

---



# DBMaster

---

## Backup Restore User's Guide

---

P-E5002-Backup/Restore user's Guide

Version: 02.00

**Document No: 43/DBM43-T02232006-01-BARG**

**Author: DBMaster Production Team, Syscom Computer Engineering CO.**

**Publication Date: February 23, 2006**

---

## Content

1. Introduction.....	1-1
1.1 Additional Resources .....	1-1
1.2 Document Conventions .....	1-2
2. Overview.....	2-1
2.1 Types of Database Failures .....	2-1
2.1.1 SYSTEM FAILURES .....	2-1
2.1.2 MEDIA FAILURES .....	2-1
2.2 Recovery from Database Failures.....	2-2
2.2.1 JOURNAL FILES .....	2-2
2.2.2 CHECKPOINT EVENTS.....	2-2
2.2.3 RECOVERY STEPS .....	2-3
2.2.4 FORCING DATABASE STARTUP.....	2-3
2.3 Backup Directory .....	2-4
2.4 Backup Server .....	2-6
2.5 Compressing Backup Files .....	2-7
2.6 JServer Manager .....	2-7
2.7 JConfiguration Tool.....	2-7
2.8 Key words in DMconfig.ini .....	2-7
2.9 JData Transfer Tool .....	2-8
2.10 Load/Unload.....	2-8
2.11 Dmsql command.....	2-8
2.12 XTT/XTM .....	2-8
2.13 Rollover .....	2-9
2.14 DMRestoreTB .....	2-9
3. JServer Manager .....	3-1
3.1 Backup a Database .....	3-1
3.1.1 ON-LINE FULL BACKUP BY BACKUP SERVER.....	3-2
3.1.2 ON-LINE FULL BACKUP INTERACTIVELY .....	3-3
3.1.3 ON-LINE FULL BACKUP TO TAPE .....	3-6
3.1.4 OFF-LINE FULL BACKUP .....	3-8
3.1.5 OFF-LINE FULL BACKUP TO TAPE .....	3-11
3.1.6 INCREMENTAL BACKUP BY BACKUP SERVER.....	3-12

3.1.7	INCREMENTAL BACKUP INTERACTIVELY .....	3-14
3.1.8	INCREMENTAL BACKUP TO CURRENT INTERACTIVELY .....	3-15
3.1.9	MULTIPLE TAPE BACKUP .....	3-17
3.2	Restore a Database.....	3-18
3.2.1	RESTORE A DATABASE FROM DISK.....	3-18
3.2.2	RESTORE A DATABASE FROM TAPE .....	3-23
4.	JConfiguration Tool.....	4-1
4.1.1	INCREMENTAL BACKUP MODE SETTINGS.....	4-1
4.1.2	BACKUP FILE OBJECT MODE .....	4-2
4.1.3	SETTING THE BACKUP FILE DIRECTORY .....	4-2
4.1.4	STARTING BACKUP SERVER.....	4-2
4.1.5	ENABLING COMPRESS BACKUP FILES .....	4-3
4.1.6	ENABLING BACKUP READ ONLY TABLESPACE.....	4-3
4.1.7	SETTINGS FOR FULL BACKUP PROCESS.....	4-3
4.1.8	SETTINGS FOR INCREMENTAL BACKUP PROCESS.....	4-4
5.	Keywords in DMconfig.ini .....	5-1
5.1	DB_BkSvr .....	5-1
5.2	DB_BkDir.....	5-1
5.3	DB_BkOdr .....	5-1
5.4	DB_BkFoM .....	5-2
5.5	DB_BkFrm.....	5-2
5.6	DB_BkFul .....	5-3
5.7	DB_BkItv .....	5-3
5.8	DB_BkCmp.....	5-3
5.9	DB_BKRTS .....	5-3
5.10	DB_BkTim .....	5-4
5.11	DB_BkZIP .....	5-4
5.12	DB_BMode .....	5-4
5.13	DB_FBkTm .....	5-4
5.14	DB_FBkTv .....	5-5
5.15	DB_RTime .....	5-5
6.	JData Transfer Tool .....	6-1
6.1	Importing data from Text.....	6-1
6.2	Importing data from XML .....	6-7
6.3	Importing data from ODBC.....	6-9
6.3.1	IMPORTING ODBC DATA FROM TABLES.....	6-11

