Follow along with a hands-on laboratory exercise to learn how to back up and recover in DB2® for z/OS®. This tutorial shows how to create backups of DB2 and how to recover lost data, using DB2 utilities and JCL — an excellent method to show customers how to use the backup and recovery features of DB2 and get them familiar with it.

Introduction

Users frequently come up with questions, asking to explain how certain features and functions of IBM products work. IBM products are installed, but users are new to it and usually have a hard time wrapping their heads around it and have no idea where to start. The best way to explain is having users do it themselves. This guide takes you through a small laboratory exercise to give users the touch and feel of what can be done and how to do it. Of course, exercises like these don't cover every possible detail or every possible parameter that could be used. The purpose is to introduce the product and give an impression of the most commonly used functions. It provides a basis on which to build further.

The exercise was developed for a user new to DB2 for z/OS. The customer had to implement the backup and recovery procedures, but, new as they were, had only a theoretical idea what options DB2 was giving them and, of course, less of how to use and implement them. This exercise basically covers the following utilities of DB2 for z/OS:

- UNLOAD
- LOAD (both RESUME and REPLACE)
- REPAIR
- COPY (both FULL and INCREMENTAL)
- RECOVER (unconditional, TOCOPY and TOLOGPOINT)
- QUIESCE
- REBUILD INDEX
The exercise was developed and executed on DB2 10 for z/OS, but can be used without any modification for DB2 11. All queries were being executed via SPUFI.

The JCL shown in this paper is a sample JCL. Job control cards are not shown. In your environment, DB2 product libraries might have different names, as will input and output data sets, DB2 subsystem, etc.

### Creating a database

#### Creating the DB2 objects

We start with the creation of a simple database with two tables. To focus on the backup and recovery features, these tables have no referential integrity or relation between them. The layout of both the tables is the same. The difference between these tables is that the first has four indices, with one being a UNIQUE index. The second table has only two indices, none being UNIQUE. The size of the columns of both tables have been chosen in such a way that no more than five records will fit into a standard 4K storage page. The creation statements look like the following.

**Listing 1. Create statements for test database**

```sql
--DROP DATABASE LAB01;
--
--COMMIT;
--
CREATE DATABASE LAB01;
--
COMMIT;
--
CREATE TABLESPACE TSLAB01
IN LAB01
PCTFREE 10
LOCKSIZE ROW;
--
CREATE TABLE TBLAB01
( T01_D_PROCES DATE NOT NULL WITH DEFAULT,
  T01_T_PROCES TIME NOT NULL WITH DEFAULT,
  T01_N_AGENCY SMALLINT NOT NULL WITH DEFAULT,
  T01_C_USERID CHAR(4) NOT NULL WITH DEFAULT,
  T01_C_TERMNL CHAR(4) NOT NULL WITH DEFAULT,
  T01_C_CURNCY CHAR(3) NOT NULL WITH DEFAULT,
  T01_C_TRNSAX CHAR(3) NOT NULL WITH DEFAULT,
  T01_N_PERIOD INTEGER NOT NULL WITH DEFAULT,
  T01_N_OLDPCT DEC(6,4) NOT NULL WITH DEFAULT,
  T01_D_OSTART DATE NOT NULL WITH DEFAULT,
  T01_N_NEWPCT DEC(6,4) NOT NULL WITH DEFAULT,
  T01_D_NSTART DATE NOT NULL WITH DEFAULT,
  T01_C_COMENT CHAR(250) NOT NULL WITH DEFAULT,
  T01_C_TEST01 CHAR(250) NOT NULL WITH DEFAULT,
  T01_C_TEST02 CHAR(250) NOT NULL WITH DEFAULT,
  T01_T_TSTAMP TIMESTAMP NOT NULL WITH DEFAULT )
IN LAB01.TSLAB01;
--
CREATE UNIQUE INDEX TBLAB01_K00
ON TBLAB01
( T01_D_PROCES ASC,
  T01_T_PROCES ASC,
  T01_N_AGENCY ASC,
  T01_C_USERID ASC )
CLUSTER
PCTFREE 10;
```

"A how-to guide: Backup and recovery in DB2 for z/OS"
CREATE TABLESPACE TSLAB02
IN LAB01
PCTFREE 10
LOCKSIZE ROW;

CREATE TABLE TBLAB02
( T02_D_PROCES      DATE      NOT NULL WITH DEFAULT,
  T02_T_PROCES      TIME      NOT NULL WITH DEFAULT,
  T02_N_AGENCY      SMALLINT  NOT NULL WITH DEFAULT,
  T02_C_USERID      CHAR(4)   NOT NULL WITH DEFAULT,
  T02_C_TERMNL      CHAR(4)   NOT NULL WITH DEFAULT,
  T02_C_CURNCY      CHAR(3)   NOT NULL WITH DEFAULT,
  T02_C_TRNSAX      CHAR(3)   NOT NULL WITH DEFAULT,
  T02_N_PERIOD      INTEGER   NOT NULL WITH DEFAULT,
  T02_N_OLDPCT      DEC(6,4)  NOT NULL WITH DEFAULT,
  T02_D_OSTART      DATE      NOT NULL WITH DEFAULT,
  T02_N_NEWPCT      DEC(6,4)  NOT NULL WITH DEFAULT,
  T02_D_NSTART      DATE      NOT NULL WITH DEFAULT,
  T01_C_COMENT      CHAR(250) NOT NULL WITH DEFAULT,
  T01_C_TEXT01      CHAR(250) NOT NULL WITH DEFAULT,
  T01_C_TEST02      CHAR(250) NOT NULL WITH DEFAULT,
  T02_T_TSTAMP      TIMESTAMP NOT NULL WITH DEFAULT )
IN LAB01.TSLAB02;

CREATE INDEX TBLAB02_K01
ON TBLAB02
( T02_C_USERID      ASC,
  T02_D_PROCES      ASC,
  T02_T_PROCES      ASC )
PCTFREE 10;

CREATE INDEX TBLAB02_K02
ON TBLAB02
( T02_N_AGENCY      ASC,
  T02_D_PROCES      ASC,
  T02_T_PROCES      ASC )
PCTFREE 10;

CREATE INDEX TBLAB02_K03
ON TBLAB02
( T02_C_CURNCY      ASC,
  T02_N_PERIOD      ASC )
PCTFREE 10;

All statements should finish with an SQLCODE 0.

