IBM DB2® Universal Database™
DB2 Problem Determination Tutorial Series

Database Engine Problem Determination

Table of Contents

Database Engine Problem Determination

About this tutorial

Audience and assumptions

Prior to taking this tutorial, the reader should already be familiar with the functionality of the DB2 product and should be comfortable executing DB2 commands and SQL statements. This includes a familiarity with backup and restore, crash and roll-forward recovery, object creation, and data access.

Tutorial objectives

One of the important tasks of the database administrator is to resolve problems with the database. Understanding how to analyze the diagnostic information that DB2 UDB provides is the first step in problem determination.

The tutorial serves as an entry-level problem determination guide to understanding and resolving DB2 UDB engine problems.

Tutorial conventions used

When a tool or utility is first mentioned, it will be in shown in **bold** text.

All commands, statements, and their output will be shown in `monospaced` font.

In general, examples shown will be Windows®-based. However, they should work fine in the UNIX® environment as well (unless otherwise noted).

Some of the examples in this tutorial show the supported options for some of the utilities. These options may change from one release to the next but the basics should always remain the same.

About the author

Aman Lalla has worked at IBM since 1997. His first two years were spent at a DB2 customer site. Aman is currently the team lead for the DB2 UDB Down Systems Division. The main responsibilities of this team include working on down systems DB2 UDB Engine Problem Determination.

You can reach Aman by locating his e-mail address in the IBM Global Directory at http://www.ibm.com/contact/employees/us .

DB2DIAG.LOG analysis

Extract of the db2diag.log

Introduction
The analysis of the db2diag.log is the first step in problem determination.

The location of the db2diag.log is determined by the DIAGPATH database manager configuration parameter. The most recent events that have occurred are at the bottom of the db2diag.log file.

Analysis
Below is an extract of a db2diag.log at DIAGLEVEL 3. For problem determination purposes it is recommended to use *DIAGLEVEL 4 if the problem can be reproduced*. The core components of the db2diag.log are as follows:

```
(1) 2002-05-17-17.30.32.140000   (2) Instance:DB2MPP  (3) Node:000
(4) PID:2204(db2bp.exe)   (5) TID:2224   (6) Appid:*LOCAL.DB2MPP.020517213032
(7) database_utilities  (8) sqlubckp   (9) Probe:26

DiagData
(10)  2cfc ffff

2002-05-17-20.17.20.793000   Instance:DB2MPP   Node:000
PID:596(db2syscs.exe)   TID:2176   Appid:
base_sys_utilities  sqleMergeSqlca   Probe:20   Database:SAMPLE

Received sqlcode 1496 for request 8000001e from node number 1

(11) Data Title:SQLCA PID:596 TID:2176 Node:000
 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: 1496   sqlerrml: 0
 sqlerrmc:
 sqlerrp : SQLESRSU
 sqlerrd : (1) 0x00000000      (2) 0x00000000      (3) 0x00000000
           (4) 0x00000000      (5) 0x00000000      (6) 0x00000001
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)       (11)
 sqlstate:
```

1. The date and timestamp of the entry made into the log.
2. The name of the instance. In this example DB2MPP.
3. The node or partition number. This number is always 0 in a single partition configuration.
4. The process ID of the application or agent.
5. The thread ID of the application or agent. This is only used on the Windows platform.
6. The application ID. This corresponds to the *LIST APPLICATIONS* command output. Each application has a unique application ID.
7. Component name. In this example the component name indicates that this entry was made by the database utilities component.
8. The name of the function in the component that is reporting an error or information.
9. The probe point in the function. This corresponds to a location in the source code of the function that has returned an error or information.
10. Diagnostic information. In this example the db2diag.log is from the Windows platform, so the information dumped is byte-reversed. In order to convert this error to a decimal SQL code you

need to byte-reverse 2cfc ffff to ffff fc2c and convert it from hexadecimal to decimal to get the SQLCODE.
Note that this number does not always convert to a valid SQLCODE. If this is the case, you should contact DB2 Support.