6.3.2	IMPORTING ODBC DATA USING SQL SELECT STATEMENTS	6-14
6.3.3	IMPORTING ODBC DATA THROUGH AN XML BATCH FILE	6-17
6.4	Exporting Data to Text	6-21
6.5	Exporting Data to XML	6-26
7.	UNLOAD/LOAD	7-1
7.1	UNLOAD	7-2
7.1.1	UNLOAD DB [DATABASE]	7-2
7.1.2	UNLOAD TABLE	7-3
7.1.3	UNLOAD SCHEMA	7-3
7.1.4	UNLOAD DATA	7-3
7.1.5	UNLOAD PROJECT	7-4
7.1.6	UNLOAD MODULE	7-4
7.1.7	UNLOAD [PROC   PROCEDURE]	7-4
7.1.8	UNLOAD [PROC DEFINITION   PROCEDURE DEFINITION]	7-4
7.2	LOAD	7-5
7.2.1	LOAD DB [DATABASE]	7-5
7.2.2	LOAD TABLE	7-6
7.2.3	LOAD SCHEMA	7-6
7.2.4	LOAD DATA	7-6
7.2.5	LOAD MODULE	7-7
7.2.6	LOAD PROJECT	7-7
7.2.7	LOAD PROC [PROCEDURE]	7-7
8.	Dmsql Command	8-1
8.1	EXPORT	8-2
8.1.1	EXPORT COMMAND INTERFACE	8-2
8.1.2	DESCRIPTION FILE	8-2
8.2	IMPORT	8-7
8.2.1	IMPORT COMMAND INTERFACE	8-7
8.2.2	DESCRIPTION FILE	8-8
9.	XTT/XTM	9-1
9.1	XML Transfer Template Tool	9-2
9.1.1	GETTING TO KNOW THE XTT TOOL	9-3
9.1.2	CREATING A NEW XTT	9-12
9.1.3	EDITING AN XTT	9-14
9.1.4	GENERATING A DTD	9-17
9.1.5	GENERATING AN XSD	9-17

9.1.6	GENERATING XML DATA .....	9-18
9.2	XML Transfer Mapping Tool .....	9-20
9.2.1	GETTING TO KNOW THE XTM TOOL.....	9-20
9.2.2	THE TOOLBAR .....	9-21
9.2.3	XTM OBJECT TREE .....	9-22
9.2.4	XML SCHEMA TREE .....	9-22
9.2.5	DATABASE SCHEMA TREE .....	9-22
9.2.6	CREATING AN XTM.....	9-22
9.2.7	MAPPING XPATH STATEMENTS TO XTM OBJECT NODES.....	9-24
9.2.8	EXECUTING AN XTM .....	9-26
9.2.9	XTM API FUNCTIONS.....	9-27
10.	ROLLOVER.....	10-1
10.1	Text-mode ROLLOVER usage .....	10-2
11.	DMRestoreTB .....	11-1
11.1	How to use .....	11-2
11.1.1	ENVIRONMENT:.....	11-2
11.1.2	INPUT PARAMETERS.....	11-2
11.1.3	EXAMPLES .....	11-3

# 1. Introduction

Welcome to DBMaster Backup/Restore user's guide. In every database management system, the possibility of a hardware or software failure always exists. A DBMS may fall victim to failures without warning. After a failure occurs, a DBMS should have some methods of recovering the information. This is one of the main advantages a DBMS has over the old file-based systems they replaced.

DBMaster incorporates advanced data protection features to prevent data loss and downtime due to failures. These features allow DBMaster to ensure the reliability of a database and the consistency of data by providing recovery, backup, and restoration features.

The general conception about backup/restore, please consult DBA User's Guide. This book emphasizes on the tools and ways about backup/restore a database administrator should understand. So the details and Examples of using Backup/Restore tools and ways will be described in the following sections. It is intended to teach those who are understanding with using DBMaster, and have knowledge of how a database works.

## 1.1 Additional Resources

DBMaster provides a complete set of DBMS manuals in addition to this one. For more detailed information on a particular subject, consult one of the books listed below:

- For an introduction to DBMaster's capabilities and functions, refer to the *DBMaster Tutorial*.
- For more information on designing, administering, and maintaining a DBMaster database, refers to the *Database Administrator's Guide*.
- For more information on database management, refer to the *JDBA Tool User's Guide*.
- For more information on database server management, refer to the *JServer Manager User's Guide*.
- For more information on configuring DBMaster, refer to the *JConfiguration Tool Reference*.
- For more information on the native ODBC API, refer to the *ODBC Programmer's Guide*.
- For more information on the dmSQL interface tool, refer to the *dmSQL User's Guide*.
- For more information on the SQL language used in dmSQL, refer to the *SQL Command and Function Reference*.