**Inserting data into the test database**

When we have our database created, we load into each table an (identical) set of 25 records via standard *INSERT* statements, as follows:
Listing 2. INSERT statements for test data

```
DELETE FROM TBLA01;
--
COMMIT;
--
INSERT INTO TBLA01 VALUES
( '2008-09-30', '10.03.03', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 0.0000
, '1900-01-01', 2.0000, '2008-09-30', 'RECORD NR 01', 'TEXT1 01'
, 'TEXT2 01', CURRENT TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2008-09-30', '10.04.04', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.0000
, '1900-01-01', 1.1250, '2008-09-30', 'RECORD NR 02', 'TEXT1 02'
, 'TEXT2 02', CURRENT TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2008-09-30', '10.05.05', 0, 'LAMN', 'B066', 'USD', 'TAR', 180, 0.0000
, '1900-01-01', 3.0000, '2008-09-30', 'RECORD NR 03', 'TEXT1 03'
, 'TEXT2 03', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2008-09-30', '10.06.06', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 1.1250
, '2008-09-30', 0.8500, '2008-09-30', 'RECORD NR 04', 'TEXT1 04'
, 'TEXT2 04', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2009-09-15', '08.00.00', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 2.0000
, '2008-09-30', 0.8500, '2009-09-15', 'RECORD NR 05', 'TEXT1 05'
, 'TEXT2 05', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2009-09-15', '08.01.01', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 2.0000
, '2008-09-30', 0.8500, '2009-09-15', 'RECORD NR 06', 'TEXT1 06'
, 'TEXT2 06', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2009-09-15', '08.02.02', 0, 'LAMN', 'B066', 'USD', 'TAR', 180, 4.5000
, '2008-09-30', 1.1500, '2009-09-15', 'RECORD NR 07', 'TEXT1 07'
, 'TEXT2 07', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2009-10-22', '16.26.26', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.8500
, '2009-09-15', 0.5000, '2009-10-22', 'RECORD NR 08', 'TEXT1 08'
, 'TEXT2 08', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2010-07-15', '21.53.53', 0, 'LAMN', 'SO53', 'USD', 'TAR', 90, 0.8500
, '2009-09-15', 1.0000, '2010-07-15', 'RECORD NR 09', 'TEXT1 09'
, 'TEXT2 09', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2010-07-15', '21.54.54', 0, 'LAMN', 'SO53', 'USD', 'TAR', 360, 1.1500
, '2009-09-15', 1.3000, '2010-07-15', 'RECORD NR 10', 'TEXT1 10'
, 'TEXT2 10', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2010-07-31', '18.04.04', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.5000
, '2009-10-22', 0.3000, '2010-07-31', 'RECORD NR 11', 'TEXT1 11'
, 'TEXT2 11', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2011-04-11', '07.55.55', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 1.0000
, '2010-07-15', 1.5000, '2011-04-11', 'RECORD NR 12', 'TEXT1 12'
, 'TEXT2 12', CURRENT_TIMESTAMP );
--
INSERT INTO TBLA01 VALUES
( '2011-04-11', '07.56.56', 0, 'LAMN', 'B066', 'USD', 'TAR', 360, 1.3000
, 'TEXT2 13', CURRENT_TIMESTAMP );
--
```
CREATE TABLE TBLAB01
    ( '
    '2011-04-30', '19.48.48', 0, 'LAMN', 'B065', 'USD', 'TAR', 0,0.3000
    , '2010-07-31', 0.4000, '2011-04-30', 'RECORD NR 13', 'TEXT1 13'
    , 'TEXT2 13',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
    ( '2012-09-24', '07.43.43', 0, 'LAMN', 'B065', 'USD', 'TAR', 90,1.5000
    , '2011-04-11', 1.1500, '2012-09-24', 'RECORD NR 14', 'TEXT1 14'
    , 'TEXT2 14',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
    ( '2012-09-24', '07.44.44', 0, 'LAMN', 'B065', 'USD', 'TAR',180,1.6000
    , '2011-04-11', 1.3500, '2012-09-24', 'RECORD NR 15', 'TEXT1 15'
    , 'TEXT2 15',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
    ( '2012-10-31','18.30.30',0,'LAMN','B065','USD','TAR',90,1.1500
    , '2012-09-24',0.9000,'2012-10-31','RECORD NR 16','TEXT1 16'
    , 'TEXT2 16',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
    ( '2012-10-31','18.31.31',0,'LAMN','B065','USD','TAR',180,1.3500
    , '2012-09-24',0.7000,'2012-10-31','RECORD NR 17','TEXT1 17'
    , 'TEXT2 17',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
    ( '2012-09-24','17.43.43',0,'LAMN','B065','USD','TAR',90,1.5000
    , '2011-04-11',1.1500,'2012-09-24','RECORD NR 18','TEXT1 18'
    , 'TEXT2 18',CURRENT_TIMESTAMP );
--
COMMlT;
--
DELETE FROM TBLAB02;
-- COMMIT;
--
INSERT INTO TBLAB02 VALUES
( '2008-09-30', '10.03.03', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 0.0000,
 '1900-01-01', 2.0000, '2008-09-30', 'RECORD NR 01', 'TEXT1 01',
 'TEXT2 01', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2008-09-30', '10.04.04', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.0000,
 '1900-01-01', 1.1250, '2008-09-30', 'RECORD NR 02', 'TEXT1 02',
 'TEXT2 02', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2008-09-30', '10.05.05', 0, 'LAMN', 'B066', 'USD', 'TAR', 180, 0.0000,
 '1900-01-01', 3.0000, '2008-09-30', 'RECORD NR 03', 'TEXT1 03',
 'TEXT2 03', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2009-09-15', '08.00.00', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 1.1250,
 '2008-09-30', 0.8500, '2009-09-15', 'RECORD NR 04', 'TEXT1 04',
 'TEXT2 04', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2009-09-15', '08.01.01', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 2.0000,
 '2009-09-30', 0.8500, '2009-09-15', 'RECORD NR 05', 'TEXT1 05',
 'TEXT2 05', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2009-09-15', '08.02.02', 0, 'LAMN', 'B066', 'USD', 'TAR', 360, 4.5000,
 '2009-09-30', 1.1500, '2009-09-15', 'RECORD NR 06', 'TEXT1 06',
 'TEXT2 06', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2009-10-22', '16.26.26', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.8500,
 '2009-09-15', 0.5000, '2009-10-22', 'RECORD NR 07', 'TEXT1 07',
 'TEXT2 07', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2010-07-15', '21.53.53', 0, 'LAMN', 'SO53', 'USD', 'TAR', 90, 0.8500,
 '2009-09-15', 1.0000, '2010-07-15', 'RECORD NR 08', 'TEXT1 08',
 'TEXT2 08', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2010-07-15', '21.54.54', 0, 'LAMN', 'SO53', 'USD', 'TAR', 360, 1.1500,
 '2009-09-15', 1.3000, '2010-07-15', 'RECORD NR 09', 'TEXT1 09',
 'TEXT2 09', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2010-07-31', '18.04.04', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.5000,
 '2009-10-22', 0.3000, '2010-07-31', 'RECORD NR 10', 'TEXT1 10',
 'TEXT2 10', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2011-04-11', '07.55.55', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 1.0000,
 '2010-07-15', 1.5000, '2011-04-11', 'RECORD NR 11', 'TEXT1 11',
 'TEXT2 11', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2011-04-11', '07.56.56', 0, 'LAMN', 'B066', 'USD', 'TAR', 360, 1.3000,
 'TEXT2 12', CURRENT_TIMESTAMP);
--
INSERT INTO TBLAB02 VALUES
( '2011-04-30', '19.40.40', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.3000,
 '2010-07-31', 0.4000, '2011-04-30', 'RECORD NR 13', 'TEXT1 13',
 'TEXT2 13', CURRENT_TIMESTAMP);
All statements should finish with an SQLCODE 0 or 100.