11. A dump of the SQLCA structure which contains the SQLCODE

Example analysis of the db2diag.log

Introduction

For this example you assume that a user has complained that they are trying to take a backup and it fails. The user does not know the error message.

Analysis

The problem determination steps are as follows:

- Open the db2diag.log (see extract in Section 1) and go to the bottom of the file. The most recent entries are at the bottom. The last entry was at 2002-05-17-20.17.20.793000.
- Make a note of the process ID (PID). In this example it is 596. This PID can be followed back in the db2diag.log to determine if there were earlier problems with this process that could have led to the reported error or warning.
- In this example PID:596 has dumped an SQLCA **(11)** which has a sqlcode of 1496. Issue the command `db2 ? sql1496` for details:

```
SQL1496W Deactivate database is successful, but the database was not
activated.

Explanation:  Database was not explicitly started on one or more nodes
when deactivate database was executed.

User Response:  Refer to the diagnostic log to see which node returns
this warning.
```

The message code (SQL1496W) indicates that the SQLCODE returned is actually a warning and not an error. Also, this message does not give us any information about our backup problem. At this point you can ignore db2diag.log entries made by PID 596.

- The next step is to look at an earlier entry in the db2diag.log that is on a different PID. In this example the entry is at 2002-05-17-17.30.32.140000 - and the PID is 2204.
- The Diag Data **(10)** has the dumped 2cfc ffff. Since this db2diag.log is from the Windows platform and is byte reversed the actual format is ffff fc2c Convert ffff fc2c to decimal and you get - 980. For details on this error issue the command `db2 ? sql0980`.

```
SQL0980C A disk error occurred.  Subsequent SQL statements
cannot be processed.

Explanation:  A disk error occurred that prevented successful
execution of the current and subsequent SQL statements.  The
application program is not permitted to issue additional SQL
statements.  For example, a recovery routine associated with
the application program cannot issue additional SQL statements.
```

```
      The database is marked as needing recovery and all applications
      using the database are prevented from accessing the database.
```

This looks like a serious error. Look at the function name **(8)** that returned this error (sqlubckp). You can look up the function name in the *API Reference*, where we find that this is the backup function. The component name **(7)** confirms that a database utility has failed.

You now have enough information to determine that the backup failed due a disk error and pursue the reason for the disk problem outside of DB2 UDB.

Summary
The above methodology can be applied to all types of DB2 UDB engine problems. This concludes our analysis of the db2diag.log.

Engine components

When analyzing the db2diag.log it is useful to know the names of the components that report the errors or warnings. All function names in DB2 UDB begin with the letters "sql" followed by an abbreviated component name. Below is a list of some of the most commonly used engine components.

sql, squh -

- DB2 Backup and Restore

Sqb -

- DB2 Buffer Pool Services - Responsibilities include buffer pools, data storage management, table spaces, containers, I/O, prefetching, page cleaning, and occasionally data corruption (checksum errors, bad page headers, etc.).

Clp -

- DB2 Command Line Processor

Sqng -

- CodeGen - CodeGen component is a part of the SQL compiler. It represents the last phase of statement compilation. CodeGen takes as input the compiler's internal representation of the statement (i.e. the Query Graph Model (QGM) graph) and the optimizer's "plan" and produces a "section" which is later executed by the run-time component.
- Sqv - Data Services - Data services is responsible for the data type comparison and conversion routines (not NLS conversion). This component includes routines that do the following things:
    1. Convert between char, graphic, and numeric types
    2. Perform decimal, floating point, and integer arithmetic
    3. Perform date, time, timestamp arithmetic and conversions

4. Compare any data type to any other data type (limited to the comparisons DB2 supports). These comparison routines are used by Index Manager, Sort Services, Runtime, and other components.

Dlfm -

- Provides Datalinks file manager functionality to DB2 UDB which allows DB2 to manage files that are stored external to the database.

sqd, sqdx, sqdl, dart -

- DB2 Data Management Services:
    - Tables, records, long field and large-object (LOB) columns.
    - Recovery (rollforward, rollback/undo).
    - Table and record locking.

sqp, sqdz -

- DB2 Data Protection Services, Logging.