- For more information on error and warning messages, refer to the *Error and Message Reference*.
- For more information on the DBMaster COBOL Interface, refer to the *DCI User's Guide*.

## 1.2 Document Conventions

This book uses a standard set of typographical conventions for clarity and ease of use. The NOTE, Procedure, Example, and Command Line conventions also have a second setting used with indentation.

CONVENTION	DESCRIPTION
<i>Italics</i>	Italics indicate placeholders for information that must be supplied, such as user and table names. The word in italics should not be typed, but is replaced by the actual name. Italics also introduce new words, and are occasionally used for emphasis in text.
<b>Boldface</b>	Boldface indicates filenames, database names, table names, column names, user names, and other database schema objects. It is also used to emphasize menu commands in procedural steps.
KEYWORDS	All keywords used by the SQL language appear in uppercase when used in normal paragraph text.
SMALL CAPS	Small capital letters indicate keys on the keyboard. A plus sign (+) between two key names indicates to hold down the first key while pressing the second. A comma (,) between two key names indicates to release the first key before pressing the second key.
<b>NOTE</b>	Contains important information.
➡ <b>Procedure</b>	Indicates that procedural steps or sequential items will follow. Many tasks are described using this format to provide a logical sequence of steps for the user to follow
➡ Example	Examples are given to clarify descriptions, and commonly include text, as it will appear on the screen. Other forms of this convention include Prototype and Syntax.
CommandLine	Indicates text, as it should appear on a text-delimited screen. This format is commonly used to show input and output for dmSQL commands or the content in the dmconfig.ini file

Table 1-1 Document Conventions

## 2. Overview

This chapter introduces some basic knowledge and ways about how to backup and restore a database with concrete examples. Users can select the way which you acquaint with or convenient to accomplish this function.

DBMaster provided many useful tools and ways as follow sections to backup and restore database.

### 2.1 Types of Database Failures

Database failures can be divided into two types: system failures and media failures. When either of these types of failure occurs, there is the possibility of data inconsistency or data loss in a database. A DBMS should provide facilities for recovering from failures and for replacing a damaged database with a backup copy.

#### **2.1.1 SYSTEM FAILURES**

---

A system failure, known as an instance failure, is a failure from the main memory in a computer system. System failures may be caused by a power failure, an application or operating system crash, a memory error, or other reason. The result is the unexpected termination of DBMS.

Applications and active transactions can terminate abnormally when a system failure occurs. Since the exact state of a transaction in progress or a transaction that has not been completely written to disk cannot be reliably be determined after a system failure, these types of transactions require recovery. The most common method of protection against system failures is the use of a transaction log, or a journal file.

#### **2.1.2 MEDIA FAILURES**

---

Media failure (also known as disk failure) is a failure of the disk storage system of a computer system. Media failures are usually caused by physical trauma to the disk itself, such as a head crash, fire, or exposure to vibration or g-forces outside its physical operating limits.

There is nothing to prevent the loss of data on an affected disk when a media failure occurs. One or more files may be physically damaged because of the failure, requiring restoration of the database. However, the database can be successfully restored if the database provides backup and restoration facilities.



## 2.2 Recovery from Database Failures

The goals of recovery after a database failure are to ensure committed transactions are reflected in the database, ensure uncommitted transactions are not reflected in the database, and to return to normal operation as quickly as possible while insulating users from problems caused by the failure.

DBMaster uses journal files and checkpoints to achieve these goals. The journal files and checkpoints work together to ensure that all transactions are recovered in as short a time as possible, with as little effect on users as possible.

### 2.2.1 JOURNAL FILES

---

Journal files provide a real-time, historical record of all changes made to a database, and the status of each change. In the event of a system failure, the historical record of changes maintained in the journal file allows DBMaster to recover and redo changes made by transactions that completed but were not written to disk, or undo changes made by transactions that terminated abnormally.

If a database is running in backup mode, the journal files will also store additional information that DBMaster can use to restoration. This information will remain in the journal files until they are backed up; after you back up the journal files DBMaster will free this space for use by new transactions.

During the restoration process, DBMaster will add the information from the backup journal files to a backup copy of the database. Therefore, only the journal files that contain the changes made to the database between full backups need to be backed up.