**Check the contents of the tables**

Once the data is inserted, we execute the following query to check.

```sql
-- INSERT INTO TBLAB02 VALUES
( '2012-09-24', '07.43.43', 0, 'LAMN', 'B065', 'USD', 'TAR', 90, 1.5000,
  '2011-04-11', 1.1500, '2012-09-24', 'RECORD NR 14', 'TEXT1 14',
  'TEXT2 14', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-09-24', '07.44.44', 0, 'LAMN', 'B065', 'USD', 'TAR', 180, 1.6000,
  '2011-04-11', 1.3500, '2012-09-24', 'RECORD NR 15', 'TEXT1 15',
  'TEXT2 15', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-10-31', '18.30.30', 0, 'LAMN', 'B065', 'USD', 'TAR', 90, 0.9000,
  '2012-09-24', 0.9000, '2012-10-31', 'RECORD NR 16', 'TEXT1 16',
  'TEXT2 16', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-10-31', '18.31.31', 0, 'LAMN', 'B065', 'USD', 'TAR', 180, 1.3500,
  '2012-09-24', 0.8000, '2012-10-31', 'RECORD NR 17', 'TEXT1 17',
  'TEXT2 17', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-10-31', '18.32.32', 0, 'LAMN', 'B065', 'USD', 'TAR', 360, 1.4000,
  '2012-09-24', 0.7000, '2012-10-31', 'RECORD NR 18', 'TEXT1 18',
  'TEXT2 18', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-11-06', '18.19.19', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 0.9000,
  '2012-10-31', 0.9000, '2012-11-06', 'RECORD NR 19', 'TEXT1 19',
  'TEXT2 19', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-11-06', '18.20.20', 0, 'LAMN', 'B066', 'USD', 'TAR', 360, 0.7000,
  '2012-10-31', 1.4000, '2012-11-06', 'RECORD NR 20', 'TEXT1 20',
  'TEXT2 20', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-11-06', '18.16.16', 0, 'LAMN', 'B066', 'USD', 'TAR', 90, 1.1500,
  '2011-04-30', 1.1500, '2012-11-06', 'RECORD NR 21', 'TEXT1 21',
  'TEXT2 21', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2012-11-06', '18.17.17', 0, 'LAMN', 'B066', 'USD', 'TAR', 360, 1.4000,
  '2012-11-06', 0.7000, '2012-11-12', 'RECORD NR 22', 'TEXT1 22',
  'TEXT2 22', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2013-01-11', '14.41.41', 0, 'LAMN', 'B066', 'USD', 'TAR', 0, 0.4000,
  '2011-04-30', 0.2000, '2013-01-11', 'RECORD NR 23', 'TEXT1 23',
  'TEXT2 23', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2013-09-25', '11.55.55', 0, 'MLLU', 'V107', 'USD', 'TAR', 180, 0.8000,
  '2012-11-12', 8.0000, '2013-09-25', 'RECORD NR 24', 'TEXT1 24',
  'TEXT2 24', CURRENT_TIMESTAMP );
-- INSERT INTO TBLAB02 VALUES
( '2013-10-15', '15.57.57', 0, 'FAAN', 'V108', 'EUR', 'TAR', 360, 0.0000,
  '1900-01-01', 0.7800, '2013-10-15', 'RECORD NR 25', 'TEXT1 25',
  'TEXT2 25', CURRENT_TIMESTAMP );
```
### Listing 3. Select statements to check test data

```sql
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES,
         T01_T_PROCES,
         T01_N_AGENCY,
         T01_C_USERID;
```

```sql
--
SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES,
         T02_T_PROCES,
         T02_N_AGENCY,
         T02_C_USERID;
```

Both tables should show 25 records.

### Unloading and loading data

#### Unload both table spaces

Now let's go through the use of the `UNLOAD` utility first. We run the following job:

### Listing 4. Sample JCL to run UNLOAD utility

```plaintext
//**************************************************************
//** STEP DELETE: DELETE OLD DATA SETS
//**************************************************************
//@
//DELETE  EXEC PGM=IEFBR14
//CNTL01   DD DSN=IBMWG.CNTL.TSLAB01,
//         UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//UNLD01   DD DSN=IBMWG.UNLD.TSLAB01,
//         UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//CNTL02   DD DSN=IBMWG.CNTL.TSLAB02,
//         UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//UNLD02   DD DSN=IBMWG.UNLD.TSLAB02,
//         UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//@
//**************************************************************
//** STEP UNLD1: UNLOAD TABLE SPACE
//**************************************************************
//@
//UNLD1   EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.UNLD01'
//SYSPUNCH DD DSN=IBMWG.CNTL.TSLAB01,
//         SPACE=(TRK,(5,5),RLSE),
//         UNIT=SYSDA,
//         DISP=(,CATLG,DELETE)
//SYSSREC DD DSN=IBMWG.UNLD.TSLAB01,
//         DISP=(,CATLG,DELETE),
//         DCB=(BLKSIZE=8192),
//         SPACE=(TRK,(35,5),RLSE),
//         UNIT=SYSDA
//SYسين DD *
UNLOAD TABLESPACE LAB01.TSLAB01
FROM TABLE "IBMWG"."TBLAB01"
//@
//@
//**************************************************************
//** STEP UNLD2: UNLOAD TABLE SPACE
//**************************************************************
//@
```
Note: The first step of this job deletes the unload data sets that are going to be created, in case they still exist from a previous execution. In an actual scenario where you work with real data, you would have to be careful while deleting unload data sets because you may need them later.

The execution of this job should end with RC=00. Once finished, we have the user look at the following items:

- The job output under SDSF — In the SYSPRINT, information is shown about the utility execution.
- The SYSREC files — They each contain 25 records with the contents of the table space in tabular format.
- The SYSPUNCH files — These contain the layout of the unloaded data and the necessary statements to be used for reloading the data.

It is important that the user understands what the contents of each file are, what they are used for, and what is the difference between the two.

**Load both table spaces with RESUME YES**

We now execute the **LOAD** utility for both table spaces, using the SYSPUNCH and SYSREC files that were the result of our previous step. We do not change anything on the **LOAD** statement, which means the **LOAD** utility will execute as below.

**Listing 5. Sample **LOAD** card to run **LOAD** (RESUME) utility**

```
LOAD DATA INDDN SYSREC  LOG NO  RESUME YES
```

The following job is executed.

