(direct\_writes) / nullif(direct\_reads,0) )) \* 100) as Direct\_RW\_Ratio from table (snapshot\_bp ('curamtst', -1) ) as snapshot\_bp

<http://www.dbazine.com/db2/db2-disarticles/gunning1>

## Troubleshooting

<ftp://ftp.software.ibm.com/ps/products/db2/info/vr82/pdf/en_US/db2p0e82.pdf>

## Database Commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| db2level  OR  select \* from table(env\_get\_inst\_info()) t | Shows current version and Service Level of the installed DB2 product |
| db2licm -l | DB2 License Manager  (-l is for license; returns edition level, -v is DB2 version) |
| db2support . -d <database name> -c | Collect all DB2 and system diagnostic data automatically |
| Adp | Find database memory usage |
| ps –ef | grep db2sysc | Check instance process running |
| db2 get instance or echo $DB2INSTANCE | Find current attached instance name |
| db2 autoconfigure apply none | Calculates and displays optimum values for DB CFG, DBM CFG and Buffer pool settings |
| db2pd | DB2 problem determination tool |
| GET ADMIN CFG |  |
| GET DBM CFG |  |
| GET DBM CFG SHOW DETAIL | Lists the current values of the database manager configuration file |
| LIST NODEGROUPS SHOW DETAIL |  |
| GET DB CFG FOR CURAMTST |  |
| GET DB CFG FOR CURAMTST SHOW DETAIL |  |
| GET CLI CFG |  |
| LIST ACTIVE DATABASES |  |
| LIST ADMIN NODE DIRECTORY |  |
| LIST DATALINKS MANAGERS FOR DATABASE CURAMTST |  |
| LIST HISTORY SINCE 20070307 FOR DB CURAMTST |  |
| LIST DB DIRECTORY |  |
| DB2SET –LR |  |
| GET SNAPSHOT FOR DBM GLOBAL |  |
| GET SNAPSHOT FOR ALL DATABASES GLOBAL |  |
| LIST DCS DIRECTORY |  |
| LIST DCS APPLICATIONS |  |
| LIST APPLICATIONS FOR DATABASE CURAMTST GLOBAL SHOW DETAIL |  |
| LIST COMMAND OPTIONS |  |
| LIST DATALINKS MANAGERS FOR DATABASE CURAMTST |  |
| LIST DRDA INDOUBT TRANSACTIONS |  |
| LIST NODES |  |
| DATABASE PARTITION NUMBER |  |
| LIST PACKAGES FOR ALL |  |
| LIST TABLES FOR ALL SHOW DETAIL |  |
| LIST INDOUBT TRANSACTIONS |  |
| LIST NODE DIRECTORY |  |
| LIST ODBC DATA SOURCES |  |
| QUERY CLIENT |  |
| LIST TABLESPACES SHOW DETAIL |  |
| LIST TABLESPACE CONTAINERS FOR <> SHOW DETAIL |  |
| db2 create schema <schema name> | Create schema |
| RENAME TABLE STAGEELAB.CLIENT TO CLIENT\_BKP | Rename table |
| REORGCHK UPDATE STATISTICS ON TABLE ALL | To collect updated statistics and identify if tables or indexes need to be REORGED |
| REORGCHK CURRENT STATISTICS ON TABLE ALL | To identify if tables or indexes need to be reorged based on the current statistics |
| REORG TABLE schema.tablename | To perform a REORG on a table |
| REORG INDEXES ALL FOR TABLE schema.tablename | To perform a REORG on the all the indexes for a table |
| db2set –all | Lists all currently set DB2 registry variables |
| db2set –g | -i **variable**=**value** | Sets the specified DB2 registry variable at either the global (-g) or instance (-i) level |
| UPDATE DBM CFG USING **config\_param value** | Sets the specified database manager configuration parameter to the specified value. |
| GET DB CFG FOR **db\_name** [SHOW DETAIL] | Lists the current values of a specific database's configuration file.. |
| UPDATE DB CFG FOR **db\_name** USING **config\_param value** | Sets the specified database manager configuration parameter to the specified value. |
| RUNSTATS ON TABLE **schema.table** | Collects statistics for a specific table. |
| RUNSTATS ON TABLE **schema.table** AND INDEXES ALL | Collects statistics for a specific table and all of its indexes. |
| RUNSTATS ON TABLE **schema.table** AND SAMPLED DETAILED INDEXES ALL | Collects statistics on a specific table and all of its indexes using extended index statistics and a CPU sampling technique, which is useful for very large indexes (1+ million rows) where time available for RUNSTATS is limited. |
| RUNSTATS ON TABLE **schema.table** WITH DISTRIBUTION | Collects additional statistics on a specific table (and optionally, specific columns), which is useful when data is not evenly distributed. |
| catalog admin tcpip node NODE000 remote 9.73.99.70 server 50000 remote\_instance db2inst1 system bosicontnt.usin.ibm.com ostype linux | Catalog Node |
| catalog tcpip node NODE0001 remote bosicontnt.usin.ibm.ibm SERVER db2c\_db2inst1 REMOTE\_INSTANCE db2inst1 SYSTEM 9.73.99.70 ostype linux | Catalog Node |
| catalog database curamtst as curamtst at node NODE0001 | Catalog Database |
| backup database curamtst online to /home/db2inst1/db2dbbkp | Database online backup |
| db2 "export to captiva\_envelop.del of del modified by coldel| datesiso select \* from captiva.envelope" | Export table |