### 2.2.2 CHECKPOINT EVENTS

---

A checkpoint is a system event in which the database is brought to a clean state. DBMaster writes all journal records and all dirty data pages from its internal memory buffers to disk, and reclaims journal blocks that are no longer required for backup or recovery purposes. DBMaster can reclaim journal blocks that contain non-active transactions that completed before the start of the oldest active transaction.

Startup time after an instance failure is reduced after taking a checkpoint. DBMaster writes the time of the last checkpoint and a list of all transactions active at the time of the checkpoint to the journal file header. During database recovery, DBMaster uses this information to determine which transactions should be undone, which should be redone, and which should be ignored.

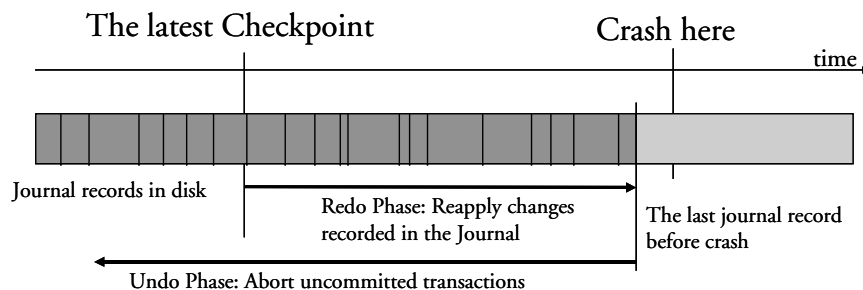
DBMaster will automatically take a checkpoint when the journal files are full to try to reclaim some journal blocks to reuse. If the checkpoint cannot reclaim enough space to complete the current transaction, the transaction will be aborted. DBMaster will also automatically take a checkpoint when the database starts and shuts down, and when an online backup is performed.

Database administrators can initiate a checkpoint manually by executing the CHECKPOINT command. The optimal interval between two checkpoints depends on the frequency of activity in the database, the average size of transactions, and the size and number of journal files. Since these factors may vary significantly from database to database, the optimal interval is best determined through experience. Manual checkpoints reduce the amount of time required to start, terminate, and backup a database, as well as the possibility that a full journal will be encountered.

Checkpoints may require a significant amount of time to complete, depending on the size and number of transactions since the last checkpoint. Any transactions that are active when a checkpoint occurs need to wait for DBMaster to calculate which journal records it can reclaim, but do not need to wait while DBMaster actually writes journal records and dirty data pages to disk.

### 2.2.3 RECOVERY STEPS

DBMaster provides support for automatic recovery when the database is started after a system failure or when an error occurs during startup. During the recovery process, DBMaster always performs two separate steps: redo and undo.



The first step in the recovery process is to redo (or reapply) all changes made to the database that are recorded in the journal. This step is necessary since it is possible for a transaction to have completed before the system failure, without having all the changes made by the transaction written to the database. However, these changes are stored in the journal, and can be written to the database during this step. At the end of this step, the database contains the changes made by all committed transactions, as well as the changes made by all uncommitted transactions.

The second step in the recovery process is to undo (or rollback) all the changes made by transactions that were not completed before the system failure occurred. This step is necessary since the exact state of a transaction in progress cannot be reliably determined in the event of a system failure. These incomplete transactions must be removed since a transaction is self-contained by definition and must either complete successfully and change the data, or fail and leave the data unchanged. At the end of this step, the database contains the changes made by all committed transactions, but does not contain any changes made by uncommitted transactions.

DBMaster also provides support for starting a database after a media failure or after a system failure, which causes inconsistencies in a database that cannot be repaired during the automatic recovery process. In these cases, the database will fail to start and you would normally need to restore your database from a backup copy. However, if you have never backed up your database, you can force the database to start by setting the forced-start mode using the **DB\_ForcS** keyword in the dmconfig.ini file. This will allow you to start the database and unload the unaffected data. For more information on the forced-start mode, see "Forcing Database Startup" in *Database Administrator's Guide*.

### 2.2.4 FORCING DATABASE STARTUP

DBMaster automatically performs recovery operations if errors occur when a database starts normally. If the database cannot start up, there may be some disk errors. Disk errors require the database be restored from the most recent backup to repair it. If the database has no backups and cannot start, use the *forced startup* mode provided by DBMaster.