**Listing 6. Sample JCL to run **LOAD** (RESUME) utility**

```
/*
************************************************************************************************************
/* STEP LOAD1: LOAD TABLE SPACE
************************************************************************************************************
/*
//LOAD1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.LOAD01'
//DSNUPROC.SYSREC DD DSN=IBMWG.UNLD.TSLAB01,DISP=(OLD),UNIT=SYSDA
//DSNUPROC.SYSUT1 DD DSN=&SYSUT1,DISP=(NEW,PASS),UNIT=SYSDA,
//     SPACE=(TRK,(16,16))
```
This job should end with an RC=04. Once finished, we have the user look at the following items:

- The job output under SDSF. In the SYSPRINT of both utility executions, important information is shown. Why the execution gives an RC=04: both table spaces are left in status COPY PENDING. Apart from that, due to the UNIQUE index we created there, in the first table space, all records are being rejected because of a duplicate key condition (the other table has no UNIQUE index, so all records are being loaded).
- The status of the table and index spaces, using DB2 commands under Interactive DB2. This will show both table spaces in status RW,COPY.

The command to be used:

```
-DISPLAY DATABASE(LAB01) SPACENAM(*)
```

The output of this command shows something similar to the following.

**Listing 7. Result of the DB2 command to show the status of the table spaces within a database**

```
DSNT360I -DSN1 ***********************
DSNT361I -DSN1 * DISPLAY DATABASE SUMMARY
   * GLOBAL
DSNT360I -DSN1 ***********************
DSNT362I -DSN1 DATABASE = LAB01 STATUS = Rw
    DBD LENGTH = 8066
DSNT397I -DSN1
NAME   TYPE PART  STATUS       PHYERRLO PHYERRHI CATALOG  PIECE
------- ---- ----- ----------------- -------- -------- -------- -----
TSLAB01 TS         Rw, COPY
TSLAB02 TS         Rw, COPY
TBLA1F1L IX         Rw
TBLA1TC1 IX         Rw
TBLA1VD1 IX         Rw
TBLA1V0E IX         Rw
TBLA081R IX         Rw
TBLA082R IX         Rw
****** DISPLAY OF DATABASE LAB01 ENDED ***********************
DSN9022I -DSN1 DSNTDIS 'DISPLAY DATABASE' NORMAL COMPLETION
***
```
The COPY PENDING status is the result of the \texttt{LOG NO} condition on the \texttt{LOAD} utility. Since DB2 didn't log the changes to the table space, it instead directs the user to create a backup to cover the missing log information.

In this situation, there are two solutions:

- Take an image copy of both table spaces
- Run the \texttt{REPAIR} utility to remove the COPY PENDING status

It is important to explain which solution is used under which circumstance. Usually, when you have a simple way to re-create the contents of a table space after a \texttt{LOAD}, there is no need to take an image copy. The image copy is taken in case you definitely need a backup and if that is your only way to recover the contents of the table space.

**Check the contents of the tables**

Once the data is loaded, we execute the following query to check.

**Listing 8. Select statements to check test data**

```sql
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES,
         T01_T_PROCES,
         T01_N_AGENCY,
         T01_C_USERID;
```

```sql
SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES,
         T02_T_PROCES,
         T02_N_AGENCY,
         T02_C_USERID;
```

The first table should still show 25 records. The second table should now show 50 records.

**Remove COPY PENDING status**

In this step, we run the \texttt{REPAIR} utility to remove the COPY PENDING status of the two table spaces.

**Listing 9. Sample JCL to run REPAIR utility**

```plaintext
//*
//******************************************************************************
//* STEP REPAIR1: REPAIR TABLE SPACES
//******************************************************************************
//*
//REPAIR1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.REPAIR'
//SYIN DD *
REPAIR OBJECT
SET TABLESPACE LAB01.TSLAB01 NOCOPYPEND
SET TABLESPACE LAB01.TSLAB02 NOCOPYPEND
/*
```

The execution of this job should give a RC=00. When finished, look at the following items:
• The job output under SDSF — In the SYSPRINT of the utility execution, if all important information is shown.
• The status of the table and index spaces, using DB2 commands under Interactive DB2 — They will now show that both table spaces are in RW status.

**Load both table spaces with REPLACE**

Now we are going to load both table spaces with a `LOAD REPLACE` to show the difference between that and `LOAD RESUME`. For illustration purposes, we have already prepared two different SYSNUCH files with the following `LOAD` instructions:

```
LOAD DATA INDDN SYSREC   LOG NO  RESUME NO REPLACE NOCOPYPEND
```

Please note that we not only changed the load instruction to `RESUME NO REPLACE`, but also added `NOCOPYPEND`. It is important to explain the reason for this parameter (referring to the COPY PENDING status that resulted from our `LOAD RESUME` step). We could also have changed the `LOAD` card to `LOG YES` instead.

We also created two new SYSREC files, which are basically copies from the original files, where we changed the dates on some of the records.

**Listing 10. Sample of changed load data to be used in LOAD (REPLACE) utility**

```
..2008-09-3010.03.03..LAMNB066USDTAR...!....1900-01-01....2008-09-30RECORD NR 01
..2008-09-3010.04.04..LAMNB066USDTAR.........1900-01-01....2008-09-30RECORD NR 02
..2008-09-3010.05.05..LAMNB066USDTAR...©....1900-01-01....2008-09-30RECORD NR 03
..2009-09-1508.00.00..LAMNB066USDTAR........2008-09-30..&.2009-09-15RECORD NR 04
..2009-09-1508.01.01..LAMNB066USDTAR...!....2008-09-30..&.2009-09-15RECORD NR 05
..2009-09-1508.02.02..LAMNB066USDTAR...Ç....2008-09-30..&.2009-09-15RECORD NR 06
..2009-10-2216.26.26..LAMNB066USDTAR......&.2009-09-15....2009-10-22RECORD NR 07
..2012-07-3118.28.28..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 08
..2012-07-3118.29.29..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 09
..2012-07-3118.30.30..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 10
..2012-07-3118.31.31..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 11
..2012-07-3118.32.32..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 12
..2012-07-3118.33.33..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 13
..2012-07-3118.34.34..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 14
..2012-07-3118.35.35..LAMNB066USDTAR........2009-10-22....2012-07-31RECORD NR 15
```

Then we run the following job.