**Set-up instance for Autostart**

db2set -i udb11 DB2AUTOSTART=TRUE

db2iauto -on udb11

Note: db2iauto will setup DB2AUTOSTART=YES

**Rebind**

After performing a REORG and RUNSTATS, you will want to REBIND all database packages so that their static SQL can take advantage of the most recent system statistics. Use DB2RBIND to rebind all the database packages:

db2rbind **dbname** -l **logfile.out** ALL

REBIND can be used to rebind single packages.

CALL SYSPROC.REBIND\_ROUTINE\_PACKAGE('P', 'MYSCHEMA.MYPROC', 'CONSERVATIVE')

<http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp>

# 

# Tools and Software

### Tools and Software

|  |  |
| --- | --- |
| **Vendor** | **Tool Name** |
| Embarcadero Technologies Inc | DBArtisan v7.1 |
| Quest Software | Quest Central for DB2 v2.1.0.262 |
| BMC | DBXray v2.0.01, PATROL v6.4.0, SmartDBA Cockpit v1.6.01, Space Expert v1.1.00 |
| Unicenter | Fast Load and Unicenter Fast Recovery for DB2 |
| LECCO | SQL Expert Pro v2.6.1 for DB2 UDB |
| Precise Software | Precise/Indepth for IBM DB2 v2.0, DGI, Flight Deck v1.4 |
|  |  |

SNAPSHOT FUNCTIONS refer:

**Note:** For more information on table functions, refer to Part 3,”Using snapshot monitor” in *System Monitor Guide and Reference,* SC09-4847-00.

**Rules to be checked before taking Backup:**

1 ) You can only do an online backup if you are using archive logging rather than circular logging. This means you can do backups while connections are still active and transactions continue to get logged and you can recover to a certain point in time rather than just to the point of the last backup.

### [HADR performance - part 1](http://udbworld.blogspot.com/2009/10/hadr-performance-part-1.html)

HADR stands for high availability disaster recovery .This solution is used for disaster recovery purpose on db2 udb databases.HADR should be configured properly inorder to have optimal performance:  
  
1)HADR synchronization mode:HADR can be run in 3 different modes SYNC,NEARSYNC,ASYNC.SYNC mode gives the best protection to data.In this mode primary has to wait until the changes are committed and written on the standby.Primary waits for the acknowledgement from the standby server.In NEARSYNC mode, standby sends acknowledgement as soon as the logs are in memory of standby server.And in ASYNC mode , primary does not wait for any kind of acknowledgement from the standby.Proper synchronization mode has to be chosen for optimal performance  
  