Sqx -

- DB2 Index Manager

squ, sqi, sqs, squs -

- DB2 Load, Import, Export, Sort

sqno, sqnx, Runstats (spans multiple components) -

- DB2 Query Optimizer, Explain and Runstats

sqo, sqt, sqz -

- DB2 Engine Operating System Services

sqe, sqkd, sqkf -

- DB2 Process Model</yy>
- BSU
- BDS
- FCM EEE infrastructure
- DB2 Connect/Gateway Connection pooling
- DB2 Connect XA
- Concentrator
- db2start / db2stop
- Create database / Drop database at node
- Activate database / Deactivate database
- Add node / Drop node
- Interrupt handling
- Node failure and recovery
- DB2 Engine Operating System Services

Database crashes

Introduction

As the database administrator you need to understand and distinguish the difference between a database crash and an application crash. Often the problem description suggests a user application crash and it is up to the database administrator to determine if the database has crashed as a result of this or vice versa.

Analysis: Determine if the database instance has crashed

To determine if the database instance has crashed the db2diag.log at various probe points must be examined. As mentioned in Section 1, the location of the db2diag.log is determined by the *DIAGPATH* database manager configuration parameter. From this file we get the following example extract, generated when DB2 crashes:

```
2002-05-13-18.17.10.131162   Instance:db2inst1   Node:000
PID:53003(db2agent (GCXSDSN) 0) Appid:*LOCAL.db2inst1.020513181115
oper_system_services  sqloEDUCodeTrapHandler   Probe:20
Database:UUSERDB
.
Signal number received
0000 000b                                    ....
```

In this example the function sqloEDUCodeTrapHandler returns a Signal 0000 000b in hexadecimal (a signal generated only when DB2 has crashed). Convert this to decimal and you get 11. This means that the db2 signal handler has caught a signal #11.

On the UNIX platform the header file called signal.h is usually located in /usr/include/sys. In this example you will determine that a signal #11 is a segmentation violation (SIGSEGV):

Extract of the signal.h header file

```
…
#define SIGBUS    10    /* (*) bus error (specification exception) */
#define SIGSEGV   11    /* (*) segmentation violation */
#define SIGSYS    12    /* (*) bad argument to system call */
…
```

This is the first indication that the database has indeed crashed due to a segmentation violation and the database signal handler has caught the signal. The next step is to determine the process ID (PID) that has crashed. We return to the db2diag.log file, to find the "abnormally terminated process":

```
2002-05-13-18.17.10.199694   Instance:db2inst1   Node:000
 PID:11322(db2tcpcm 0)   Appid:none
 oper_system_services  sqloEDUSIGCHLDHandler   Probe:20
 .
 PID of abnormally terminated child process:
 0000 cf0b                                    ....
```

The function sqloEDUSIGCHLDHandler at probe 20 has dumped the PID in hexadecimal 0000 cf0b. Convert this to decimal and you get 53003.

The db2 signal handler for a signal #11 will dump a trap file, and the naming convention use the PID in the file name. For this example you will get a file called t53003.000 in the **../sqllib/db2dump** directory. On some platforms such as AIX a CORE file is generated as well.

The trap file contains a stack traceback of all the functions on the stack for the process that crashed. Start at the top of the trap file and look for the point where a Signal#11 was encountered:

Header from t53003.000 trap file

```
DB2 (db2inst1.000) : db2agent (GCXSDSN) 0 (0x1)
2002-05-13-18.17.03.884465
signal #11 encountered, stack traceback follows:
siginfo_t (length=128)
...
PC location: __0FKMemTreeDelPP6ISMemNodeP6HSMemSeg + 0x150
.
/home01/db2inst1/sqllib/adm/db2sysc: sqlo_trce + 0x310
/home01/db2inst1/sqllib/adm/db2sysc: sqloEDUCodeTrapHandler + 0x34

__0FNMemReleaseSegP6HSMemSeg + 0x44
  sqlofmblkEx + 0x390
```

In this example you can see that you have trapped in a function called MemTreeDel. More information on the crash can be found in the operating systems error logs such as the error report (errpt) on AIX and the Messages files on Sun Solaris.