**Listing 11. Sample JCL to run LOAD (REPLACE) utility**

```
//**********************************************************************
//* STEP LOAD1: LOAD TABLE SPACE
//**********************************************************************
```
//LOAD1   EXEC DSNUPROC,SYSTEM=DSN1,UID='IBM.W LOAD01'
//DSNUPROC.SYSREC DD DSN=IBM.W.UNLD.TS LAB01A,DISP=(OLD),UNIT=SYSDA
//DSNUPROC.SYSUT1 DD DSN=&SYSUT1,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SORTOUT DD DSN=&SORTOUT,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SYSMAP DD DSN=&SYSMAP,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SYSIN DD DSN=IBM.W.CNTL.TS LAB01A,DISP=(OLD),UNIT=SYSDA
//*
//**********************************************************************
//* STEP LOAD2: LOAD TABLE SPACE
//**********************************************************************
//*
//LOAD2   EXEC DSNUPROC,SYSTEM=DSN1,UID='IBM.W LOAD02'
//DSNUPROC.SYSREC DD DSN=IBM.W.UNLD.TS LAB02A,DISP=(OLD),UNIT=SYSDA
//DSNUPROC.SYSUT1 DD DSN=&SYSUT1,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SORTOUT DD DSN=&SORTOUT,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SYSMAP DD DSN=&SYSMAP,DISP=(NEW, PASS),UNIT=SYSDA,
//                  SPACE=(TRK,(10,10))
//DSNUPROC.SYSIN DD DSN=IBM.W.CNTL.TS LAB02A,DISP=(OLD),UNIT=SYSDA

The execution of this job ends with an RC=00. After running the job, we look at the following items:

- The job output under SDSF — In the SYSPRINT of the utility execution, if all important information is shown.
- The status of the table and index spaces, using DB2 commands under Interactive DB2 — They will show that both table spaces are in RW status.

**Review the contents of the tables**

With the **LOAD REPLACE** finished, we again select the contents of both tables in order to check.

**Listing 12. Select statements to check test data**

```
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES
  , T01_T_PROCES
  , T01_N_AGENCY
  , T01_C_USERID;
```  
```
SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES
  , T02_T_PROCES
  , T02_N_AGENCY
  , T02_C_USERID;
```

Both tables should show 25 records again, but in both cases with a new set of data, in accordance with the (changed) SYSREC files we used for our **LOAD REPLACE**.

**Creating image copies and recovering data**

**Create image copies of both table spaces**

With the new contents of both tables in place, we are now creating a (full) image copy of both table spaces.
Listing 13. Sample JCL to run COPY (FULL) utility

```java
//*
//**********************************************************************
//* STEP DELETE: DELETE OLD DATA SETS
//**********************************************************************
//*
//DELETE EXEC PGM=IEFBR14
//COPY01 DD DSN=IBMwG.COPY.TSLAB01.FULL01,
//UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//COPY02 DD DSN=IBMwG.COPY.TSLAB02.FULL01,
//UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//*
//**********************************************************************
//* STEP COPY1: FULL IMAGE COPY OF TABLE SPACE
//**********************************************************************
//*
//COPY1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMwG.COPYFL'
//SYSCOPY DD DSN=IBMwG.COPY.TSLAB01.FULL01,UNIT=SYSDA,
//SPACE=(CYL,(10,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE LAB01.TSLAB01
/*
//*
//**********************************************************************
//* STEP COPY2: FULL IMAGE COPY OF TABLE SPACE
//**********************************************************************
//*
//COPY2 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMwG.COPYFL'
//SYSCOPY DD DSN=IBMwG.COPY.TSLAB02.FULL01,UNIT=SYSDA,
//SPACE=(CYL,(10,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE LAB01.TSLAB02
/*
```

Note: The first step of this job deletes the image copy data sets that are going to be created, in case they still exist from a previous execution. In a live situation, you shouldn't delete any image copy data sets, unless you are sure you would not need them anymore. Be aware that image copy data sets cannot be reused because the catalog of DB2 retains information about the image copy activities and the data sets used. If you decide to delete image copy data sets, it is important to clean up the catalog as well, using the MODIFY RECOVERY utility, which is not covered in this exercise.

The execution of this job terminates with an RC=00. When finished, look at the following items:

- The output of the job under SDSF — Especially the SYSPRINTs of both COPY utilities contain valuable information.
- The output data sets — It is important to show the difference between the SYSRECs from the UNLOAD utility (which contain the data from the table space in tabular format) and the SYSCOPYs from the COPY utility (which contain the data from the table space in page format). The image copy data sets show two interesting things:
  1. Both table spaces are occupying seven data pages, each with four records (the last page only one record).
  2. The records were stored on the data pages of the table space in the same order they were loaded by the LOAD utility.
Review the information in the catalog of DB2
With the full image copies taken, we now select data from the catalog table SYSCOPY to show what kind of information is being stored in the catalog of DB2 regarding backup and recovery activities. We use the following query:

```sql
SELECT * FROM SYSIBM.SYSCOPY
WHERE DBNAME = 'LAB01';
```

The result is the following (only showing the first columns).

**Listing 14. Output of catalog data**

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>ICTYPE</th>
<th>ICDATE</th>
<th>START_RBA</th>
<th>FILESEQNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3©»</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§§E²</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§³©ç</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§Ú²J</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.É,</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.ÓH</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ü.</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.Wo</td>
<td>0</td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 8

It is important to explain the meaning of the columns ICTYPE and START_RBA:

- **ICTYPE** indicates the type of backup and recovery activity (C = create, Y = load resume, S = load replace, F = full image copy, etc.).
- **START_RBA** (RBA = Relative Byte Address) indicates the point in time when the activity took place; the indication is expressed as a pointer to the log.

**Insert additional records into both tables**
We now insert five additional records in each of our test tables, using standard **INSERT** statements.

```sql
INSERT INTO TBLAB01 VALUES
('2010-09-30','10.03.03',0,'LAMN','B066','USD','TAR', 90,0.0000
,'1900-01-01',2.0000,'2010-09-30','RECORD NR 26','TEXT1 26'
,'TEXT2 26',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
('2010-09-30','10.04.04',0,'LAMN','B066','USD','TAR',  0,0.0000
,'1900-01-01',1.1250,'2010-09-30','RECORD NR 27','TEXT1 27'
,'TEXT2 27',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
('2010-09-30','10.05.05',0,'LAMN','B066','USD','TAR',180,0.0000
,'1900-01-01',3.0000,'2010-09-30','RECORD NR 28','TEXT1 28'
,'TEXT2 28',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
('2011-09-15','08.00.00',0,'LAMN','B066','USD','TAR',  0,1.1250
,'2010-09-30',0.8500,'2011-09-15','RECORD NR 29','TEXT1 29'
,'TEXT2 29',CURRENT_TIMESTAMP );
--
INSERT INTO TBLAB01 VALUES
('2011-09-15','08.01.01',0,'LAMN','B066','USD','TAR', 90,2.0000
,'2010-09-30',0.8500,'2011-09-15','RECORD NR 30','TEXT1 30'
,'TEXT2 30',CURRENT_TIMESTAMP );
--
```
All inserts should give an SQLCODE 0.

Check the contents of the tables

With these records inserted, we again select the contents of both tables in order to check.

Listing 15. Select statements to check test data

```
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES
 , T01_T_PROCES
 , T01_N_AGENCY
 , T01_C_USERID;

SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES
 , T02_T_PROCES
 , T02_N_AGENCY
 , T02_C_USERID;
```

Both tables should now show 30 records.