2)DB2\_HADR\_BUF\_SIZE:This registry variable controls the size of the receive buffer .Receive buffer is the area of memory where the logs are received before they are replayed.You can use the db2pd -db dbname -hadr on standby to monitor the usage of receive buffer.If you see it reaching 100 during the workload, you need to increase the value of DB2\_HADR\_BUF\_SIZE  
  
3)DB2\_HADR\_SOSNDBUF and DB2\_HADR\_SORCVBUF:There are the socket send buffer size and socket receive buffer size respectively.If the size for these parameters is too small then the full bandwidth can not be utilized.Generally increasing this to a bigger value would not impact performance negatively.  
  
  
4)Logfilsz:Size of the logfile plays an important role in the performance,Generally this size should be few hundred MB.

Prod URL’s

The URL's are :

**S990apap01 URLs:**

Application URL (Node 1) <http://s990apap01.iem.local:10212/HCSSApplication>

Request URL (Node 1) <http://s990apap01.iem.local:10212/HCSSRequest>

Agency Portal <http://s990apap01.iem.local:10212/HCSSPartner>

Web Services URL (Node 1) <http://s990apap01.iem.local:10212/CuramWS/services>

**s990apap02 URL's :**

Application URL (Node 1) <http://s990apap02.iem.local:10212/HCSSApplication>

Request URL (Node 1) <http://s990apap02.iem.local:10212/HCSSRequest>

Agency Portal <http://s990apap02.iem.local:10212/HCSSPartner>

Web Services URL (Node 1) <http://s990apap02.iem.local:10212/CuramWS/services>

**Curam Url's From Web servers :**

Internal Application URL : <https://s990lpap02.iem.local/HCSSApplication/logon.jsp>

External Application : <https://www.ifcem.com/HCSSRequest/en_US/External_screeningHomePage.do>

Agency Portal URL :

<https://www.ifcem.com/HCSSPartner/logon.jsp>

ssh -l iudb24 s990apap02.iem.local

cd /home/SCHEMANAME/recycle

**Usage Example :**

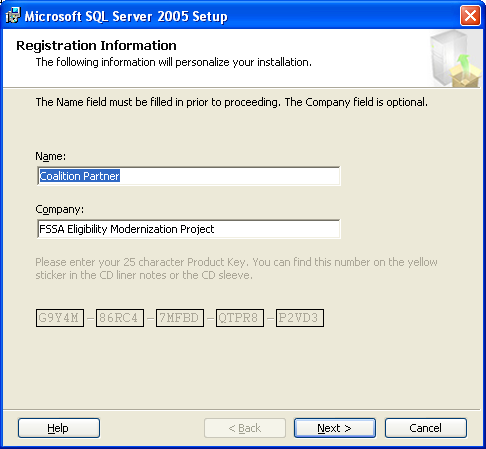
./recycleApplication.sh s990apap02 HCSSPartner

OR

./recycleApplication.sh s990apap01 HCSSRequest

OR

./recycleApplication.sh s990apap01 HCSSPartner



Db2 download link

<http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg21256059>

DB29 Version Commands:

**How to Find who is holding the Locks:**

SELECT SMALLINT(AGENT\_ID) AS WAITING\_ID,  
 SUBSTR(APPL\_NAME, 1,10) AS WAITING\_APP,  
 SUBSTR(AUTHID,1,10) AS WAITING\_USER,  
 SMALLINT(AGENT\_ID\_HOLDING\_LK) AS HOLDER\_ID,  
 LOCK\_MODE AS HELD,   
 LOCK\_OBJECT\_TYPE AS TYPE,   
 LOCK\_MODE\_REQUESTED AS REQUEST  
FROM SYSIBMADM.LOCKWAITS

Non-index monthly report path

/etldata/export/scripts/non\_index\_doc

SIT E Server

fssesbq01vw.iem.local

password = password

**UAT-D Job Run**

**Fssesbz01pw**

**To log ion from esb box:**

[\\Fssesbz01pw.iem.local\e$](file:///\\Fssesbz01pw.iem.local\e$)