<u>Conclusion</u>
Signal #11 usually indicates a defect with DB2 and as such all relevant information should be passed over to the DB2 Support Team.

Troubleshooting locking problems

Introduction

DB2 UDB is a multi-user database and as such the database engine provides a locking mechanism to avoid resource conflicts and to provide data integrity.

Understanding locking is important for both the database administrator and application developer. The database administrator must be able to analyze snapshots and even monitors to determine which application is causing the locking problems. The application developer will design the application so as to minimize the impact of the application locks.

Collecting data for locking problems

A common user symptom of a locking problem is an application hang. A hang usually appears as "Lock Wait" within the database engine. To confirm that an application is in "Lock Wait", the database administrator should take a lock snapshot and an application snapshot.

When there is reason to believe that a locking conditions is being encountered the following steps are suggested:

Step 1: Take a lock snapshot

```
db2 get snapshot for locks on sample
```

Step 2: Take an application snapshot

```
db2 get snapshot for applications on sample
```

Step 3: Get a list of applications connected to the database

```
db2 list applications all
```

Analyzing a lock snapshot

- Open a lock snapshot that you captured and search for an application status of **"Lock-wait"**. There may be more than one application is this state.

```
Extract Of A Lock Snapshot For Application Handle 1

Application handle                      = 1
Application ID                          = *LOCAL.amanl.021106192019
………
Authorization ID                        = AMANL
Application status                      = Lock-wait
Status change time                      = 11-06-2002 14:20:28.739223
        ……….
        Locks held                              = 3
```

```
Total wait time (ms)                        = 59003
```

Take note of the application handle which is 1 in this example.

- Determine the ID of the application that is holding the lock(s) that application handle 1 is waiting on.

```
…………………….
Subsection waiting for lock               = 0
ID of agent holding lock                  = 0
Application ID holding lock               = *LOCAL.amanl.021106191959
……………………
```

It can now be determined that the application with handle 1 is waiting on an application with handle 0.

- Determine the state of application handle 0

```
Snapshot timestamp                          = 11-06-2002 15:06:35.977515
……………….
Application handle                          = 0
Application ID                              = *LOCAL.amanl.021106191959
Sequence number                             = 0001
Application name                            = db2bp
Authorization ID                            = AMANL
Application status                          = UOW Waiting
Status change time                          = 11-06-2002 14:20:02.057647
```

Note that the application is in **"UOW Waiting"** state. This means that application is either doing some other processing or has an open UOW (unit of work). If the application is idle, i.e. there is no more work to do then the status will read "Idle". An idle application does not hold any database locks.

Also compare the "status change time" and the "snapshot timestamp" which indicates in the above example that the application has been in "UOW Waiting" status for over 40 minutes.

```
………………………….
Lock object type                          = Row
Lock mode                                 = Exclusive Lock (X)
Lock mode requested                       = Next Key Share (NS)
……………………………..
```

The lock snapshot shows that application handle 1 is requesting a Next Key Share (NS) lock but is waiting on application handle 0 that has an Exclusive Lock (X). Analyze the application snapshot to get more details about application handle 0.

An (X) lock is not compatible with an (NS) lock, hence the Lock-Wait. The DB2 UDB Administration Guide outlines lock compatibilities.

Extract Of An Application Snapshot For Application Handle 0

```
……………..
Elapsed time of last completed uow (sec.ms)= 0.000000
UOW start timestamp                        = 11-06-2002 14:20:01.972039
UOW stop timestamp                         =
UOW completion status                      =
Open remote cursors                        = 0
……………..
……………..
Rows inserted                              = 0
Rows fetched                               = 0
Blocking cursor                            = NO
Dynamic SQL statement text:
insert into staff values (99,'aman',20,'Sales',5,10000,75.60)
Agent process/thread ID                  = 355954
```

The application snapshot confirms that the UOW has begun but has not yet been completed. This indicates that the UOW is still open. If the database were to crash at this point, this transaction would be lost since it has not yet been written out to disk.