Create incremental image copy

We then create an incremental image copy of the second table space, using the following job.
Listing 16. Sample JCL to run COPY (INCREMENTAL) utility

```plaintext
//*
//#**********************************************************************
//#* STEP DELETE: DELETE OLD DATA SETS
//#**********************************************************************
//*
//DELETE EXEC PGM=IEFBR14
//COPY01 DD DSN=IBMwG.COPY.TSLAB02.INCR01,
//   UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//*
//#**********************************************************************
//#* STEP COPY1: INCREMENTAL IMAGE COPY OF TABLE SPACE
//#**********************************************************************
//*
//COPY1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMwG.COPYIN'
//SYSCOPY DD DSN=IBMwG.COPY.TSLAB02.INCR01,UNIT=SYSDA,
//   SPACE=(CYL,(1,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE LAB01.TSLAB02 FULL NO
/*
```

Note: The first step of this job deletes the image copy data set to be created, in case it still exists from a previous execution. In a real scenario, you should not delete any image copy data sets, unless you are sure you would not need them anymore.

The execution of the job should end with an RC=00. Look at the following items:

- The job output under SDSF, especially the SYSPRINT of the utility.
- The data set that resulted from the utility execution. As an image copy data set shows the data in page format, an incremental image copy should contain only those pages changed since the last image copy. In our exercise, we see the following interesting information:
  - That the INSERT statement has been placing the five additional records on existing data pages, filling the free space that was initially kept open by the LOAD utility, due to the PCTFREE parameter.
  - Since only five additional records were inserted, only five data pages were changed, so the incremental image copy contains only five data pages. The two data pages that have not changed are not present in the incremental image copy data set.

Re-review the information in the catalog of DB2

With the incremental image copy taken, we again select data from the catalog table SYSCOPY to show what kind of information is being stored in the catalog of DB2 regarding backup and recovery activities.

```
SELECT * FROM SYSIBM.SYSCOPY
WHERE DBNAME = 'LAB01';
```

The result is the following (only showing the first columns).
Listing 17. Output of catalog data

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>ICTYPE</th>
<th>ICDATE</th>
<th>START_RBA</th>
<th>FILESEQNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3©»</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§5퐅</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§7©¢</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§5每日经济</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.É,</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.ÓH</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ü.</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.Wo</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>I</td>
<td>150414</td>
<td>..¶.f©</td>
<td>0</td>
</tr>
</tbody>
</table>
| DSNE610I NUMBER OF ROWS DISPLAYED IS 9

The catalog table now shows the information of the new incremental image copy for table space TSLAB02.

Recover both table spaces without any parameters

Here, we execute a recovery of both table spaces without any parameters. We use the following JCL.

Listing 18. Sample JCL to run RECOVER (FULL) utility

```*/
//***************************************************************
//** STEP RECOV1: RECOVER TABLE SPACES
//***************************************************************
//**
//RECOV1  EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.RECOVER'
//SYSPRINT DD *
RECOVER TABLESPACE LAB01.TSLAB01
RECOVER TABLESPACE LAB01.TSLAB02
/*
```

The execution gives an RC=00. When the job has finished, look at the output in SDSF. The SYSPRINT contains important information.

Check the contents of the tables

After our recovery, we again select the contents of both tables.

Listing 19. Select statements to check test data

```SELECT *
  FROM TBLAB01
  ORDER BY T01_D_PROCES
    , T01_T_PROCES
    , T01_N_AGENCY
    , T01_C_USERID;
--
SELECT *
  FROM TBLAB02
  ORDER BY T02_D_PROCES
    , T02_T_PROCES
    , T02_N_AGENCY
    , T02_C_USERID;
```
Both tables contain 30 records, as before. Although for the first table space, the last (full) image copy has been taken before we inserted five additional records, these additional records have been recovered by the RECOVER utility anyway. If we run the RECOVER utility without parameters, DB2 will recover the table space up until the last available information in the log. It recovers the last available image copy (full and/or incremental) and from there on reads the log to recover the rest of the information. The RECOVER utility uses the value in the column START_RBA to find the point where to start reading the log.

**Recover the second table space to the last full image copy**

In case of the second table space (TSLAB02), we do have a full image copy and an incremental image copy. We will run a recovery to the last full image copy that was taken.

**Listing 20. Sample JCL to run RECOVER (TOCOPY) utility**

```plaintext
//**
//** *******************************************************************************
//**  STEP RECOV1: RECOVER TO LAST FULL COPY OF TABLE SPACE
//** *******************************************************************************
//**
REC0V1  EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.RECOVR'
//SYSPRINT DD *
RECOVER TABLESPACE LAB01.TSLAB02
TOCOPY IBMWG.COPY.TSLAB02.FULL01
/**
```

The execution of this job returns a RC=04. Look at two things:

- The job output in SDSF. Here we discover why the job returned a code 04. In case of a RECOVER TOCOPY, the indices are not being recovered. The index spaces of both indices are left in REBUILD PENDING status.
- We check the status of the table and index spaces via a DB2 command, as before.

The command to use:

```plaintext
-DISPLAY DATABASE(LAB01) SPACENAM(*)
```

The output of this command shows something similar to the following code.

**Listing 21. Result of the DB2 command to show the status of the table spaces within a database**

```plaintext
DSNT360I -DSN1 *******************************************************************************
DSNT361I -DSN1  * DISPLAY DATABASE SUMMARY
  * GLOBAL
DSNT360I -DSN1 *******************************************************************************
DSNT362I -DSN1  DATABASE = LAB01  STATUS = RW
  DBD LENGTH = 8066
DSNT371I -DSN1  **************************************************************************
NAME    TYPE  PART  STATUS  PHYERRLO  PHYERRHI  CATALOG  PIECE
-------- ---- ---- -------- -------- -------- -------- -----
TSLAB01  TS     RW
TSLAB02  TS     RW
TBLA1F1L IX     RW,RBDP
```
Rebuilding the indices

In order to re-create the indices, we need to run the `REBUILD INDEX` utility with the following JCL.

**Listing 22. Sample JCL to run REBUILD INDEX utility**

```plaintext
//**
//**************************************************
//** STEP REBLD1: REBUILD INDEXES
//**************************************************
//**
//REBLD1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.RBLDIX'
//SYIN DD *
REBUILD INDEX (ALL) TABLESPACE LAB01.TSLAB02
/**
```

The execution of this job gives an RC=00. When the job has finished, we look at the following items:

- The output of the job in SDSF — The SYSPRINT gives important information about the execution of the utility.
- We check the status of the table and index spaces in the same way as we did in the previous step. We see now that both index spaces are in RW status.

Check the contents of the tables

After this `RECOVER TOCOPY` and `REBUILD INDEX`, we again select the contents of both tables.

**Listing 23. Select statements to check test data**

```plaintext
SELECT * 
FROM TBLAB01
ORDER BY T01_D_PROCES
   , T01_T_PROCES
   , T01_N_AGENCY
   , T01_C_USERID;
--
SELECT * 
FROM TBLAB02
ORDER BY T02_D_PROCES
   , T02_T_PROCES
   , T02_N_AGENCY
   , T02_C_USERID;
```

The first table (of course) still contains 30 records, but the second table now contains 25 records. The last full image copy was taken before we inserted the five additional records. Now we recovered the table space to the situation before we did the insert.
New full image copy of first table space

We now take a new full image copy of our first table space by running the following job.