**To log into WAS box:**

\\fsswasz21pw.iem.local\e$\Curam\Batch

CURAM SERVER NAME : fsswasz21pw.iem.local

Password :: ces\batchuserstage / Pa$$word123

**PROD ESB BOX:**

WAS JOB triggers:

**E:\Curam\Batch\launchScript.bat DEV\_B runScanTasks**

fssesbp01pw

batchuserprod/Pa$$word123

\\fssesbp01pw\e$\Interfaces\Interfaces\_A\data

CURAM SERVER NAME : fsswasp21pw.iem.local

Password :: ces\batchuserprod

\\fsswasp21pw\e$

To see logs go to E:\Curam\Batch\tmp\ runTemporalEvents.log

PS E:\Interfaces\Scripts> ."E:\Interfaces\Interfaces\_A\scripts\functions\allFunctions.ps1"

PS E:\Interfaces\Scripts> $inputfile="SC\_TEMPORAL\_EVENTS"

PS E:\Interfaces\Scripts> update-status "S"

PS E:\Interfaces\Scripts> $inputfile="PROD\_INT014\_Daily.del"

PS E:\Interfaces\Scripts> update-status "S"

To run from esb server

Exec and then press TAB space “batchjobname”

**SIT D ESB BOX Details:**

\\fssesbq01vw\e$\Interfaces\Interfaces\_A\data

fssesbq01vw

ces\batchuserstage

Pa$$word123

**SIT CURAM BOX DETAILS:**

Fsswasq01vw.iem.local

DEV F BOX BELOW

\\fssesbd01vw\e$\Interfaces\Interfaces\_C\data

**DEF ESB BOX BELWOW**

[\\fssesbd01vw\e$](file:///\\fssesbd01vw\e$)

BACKUP and RESTORE Chapter:

**Full online backup :**  To take a full online backup at first the database configuration needs to be checked for the LOGARCHMETH1 is set or not.It needs to be set.

USEREXIT/LOGRETAIN should be ON.

In db2diag.log you can see whether the backup was full online or not. If it is full online backup it will say Started Online backup while you start the backup.By list history backup command also you can check the type of backup.This backup image is independent. Archive logs are required to restore the point in time database.

**Online Incremental backup :**  TRACKMOD should be ON to enable online incremental backup.

In db2diag.log you can see whether the backup was incremental online or not. If it is incremental online backup it will say Started Online incremental backup while you start the backup.By list history backup command also you can check the type of backup.A full online or offline backup is required to make a base database copy on which these incremental image can be added to restore point in time recovery.

**Delta backup :**  In case of delta backup every delta backup image is required to restore the database.

**Offline backup :**  It is a independednt backup.. Connections needs to be forced before starting the offline backup otherwise "Database is in use" error will come and backup will fail.

**DB2 FEDERATION:**

# A Fast Way to Get Started with DB2 Federation

* Posted by [david t](http://www.channeldb2.com/profile/SideshowDave) on May 13, 2010 at 9:30pm
* [View Blog](http://www.channeldb2.com/profiles/blog/list?user=SideshowDave)

Data federation (or virtualization) is a data integration technology that lets you build what some people think of as a database cloud. It's found in DB2, InfoSphere Warehouse, and a few other IBM products. It can be extended or used stand-alone through IBM's InfoSphere Federation Server.

However, even with it available many different ways, I regularly meet people who know little about it and are under the impression that it's difficult to learn. It's not :) That got me thinking about the fastest way to get started with it. Personally, I think it's best to set up a simple scenario and go from there. That's what I'm going to show in this post. It should take about 15 minutes max. You can then extend your scenario using the documentation in [DB2's Information Center](http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.swg.im.iis.db.prod.fed.nav.doc/dochome/iiypfnav_dochome.html) or other posts I have.

Naturally, if you prefer an approach where you learn more of the underlying terminology and concepts as you go, see [this very good developerWorks article about federation](http://www.ibm.com/developerworks/data/library/techarticle/dm-0506lin/). Otherwise, read on.