The snapshot also tells us that the application handle is executing a dynamic SQL statement, an insert into the table STAFF in this case.

```
…………………….. .
Number of SQL requests since last commit   = 1
Commit statements                          = 0
Rollback statements                        = 0
…………………….. .
Locks held by application                  = 3
Lock waits since connect                   = 0
Time application waited on locks (ms)      = 0
……………………….. .
```

Application handle 0 has not yet committed and is holding 3 locks. Details about these locks can be seen in the Lock Snapshot.

```
……………………….. .
……………………….. .
 List Of Locks
 Lock Object Name            = 39
 Node number lock is held at = 0
 Object Type                 = Row
 Tablespace Name             = USERSPACE1
 Table Schema                = AMANL
 Table Name                  = STAFF
 Mode                        = X
 Status                      = Granted
 Lock Escalation             = NO
…………………………….
…………………………….
```

One of the 3 locks that application handle 0 is holding is an X lock which has been granted.

From the above analysis it can be determined that the cause of this locking condition is that an application is executing an INSERT into a table which requires exclusive access and has been granted all the required locks but has not committed. Any application with a weaker lock will have to wait for the UOW to commit. (This is the default behavior for the Cursor Stability (CS) isolation level.) Application handle 0 should commit more frequently.

Troubleshooting recovery problems

Introduction

Taking frequent database backups is essential for the recovery of databases in the event of a disaster. Understanding the recovery options and high level processing of recovery can reduce the anxiety that is commonly associated with recovery. Database recovery is one of the crucial tasks that is performed by the database administrator.

Main backup / restore processes

db2agent
co-ordinator of all child processes and handles all communications with the application.

db2bm
db2 buffer manipulator - uses prefetchers to process I/O requests. Places data into or reads data from the backup restore buffers. Passes data to the db2med process.

db2med
db2 media controllers - performs I/O with backup destination . Data is written from or read into the backup / restore buffers. Buffers passed back to db2bm.

Using db2ckbkp to check a backup image

DB2 UDB now provides a utility called **db2ckbkp** that can be used to validate a backup image. For details about the options, run db2ckbkp with no arguments.
Below is an example scenario in which the db2ckbkp utility may be used.

```
SQL1013N  The database alias name or database name "ITB20" could not be
found.

2002-10-12-10.28.52.710000   Instance:DB2   Node:000
PID:-2296-439445(DB2SYSC.EXE)   TID:-439445
Appid:*LOCAL.DB2.021012172526
database_utilities  sqludMRWarn   Probe:40   Database:ITB20
.
16f6 ffff 433a 5c44 4232 4261 636b 7570        .öÿÿC:\DB2Backup
5c49 5442 3230 2e30 5c44 4232 5c4e 4f44        \ITB20.0\DB2\NOD
4530 3030 305c 4341 544e 3030 3030 5c32        E0000\CATN0000\2
3030 3230 3931 335c 3039 3237 3339 2e30        0020913\092739.0
3031 00                                       01.
```

The above error indicates that the backup image cannot be found. It is possible that the image exists in that specified location but may be corrupted. To verify that the backup image is good, run db2ckbkp with the -a option to dump as much information as possible.

If there is no problem with the backup image the following message will be returned :

```
Image Verification Complete - successful.
```

Basic history file analysis

Every DB2 UDB database has a history file that records administrative operations. A recovery history file is created with each database and is automatically updated. During a database migration, the history file is migrated as well. The history file can be accessed by issuing the following command:

```
db2 list history all for <dbname>
```

The database history file is invaluable in a recovery scenario. The history file is individually restorable from any backup image. If the current database is unusable or not available, and the associated recovery history file is damaged or deleted, an option on the **RESTORE** command allows only the recovery history file to be restored. The recovery history file can then be reviewed to provide information on which backup to use to restore the database. For example, you can restore the history file for our sample database with this command:

```
db2 restore database sample history file
```

The recovery history file provides enough information to recover a database or table space using a backup image.