**Listing 24. Sample JCL to run COPY (FULL) utility**

```plaintext
//*
//**********************************************************************
//* STEP DELETE: DELETE OLD DATA SETS
//**********************************************************************
//*
//DELETE EXEC PGM=IEFBR14
//COPY01 DD DSN=IBMWG.COPY.TSLAB01.FULL02,
//      UNIT=SYSDA,DISP=(MOD,DELETE,DELETE),SPACE=(TRK,1)
//*
//**********************************************************************
//* STEP COPY1: FULL IMAGE COPY OF TABLE SPACE
//**********************************************************************
//*
//COPY1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.COPYFL'
//SYSCOPY DD DSN=IBMWG.COPY.TSLAB01.FULL02,UNIT=SYSDA,
//      SPACE=(CYL,(10,1)),DISP=(NEW,CATLG,CATLG)
//SYSIN DD *
COPY TABLESPACE LAB01.TSLAB01
/*
```

Note: The first step of this job deletes the image copy data set that is going to be created, in case it still exists from a previous execution. In a real scenario, you should not delete any image copy data sets, unless you are sure you would not need them anymore.

This job finishes with an RC=00. Look at the following items:

- The job output under SDSF — The SYSPRINT contains the essential information regarding the execution of the COPY utility.
- We also review the output data set. It contains one record for each of the seven data pages. The first five now contain five records each, the sixth contains four records, while the last contains only one record.

**Review the information in the catalog of DB2**

Again, we run a query against the catalog table SYSCOPY, to check which information has been created there.

**Listing 25. Select statements to verify catalog data**

```sql
SELECT * FROM SYSIBM.SYSCOPY
WHERE DBNAME = 'LAB01';
```

The result is the following (only showing the first columns).
Listing 26. Output of catalog data

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>ICTYPE</th>
<th>ICDATE</th>
<th>START_RBA</th>
<th>FILESEQNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3©»</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§5E²</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§³©ç</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§U²J</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶,É</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶,OH</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶,U</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶,Wo</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>I</td>
<td>150414</td>
<td>..¶,fO</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>E</td>
<td>150414</td>
<td>..¶,&gt;-</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>E</td>
<td>150414</td>
<td>..¶,\D</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>P</td>
<td>150414</td>
<td>..¶,;</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶,fo</td>
<td>0</td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 13

The catalog table now shows the information of the two recoveries, the RECOVER TOCOPY and the new full image copy we took for table space TSLAB01 in the previous steps.

Quiescing a table space and recovering data

Insert extra records in the first test table

Subsequently, we insert again a few extra records into table TBLAB01. We use standard INSERT statements via SPUFI.

Listing 27. INSERT statements for additional test data

```sql
INSERT INTO TBLAB01 VALUES
( '2006-09-30','10.03.03',0,'LAMN','B066','USD','TAR', 90,0.0000
,'1900-01-01',2.0000,'2006-09-30','RECORD NR 31','TEXT1 31'
,'TEXT2 31',CURRENT_TIMESTAMP );
```

```sql
--
INSERT INTO TBLAB01 VALUES
( '2006-09-30','10.04.04',0,'LAMN','B066','USD','TAR',  0,0.0000
,'1900-01-01',1.1250,'2006-09-30','RECORD NR 32','TEXT1 32'
,'TEXT2 32',CURRENT_TIMESTAMP );
```

```sql
--
INSERT INTO TBLAB01 VALUES
( '2006-09-30','10.05.05',0,'LAMN','B066','USD','TAR',180,0.0000
,'1900-01-01',3.0000,'2006-09-30','RECORD NR 33','TEXT1 33'
,'TEXT2 33',CURRENT_TIMESTAMP );
```

All INSERTS return an SQLCODE 0.

Check the contents of the tables

Check the contents of both tables.
Listing 28. Select statements to check test data

```
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES,
         T01_T_PROCES,
         T01_N_AGENCY,
         T01_C_USERID;
```

```
SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES,
         T02_T_PROCES,
         T02_N_AGENCY,
         T02_C_USERID;
```

The first table now contains 33 records, and the second table remained untouched with 25 records.

**QUIESCE the first table space**

This time we run the **QUIESCE** utility against the first table space. The only thing a **QUIESCE** utility does is returning as a Relative Byte Address (RBA) and Log Record Sequence Number (LRSN), which are pointers to the log that can later be used to recover data until exactly that point in time.

We now use the following JCL.

**Listing 29. Sample JCL to run **QUIESCE** utility**

```
//******************************************************************************
/* STEP QUIESCE: TAKE QUIESCE OF TABLE SPACE                                   */
*******************************************************************************/

//QUIESCE EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.QUISCE'
SYSPRINT DD *
QUIESCE TABLESPACE LAB01.TSLAB01
```

The execution of the JCL returns an RC=00. When the job has finished, review the job output under SDSF. The SYSPRINT contains essential information about the execution of the **QUIESCE** utility. It is important to take note of the LRSN or RBA.

**Listing 30. SYSPRINT output of **QUIESCE** utility**

```
DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = IBMWG.QUISCE
DSNUGUTS - PROCESSING SYSIN AS EBCDIC
DSNUGUTC - QUIESCE TABLESPACE LAB01.TSLAB01
.05 DSNUQUTA - QUIESCE SUCCESSFUL FOR TABLESPACE LAB01.TSLAB01
.06 DSNUQUTA - QUIESCE AT RBA 0000B6177747C AND AT LRSN 0000B6177747C
DSNUQUTB - QUIESCE UTILITY COMPLETE, ELAPSED TIME= 00:00:00
DSNUGBAC - UTILITY EXECUTION COMPLETE, HIGHEST RETURN CODE=0
```

**Review the catalog of DB2**

We first have another look at the catalog table **SYSCOPY** to see what information DB2 has stored so far about our activities.
Listing 31. SELECT statements to verify catalog data

```
SELECT * FROM SYSIBM.SYSCOPY
WHERE DBNAME = 'LAB01';
```

The result is the following (only showing the first columns).

### Listing 32. Output of catalog data

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>ICTYPE</th>
<th>ICDATE</th>
<th>START_RBA</th>
<th>FILESEQNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3©»</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3£6</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§4©Ç</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§4£1</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.É,</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.ÖH</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.Wo</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ßJ</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ü.</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ßQ</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.ßO</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>P</td>
<td>150414</td>
<td>..¶..§</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Q</td>
<td>150414</td>
<td>..¶.È@</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 14

The catalog table now shows the information of the QUIESCE executed in our previous step.

If we show the last entry of the table above in hexadecimal format, we see that the same RBA we found in the SYSPRINT of the QUIESCE execution is shown in column START_RBA.