I have instructions and a single SQL script that shows you the basics. You need a DB2 instance before you start. The instructions have you create two databases under your instance and then run the script. The script creates federation objects and provides an example of what's called a "federated query" or "distributed join". I'll explain more about those later in the post.

After you finish, if you want to spend a few more minutes growing your new-found skills :) see one of the following:

**o** [Using DB2 cache tables](http://www.channeldb2.com/profiles/blogs/how-easy-is-it-to-set-up-cache) (a type of distributed caching)

**o** [Federating DB2 and PostgreSQL](http://www.channeldb2.com/profiles/blogs/federating-db2-and-postgresql) (requires InfoSphere Federation Server)

**o** [Federating DB2 and solidDB](http://www.channeldb2.com/profiles/blogs/using-data-federation-with) (requires InfoSphere Federation Server and solidDB)

Your fast start is the seven steps in the next section. Note that you can do all of this with DB2 Express-C.

**Do It Yourself**

1. Enable federation in your DB2 instance by issuing the following command from a DB2 command window:

|  |  |
| --- | --- |
|  | db2 update database manager configuration using federated yes |

2. If your DB2 instance was running you must stop it and then start it again.

|  |  |
| --- | --- |
|  | db2stop db2start |

3. Create the DB2 Sample database if you haven't already. From a DB2 command window, issue:

|  |  |
| --- | --- |
|  | db2sampl |

4. Create an empty database (one with no user tables). From a DB2 command window, issue:

|  |  |
| --- | --- |
|  | db2 create database clouddb |

5. Copy and paste the script at the bottom of this post into a file named, say, mycloud.sql

6. Edit the file and make the changes described in the script's opening comments.

7. Run your SQL script by issuing the following command from a DB2 command window:

|  |  |
| --- | --- |
|  | db2 -vtf mycloud.sql > mycloud.txt |

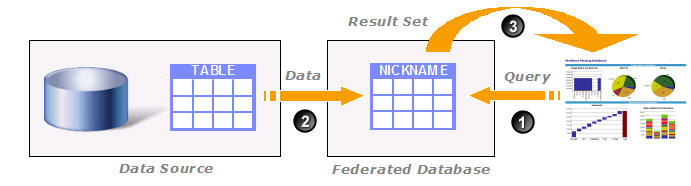
That's it. You've created a federated database (clouddb) and federated it with two tables from another database (sample). You've also joined those tables in a single query. You can now try some other joins, federate more tables from sample, or federate and join tables from yet another database.

However, if you're very new to federation, you may want to understand a two things from the script - what a nickname is and what a distributed join is.

**What is a Nickname?**

You may have noticed the script contains two "Create Nickname" statements. A nickname is a 'virtual' table in one database that points to a physical table in another database. The nickname does not contain data itself. It just has information about the physical table. You can use a nickname in an SQL statement anywhere a table name can be used. Under the covers, the federated database server issues what ever SQL is necessary to work with the physical table.

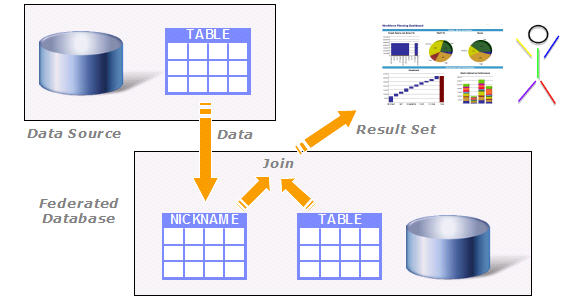
For example, in the following graphic, a select is issued to the federated database. A nickname is specified in the from-clause. This is the same thing shown in the script's first query. The federated database server retrieves just enough data from the physical table to satisy the query. The result set is returned just as though the data were stored in the federated database.



You can also use nicknames in inserts, updates, deletes, predicates, and more. However, one of the most interesting uses is in a distributed join.

**What Is a Distributed Join?**

A distributed join is one that references tables from multiple databases. More specifically, the query uses at least one nickname. For example, in the following graphic, a join is issued to the federated database. One of the referenced tables resides locally in the federated databse. The other table resides in another database and is pointed to by a nickname in the federated database. This is the same thing shown in the script's second query.