Extract Of A History File

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20021107151552001   N    D  S0000002.LOG S0000003.LOG
 ----------------------------------------------------------------------------
  Contains 2 tablespace(s):

  00001 SYSCATSPACE
  00002 USERSPACE1
 ----------------------------------------------------------------------------
    Comment: DB2 BACKUP SAMPLE ONLINE
 Start Time: 20021107151552
   End Time: 20021107151613
 ----------------------------------------------------------------------------
  00007 Location: c:\backups\SAMPLE.0\DB2MPP\NODE0000\CATN0000\20021107
```

Some of the important information in the history file is

Op: This is the operation that was performed. In the exampled above it is a "B" which stands for backup. The *UDB Administration Guide* contains a list of the possible values for **Op**.

Obj: This is the granularity of the backup. "D" for database backup . "T" for tablespace backup.

Earliest Log: In the case of an online backup, this is the first log required for the rollforward operation.

Current Log: The last log that was written to when the backup completed. In the case of an online backup, this is the minimum log required for the backup to complete.

Point-in-time recovery

The Earliest Log and the Current Log in the history file are the range of logs used when the backup was running, and a considered part of the backup. Without these logs the backup cannot complete and the database will remain in roll-forward pending state.

If an attempt is made to roll-forward to a point-in-time that is in the range of logs between the earliest log file and the current log file, the db2diag.log will dump a message such as follows:

```
2002-11-18-04.18.25.860819   Instance:ispbdw01   Node:008
PID:334510(db2agntp (DBPBDW) 8)   Appid:*N0.ispbdw01.021118091637
recovery_manager  sqlpCheckStopTime   Probe:55   Database:DBPBDW
.
Invalid stop time: 3dd7 2250 0000 0000 3dd7 d45f 0000 0000
```

Troubleshooting problems with the LOAD utility

Introduction

LOAD is a utility used to bulk-load data from a file or pipe into a database. LOAD consists of up to three phases. The **load phase** loads the data, the **build phase** builds the indexes if there are any, and the **delete phase** deletes duplicates if there are unique indexes.

LOAD processes

Main load processes: (X is a number identifying one of many)

db2agent	
db2lmr	Load Media Reader. This process reads the input.
db2lbmX	Load Buffer Manipulator. Writes loaded data to the database.
db2lfrmX	Load Formatter. Formats the input data into internal form.
db2lrid	Ridder. Organises data to be written to disk. Performs the index sort.
db2lmwX	Load Media Writers. Write the load copy.

LOAD failure

What do you do when a LOAD fails and the table space is not accessible ?
The following options are available.

- Restart the LOAD
- Terminate the LOAD
- Restore from the most recent backup and rollforward to a point-in-time before the failed LOAD
- Drop and rebuild the table space.

A typical message for a failed load in the db2diag.log is as follows:

```
2002-10-23-15.46.55.454018   Instance:db2inst1   Node:000
PID:2038(db2agent (DWCORP))   Appid:*LOCAL.db2inst1.021023174208
database_utilities  call_sqluvload   Probe:40   Database:DWCORP
.
Load Error: Load failed for table DW      .DW_PROD_MED in tablespace 7.
```

The component name and function name indicates a problem with the Load utility. In addition, you can issue the following command to examine the state of the table space whose table is being loaded into:

```
db2 list tablespaces show detail
```

Typical output is as follows :

```
Tablespace ID                          = 15
Name                                   = TS_BOYDC2
Type                                   = System managed space
Contents                               = Any data
State                                  = 0x0004
       Detailed explanation:
       Quiesced: EXCLUSIVE
...
Number of quiescers                    = 1
Quiescer 1:
Tablespace ID                      = 15
Object ID                          = 43
```

Use the **db2tbst** tool that is provided with DB2 to determine that state of the table space.

For the above example, you would issue the following command

```
db2tbst 0x0004
```

This returns:

```
State = Quiesced Exclusive
```

In this state, the table space is inaccessible. To remove the table space from this state, issue:

```
db2 quiesce tablespaces for table FDE_OPCF.ACTION reset
```

where FDE_OPCF.ACTION is the name of the table being loaded into.

In Version 7 and, previous versions, LOAD issues a QUIESCE EXCLUSIVE on the table at the start of the load, and issues a QUIESCE RESET on commit. Load also uses the table space state LOAD PENDING while in the load or build phase, or DELETE PENDING during the delete phase. These states persist after the load is interrupted or the load fails. They can only be released by performing a LOAD RESTART, a LOAD TERMINATE, or a LOAD REPLACE.

The LOAD temporary files

In the instance directory, there is a subdirectory called "load/DB2XXXXX.PID/DB2YYYYY.PID", where XXXXX and YYYYY are the pool (table space) and object ID of the table involved in the load. This subdirectory only exists while the load is running and is deleted when the load finishes. The users are warned not to tamper with this directory in any way. If they do, recovery is more difficult.

This directory typically contains the location file, the minimum recovery file, the messages file, and the load control files. The location file and the minimum recovery file should always be in this directory. If the user specified a path other than the default for the TEMPFILES PATH parameter in the db2 load command then the other files may be in this directory.

load.loc
The location file contains the directory where the control files and message files normally reside. It is controlled by the TEMPFILES PATH parameter to the load command. It defaults to the same directory as the location file. It is an ASCII file.

load.min
The minimum recovery file is used for LOAD REPLACE, just in case the control files go missing. For example, the user may specify the TEMPFILES PATH to point to some other directory and then delete that directory. LOAD REPLACE is still possible in this scenario. This is a binary file and may be converted to ASCII using the internal tool **sqluckct**.

load.CT1, load.CT2
These are the load control files. Without these files a user cannot do a LOAD TERMINATE or a LOAD RESTART. The files use a shadowing protocol to make sure that at least one version is correct at any time.They are binary files and can be converted to ASCII using sqluckct.

load.msg
This is the binary form of the load messages file that appears at the end of the load. It may be converted to ASCII by the load query command

```
db2 load query filename
```

where filename is the name of the message file without the .msg extension.

Bringing a table back online

Occasionally a table space will be stuck in a LOAD PENDING or DELETE PENDING state after a load fails. The usual method of rolling back a failed load is to use LOAD TERMINATE. The original load command is rerun substituting TERMINATE for the LOAD command. The load may also be restarted using LOAD RESTART. The original load command is substituted with a RESTART command. For example, suppose you interrupted the following load.

```
db2 load from File1 of del insert into T1
```

The table spaces involved in the load would be left in load pending state if you interrupted during the load or build phases, and the delete pending state if you interrupted during the delete phase. You may issue any of the following commands:

```
db2 load from File1 of del terminate into T1
```

LOAD TERMINATE will attempt to rollback the failed load. The file name is ignored in load terminate.

```
db2 load from File1 of del restart into T1
```

LOAD RESTART will attempt to restart the load.

```
db2 load from File1 of del replace into T1
```

LOAD REPLACE will replace all of the data in T1 with data from File1. Notes: All of these types of loads will take the table space out of LOAD PENDING (or DELETE PENDING) if they complete successfully. LOAD TERMINATE will typically mark the indexes as requiring a rebuild and will truncate the data to the size it was before the LOAD INSERT. One can also TERMINATE or RESTART a failed LOAD REPLACE, but in this case TERMINATE will truncate the table; LOAD TERMINATE cannot recover the data that existed before the LOAD REPLACE was issued.

If any of the above methods fails to bring the table back online, it can often be a quiesce problem, or, in some cases a control file problem. You might get a helpful and informative message such as

```
SQL3508N  Error in accessing a file or path of type
"RESTART/TERMINATE INFO" during load or load query.
Reason code: "1".  Path: "".
```

Conclusion
The discussion outlined will resolve most problems that are associated with the LOAD utility. This concludes the discussion on the LOAD utility.

Summary and feedback

What you should now know

You should now be familiar with diagnosing general DB2 UDB engine related problems.

You should now have a better understanding of the entries that are logged in the db2diag.log, the history files and the LOAD messages files.

Feedback

Please take a moment to give us your thoughts on this tutorial by completing the feedback form on the DB2 Technical Support site at http://www.ibm.com/software/data/db2/udb/winos2unix/support .