### Listing 33. Output of catalog data in hex format

```
LAB01  TSLAB01  0  Q  150414 ..¶.È@  0
DCCFF4444EECDCCFF444444444444F444D444444F4448B1774444444444444F
3120100002321261000000000000000000015041400674C69000000000000
```

Insert additional records into the first table

We now insert another set of additional records into our first test table. We again use SPUFI with standard INSERT statements:

### Listing 34. INSERT statements for additional test data

```sql
INSERT INTO TBLAB01 VALUES
( '2007-09-15', '08.00.00', 0, 'LAMN', 'B066', 'USD', 'TAR',  0,1.1250
 , '2009-09-15', 'RECORD NR 34','TEXT1 34', 'TEXT2 34',CURRENT_TIMESTAMP );
--

INSERT INTO TBLAB01 VALUES
( '2007-09-15', '08.01.01', 90,2.0000
 , '2009-09-15', 'RECORD NR 35','TEXT1 35', 'TEXT2 35',CURRENT_TIMESTAMP );
```

Both INSERTS give an SQLCODE 0.

### Check the contents of the tables

We check the contents of both test tables, using the following query.
A how-to guide: Backup and recovery in DB2 for z/OS

### Listing 35. SELECT statements to check test data

```sql
SELECT * 
FROM TBLAB01 
ORDER BY T01_D_PROCES 
   , T01_T_PROCES 
   , T01_N_AGENCY 
   , T01_C_USERID;
```

```sql
--
SELECT * 
FROM TBLAB02 
ORDER BY T02_D_PROCES 
   , T02_T_PROCES 
   , T02_N_AGENCY 
   , T02_C_USERID;
```

The first table now contains 35 records, and the second table remained untouched with 25 records.

### Recover TOLOGPOINT or TORBA of first table space

Now we run the RECOVER utility with the parameter TOLOGPOINT. We could also use the TORBA parameter, which would work exactly the same way. We use the following JCL (take note we indicated the LRSN that we took from the execution of the QUIESCE utility):

### Listing 36. Sample JCL to run RECOVER (TOLOGPOINT) utility

```bash
// ************************************************************************
// * STEP RECOV1: RECOVER TO LOGPOINT
// ************************************************************************
// /*
// RECOV1  EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.RECOVR'
// SYIN DD *
// RECOVER TABLESPACE LAB01.TSLAB01
// TOLOGPOINT X'0000B617747C'
// /*
```

The execution of the RECOVERY utility results in an RC=04. When the job has finished, look at the following:

- The job output in SDSF. Here we discover why the job returned a code 04. In case of a RECOVER TOLOGPOINT, the indices are not being recovered (exactly the same as with a RECOVER TOCOPY). The index spaces of all four indices are left in status REBUILD PENDING.
- We check the status of the table and index spaces via a DB2 command, the same as in step 5.

The command to use:

```bash
-DISPLAY DATABASE(LAB01) SPACENAM(*)
```

The output shows something similar to the following.

### Listing 37. Result of the DB2 command to show the status of the table spaces within a database

```
DSNT360I -DSN1 **************************************************************************
```
5.8 Rebuilding the indices

In order to re-create the indices, we need to run the `REBUILD INDEX` utility.

**Listing 38. Sample JCL to run REBUILD INDEX utility**

```plaintext
//***************************************************************
// * STEP REBLD1: REBUILD INDEXES
//***************************************************************
//REBLD1 EXEC DSNUPROC,SYSTEM=DSN1,UID='IBMWG.RBLDIX'
//SYIN  DD *
REBUILD INDEX (ALL) TABLESPACE LAB01.TSLAB01
/*
```

The execution of this job gives a RC=00. When the job has finished, look at the following items:

- The output of the job is SDSF. The SYSPRINT gives important information about the execution of the utility.
- Check the status of the table and index spaces in the same way as in the previous step. We see now that all index spaces are in RW status.

**Check the contents of the tables**

We check for the last time the contents of our test tables, using the following query.

**Listing 39. SELECT statements to check test data**

```sql
SELECT *
FROM TBLAB01
ORDER BY T01_D_PROCES
  , T01_T_PROCES
  , T01_N_AGENCY
  , T01_C_USERID;
--
SELECT *
FROM TBLAB02
ORDER BY T02_D_PROCES
  , T02_T_PROCES
  , T02_N_AGENCY
  , T02_C_USERID;
```
The first table now contains 33 records. This was the situation at the moment we ran the `QUIESCE` utility. The `RECOVER TOLOGPOINT` utility recovered the first table space to exactly that situation. The second table remained untouched with 25 records.

**Review the information in the catalog of DB2**

Finally, we review the information DB2 has stored in catalog table SYSCOPY:

```sql
SELECT * FROM SYSIBM.SYSCOPY
WHERE DBNAME = 'LAB01';
```

The result is the following (only showing the first columns).

**Listing 40. Output of catalog data**

<table>
<thead>
<tr>
<th>DBNAME</th>
<th>TSNAME</th>
<th>DSNUM</th>
<th>ICTYPE</th>
<th>ICDATE</th>
<th>START_RBA</th>
<th>FILESEQNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§3©»</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>C</td>
<td>150414</td>
<td>..§5Ӗ</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§6Ԍ</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>Y</td>
<td>150414</td>
<td>..§7 monday</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.Ӗ</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>S</td>
<td>150414</td>
<td>..¶.OH</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.Wo</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.f0</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>I</td>
<td>150414</td>
<td>..¶.&gt;.&lt;-</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>E</td>
<td>150414</td>
<td>..¶.\D</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>P</td>
<td>150414</td>
<td>..¶..f</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>F</td>
<td>150414</td>
<td>..¶.f0</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB01</td>
<td>0</td>
<td>Q</td>
<td>150414</td>
<td>..¶.E@</td>
<td>0</td>
</tr>
<tr>
<td>LAB01</td>
<td>TSLAB02</td>
<td>0</td>
<td>P</td>
<td>150414</td>
<td>..¶.?:</td>
<td>0</td>
</tr>
</tbody>
</table>

DSNE610I NUMBER OF ROWS DISPLAYED IS 15

The catalog table now shows the information of the `RECOVER TOLOGPOINT` we executed before.

**Conclusion**

We have been able to offer an idea of what can be done in DB2 with backup and recovery. It is now up to you to use this basis to implement the necessary JCL for backup and recovery in your DB2 environment.
Resources

- The Information Management area on developerWorks provides resources for architects, developers, and engineers.
- Stay current with developer technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.
- Watch developerWorks demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Get involved in the developerWorks Community. Connect with other developerWorks users while you explore developer-driven blogs, forums, groups, and wikis.
About the author

Willem de Gelder

Willem de Gelder is an internationally experienced software consultant for z/OS. He has vast experience in DB2 for z/OS from several points of view, including software development with DB2, implementation of DB2, and administration of DB2. Currently, DB2 is his main focus, but he also works with other IBM software on z/OS, such as zSecure, Optim for z/OS, and OMEGAMON for z/OS.

© Copyright IBM Corporation 2015
Trademark
(www.ibm.com/developerworks/ibm/trademarks/)