When the federated database server processes the join, its optimizer determines how to retrieve the least amount of data possible to perform the join.

There's more you can learn about federation, but I think that would take us well past a fast start :)

**The Script**

|  |  |
| --- | --- |
|  | --  -- Before using this script, make the following changes: -- -- 1. Replace all occurances of 'tolleson' with a user name -- you can use to connect to your DB2 databases. -- -- 2. Replace all occurances of 'mypassw0rd' with the -- password that goes the the user name you provided. --  connect to clouddb;  -- -- First, let's create one table in the federated database. --  drop table greeting; create table greeting (deptno char(3), dept\_greeting char(6)); insert into greeting values('E21', 'Howdy');  -- -- Now, let's create our federated objects --  drop wrapper drda; create wrapper drda;  --  -- The add dbname option is a DB2 alias from the database  -- directory where the federated server runs. --  create server mysample type db2/udb  version '9.7'  wrapper drda  authid "tolleson"  password "mypassw0rd"  options(  add dbname 'SAMPLE');  -- -- The 'remote' authid and password are the user and password -- the federated server uses to connect to the server -- 'mysample' when processing SQL for the user 'tolleson'. --  create user mapping for tolleson server mysample options ( remote\_authid 'tolleson', remote\_password 'mypassw0rd');  -- -- These nicknames point to tables in the Sample database. --  create nickname myemp for mysample.tolleson.employee; create nickname mydept for mysample.tolleson.department;  -- -- Query 1: Select from just one nickname. --  select ee.empno, ee.firstnme, ee.workdept from myemp ee order by ee.workdept;  -- -- Query 2: Join our local table with a nickname --  select gg.dept\_greeting, ee.firstnme from greeting gg, myemp ee where gg.deptno=ee.workdept order by ee.lastname;  -- -- Query 3: Join two nicknames --  select ee.empno, ee.lastname, ee.firstnme, dd.deptname from myemp ee, mydept dd where dd.deptno='E21' and ee.workdept=dd.deptno order by ee.lastname;  Commit; |

**SQL Server Commands**

SELECT t.name AS table\_name,

SCHEMA\_NAME(t.schema\_id) AS schema\_name,

c.name AS column\_name,c.max\_length as maxlength,x.name as datatype

FROM sys.tables AS t

INNER JOIN sys.columns c ON t.OBJECT\_ID = c.OBJECT\_ID

INNER JOIN sys.types AS x ON c.user\_type\_id = x.user\_type\_id;

SELECT a.[name] as 'Table',b.[name] as 'Column',c.[name] as 'Datatype',b.[length] as 'Length',

CASE

   WHEN b.[cdefault] > 0 THEN d.[text]

   ELSE NULL

  END as 'Default',

CASE

   WHEN b.[isnullable] = 0 THEN 'No'

   ELSE 'Yes'

  END as 'Nullable'

FROM  sysobjects a

INNER JOIN syscolumns b ON  a.[id] = b.[id]

INNER JOIN systypes c ON  b.[xtype] = c.[xtype]

LEFT JOIN syscomments d ON  b.[cdefault] = d.[id]

WHERE a.[xtype] = 'u'

--'u' for user tables, 'v' for views.

--and a.[name]='table name'

--AND  a.[name] <> 'tablename'

ORDER BY a.[name],b.[colorder];

BULK

INSERT [FSSA\_Custom\_I3].[dbo].[FSSA\_OFFICE\_HOURS]

FROM 'D:\Interfaces\1.txt'

WITH

(

FIELDTERMINATOR = '|',

ROWTERMINATOR = '\n'

)

GO

**SQL server Backup and Restore Commands:**

<http://stackoverflow.com/questions/6217679/create-restore-database-from-backup-sql-server-express>

<http://codeonaboat.wordpress.com/2012/02/16/sql-server-2008-creating-a-database-from-a-bak-file/>

<http://social.msdn.microsoft.com/Forums/en/sqldisasterrecovery/thread/4d419f83-9415-4669-a1c4-1e57a2587cb0>

RESTORE DATABASE YourDatabaseName

FROM DISK = N'(path to your BAK file)'

WITH FILE = 1,

MOVE N'(your DB name)' TO N'(your SQL path)database.MDF',

MOVE N'(your DB name)\_LOG' TO N'(your SQL path)database\_LOG.LDF',

NOUNLOAD,

REPLACE,

STATS = 10

GO

BACKUP DATABASE [AdventureWorks] TO

DISK = N'\\nas\Backup\L40\SQL2005\AdventureWorks\_backup\_200702120215.bak'

WITH NOFORMAT, NOINIT, NAME = N'AdventureWorks-Full Database Backup',

SKIP, NOREWIND, NOUNLOAD, STATS = 10

RESTORE DATABASE [AdventureWorksNew]

FROM DISK = N'\\nas\Backup\L40\SQL2005\AdventureWorks\_backup\_200702120215.bak'

WITH FILE = 1,

MOVE N'AdventureWorks\_Data' TO N'C:\Data\MSSQL.1\MSSQL\Data\AdventureWorksNew\_Data.mdf',

MOVE N'AdventureWorks\_Log' TO N'C:\Data\MSSQL.1\MSSQL\Data\AdventureWorksNew\_Log.ldf',

NOUNLOAD, STATS = 10

SELECT name, physical\_name AS current\_file\_location

FROM sys.master\_files

## The restoration process

Note that, for the purpose of this article, I am assuming that your database recovery mode is set to FULL.

The first step in the process is to perform a tail-log backup. You want to perform this type of backup before a database restore to ensure that any records that have changed since the last backup are available to be included in the restore process.

Next you should locate where the database backup files are stored on the machine or the network. It may be a good idea to copy these files to your target server if you are going to be restoring the database on a different server. In the backup file location, find the very last full database backup that was completed (these files usually end with the extension .bak); you need to restore this full backup. The script below applies the full backup file to the NewDatabase database:

RESTORE DATABASE NewDatabase  
FROM DISK = 'D: \BackupFiles\TestDatabaseFullBackup.bak'  
WITH  
MOVE 'PreviousDatabase' TO 'D:\DataFiles \TestDatabase.mdf',  
MOVE 'PreviousDatabase\_log' TO 'D:\DataFiles \TestDatabase\_Log.ldf',  
NORECOVERY

The code specifies that the location of the full backup file is on your server’s D drive and that you are restoring the file to the database named NewDatabase. The statement moves the data file and the log file from the full backup to new files for my TestDatabase database. The last statement in the script, NORECOVERY, is very crucial. The NORECOVERY mode is one of three available options, which are outlined below.

* **NORECOVERY:** Tells SQL Server that you are not finished restoring the database and that subsequent restore files will occur. While the database is in this state, the database is not yet available, so no connections are allowed.
* **RECOVERY:** Tells SQL Server that you are finished restoring the database, and it is ready to be used. This is the default option, and it is by far the one that is used most often.
* **STANDBY:** Tells SQL Server that the current database is not yet ready to be fully recovered and that subsequent log files can be applied to the restore. You can use this option so that connections are available to the restore database if necessary. However, future transaction logs can only be applied to the database if no current connections exist.

Once you restore the full backup using the NORECOVERY option, you can begin applying the transaction log backups or the differential backup.

**Differential backup**  
A differential backup is a backup of any changes to the database that have occurred since the last full database backup. If you have multiple differential backups, you will only need to restore the very last one taken. In this situation, there are no differential backups, so you can move directly to the transaction log backups.

**Transaction log backups**A transaction log backup keeps track of all transactions that have occurred since the last transaction log backup; it also allows you to restore your database to a point in time before a database error occurred. Transaction log backups occur in sequence, creating a chain. When restoring a sequence of transaction log backups to a point in time, it is required that the transaction log files are restored in order.

When you use a database maintenance plan to create the transaction log backups, a time indicator is typically included in the transaction log file name. The script below applies three transaction log backups using the NORECOVERY option, and the last statement restores the database to availability to the time frame at the very end of the last transaction log file.

RESTORE LOG NewDatabase  
FROM DISK = ''D: \BackupFiles\TestDatabase\_TransactionLogBackup1.trn'  
WITH NORECOVERY

RESTORE LOG NewDatabase  
FROM DISK = ''D: \BackupFiles\ TestDatabase\_TransactionLogBackup2.trn'  
WITH NORECOVERY

RESTORE LOG NewDatabase  
FROM DISK = ''D: \BackupFiles\ TestDatabase\_TransactionLogBackup3.trn'  
WITH NORECOVERY

RESTORE LOG NewDatabase  
FROM DISK = ''D: \BackupFiles\ TestDatabase\_TransactionLogBackup4.trn'  
WITH RECOVERY

**Restoring to a point in time**In the example above, you restore the database to the end of the last transaction log. If you want to recover your database to a specific point in time before the end of the transaction log, you must use the STOPAT option. The script below restores the fourth transaction log in the log sequence to 4:01 PM — just before the database mishap occurred.

RESTORE LOG NewDatabase  
FROM DISK = ''D: \BackupFiles\ TestDatabase\_TransactionLogBackup4.trn'  
WITH STOPAT = N'6/28/2007 4:01:45 PM', RECOVERY

Now that you have the database restore to a point where you need it to be, it is time to decide how to help the developers in order to make their situation a little bit easier. My suggestion is to copy the table the developers need to a separate table on the server so that you or they can correct the data problem.

**To Kill DB2 from backend:**

root[fssdb2p01vu]/home/rkienzyn: ps -fu udb68

     UID      PID     PPID   C    STIME    TTY  TIME CMD

   udb68 11075708        1   0 23:20:00      -  0:00 /home/udb68/sqllib/bin/db2bp 16646668A263 5 A

   udb68 11141150        1   0 23:20:00      -  0:00 db2

   udb68 11993326 13632202   0 23:19:55  pts/4  0:00 -ksh

   udb68 13238502 16515818   0   Oct 17      - 20:00 db2sysc 0

   udb68  8454902 16515818   0   Oct 17      -  0:49 db2acd 0

   udb68  9175688 13632202   0 23:19:55  pts/3  0:00 -ksh

   udb68 12911246 13238502   0   Oct 17      -  0:00 db2vend (PD Vendor Process - 258)

   udb68 13632202 12124816   0 23:19:55      -  0:00 sshd: udb68@pts/3,pts/4

   udb68 13697750        1   0 23:21:04      -  0:00 /home/udb68/sqllib/bin/db2bp 11993326A263 5 A

root[fssdb2p01vu]/home/rkienzyn: kill -9 11075708 11141150 13238502 8454902 12911246 13697750

root[fssdb2p01vu]/home/rkienzyn: ps -fu udb68

     UID      PID     PPID   C    STIME    TTY  TIME CMD

   udb68 11993326 13632202   0 23:19:55  pts/4  0:00 -ksh

   udb68  9175688 13632202   0 23:19:55  pts/3  0:00 -ksh

   udb68 13632202 12124816   0 23:19:55      -  0:00 sshd: udb68@pts/3,pts/4

root[fssdb2p01vu]/home/rkienzyn:

**db2move:**

db2move DB1 export -aw -l lobs -sn schema1

db2move DB2 import -io replace\_create -l lobs

db2set DB2CODEPAGE=1208

db2move emsscdb6 export -aw -l lobs -sn SCHEMANAME

db2move EMdscdb8 import -io replace\_create -l lobs

**Example 2 -** Move all data from tableA and tableB in schema1 from DB1 to database DB2

db2move DB1 export -aw -l lobs -tn schema1.tableA,schema1.tableB

db2move DB2 load -lo replace -l lobs

**Don't forget to set integrity for tables in pending status after the load.**

**Example 3 -**Move all data from tablespaceA in schema1 from DB1 to database DB2

db2move DB1 export -aw -l lobs -ts tablespaceA

db2move DB2 import -io replace\_create -l lobs