DBMaster supplies a forced startup option for this type of situation. To set the forced startup mode on, use the **DB\_ForcS** keyword in the **dmconfig.ini** file. Setting this keyword to 1 enables forced startup mode, and setting it to 0 disables it. When forced startup mode is on, DBMaster will skip errors when starting the database.

If the database still cannot be started, there is one remaining alternative provided in the procedure below. However, before performing this procedure, backup all data and journal files.

- To start a database when it will not start in force start database mode:
  1. Set the Forced Startup Mode to off in dmconfig.ini (DB\_ForcS = 0).
  2. Set the Start Mode to New Journal Mode in dmconfig.ini (DB\_SMode = 2).
  3. Restart the database.
  4. Reset Start Mode back to normal in dmconfig.ini (DB\_SMode = 1).

DBMaster provides the option to use a new journal to force the database to start without any recovery operations. Therefore, if errors serious enough to prevent the database from starting have occurred, the database may be in an inconsistent state.

After starting the database with this method, check the consistency of the database.

The follow section we introduce you about six tools and ways in general, and then chapter 3-9 will show you the detail.

## 2.3 Backup Directory

The backup directory specifies where the Backup Server will place backup files. The backup directory must be a directory that already exists. If the directory you wish to use does not exist, you must create it using operating system commands before you specify it as the backup directory. You should choose a backup directory on a different disk than the database files to prevent the loss of both the database and the backup files in the event of a media error.

The backup directory is specified by the **DB\_BkDir** keyword in the **dmconfig.ini** file. The value of the **DB\_BkDir** keyword may contain either a full or a relative path to the backup directory. If you do not specify a backup directory, the Backup Server will automatically create a default backup directory named **backup** under the database directory. The database directory is specified by the **DB\_DbDir** keyword in the **dmconfig.ini** file. The total length of the backup directory path must not exceed 79 characters in length.

It is not a good idea to allow the Backup Server to create and use the default backup directory if you have more than one database in the same directory. In this case, the backup history information from one database may overwrite or append to the backup history information from another database, rendering one or both of the backups unusable. To avoid this type of problem you can put each database in a different database directory, or explicitly specify a backup directory for each database. Placing each database in a different database directory is the preferred method, since this allows you to see exactly which files belong to which database.

DBMaster provides several different methods to set the backup directory. The method you choose depends on whether your database is online or offline, and whether you are more comfortable editing the configuration file directly or using the JServer Manager graphical utility.

## Using dmconfig.ini to Set Backup Directory

If the database is offline, you can set the backup directory used by Backup Server directly using the **DB\_BkDir** keyword in the **dmconfig.ini** file. The next time you start the database, Backup Server will use this directory as the backup directory. If the database is online, changing the value of the **DB\_BkDir** keyword will have no effect until the database is shut down and restarted.

- To set the backup directory using the dmconfig.ini configuration file:
  1. Open the **dmconfig.ini** file on the database server using any ASCII text editor.
  2. Locate the database configuration section for a database.
  3. Change the value of the **DB\_BkDir** keyword to a string containing the name of an existing directory to set the backup directory.
  4. Restart the database to begin using the new backup directory.

## Using dmSQL To Set Backup Directory on Line

The SetSystemOption command can be used to change the backup directory while the database is running. The general syntax for the command is:

```
SetSystemOption('bkdir', 'path')
```

The *'path'* is the full path of the new backup directory. The length of the string in *path* should not exceed 256 characters.

- Example

To change the directory path to E:/storage/database/backup/WebDB, enter the following line at the dmSQL command prompt.

```
dmSQL> SetSystemOption('bkdir', 'E:/storage/database/backup/WebDB');
```

## Using JServer Manager to Set Backup Directory

If the database is offline, you can set the backup directory used by Backup Server using the JServer Manager graphical utility. JServer Manager will automatically change the value of the **DB\_BkDir** keyword in the **dmconfig.ini** file. The next time you start the database, Backup Server will use this directory as the backup directory. If the database is online, JServer Manager can change the backup directory immediately or delay the change until the next time you restart the database. In either case, JServer Manager will also make a copy of the backup history file in the new backup directory. For directions on how to set the backup directory using JServer Manager, refer to the *JServer Manager User's Guide*.

## Setting Multiple Backup Paths

DBMaster also supports multiple backup file paths for users. This function is useful when a user tries to save to a backup path, but the backup path does not provide enough space for the backup to be completed. If the multiple backup option is set DBMaster will then shunt the remaining data to be backed up to secondary backup locations so that the backup can be properly performed. Users are able to use multiple backup paths on full or incremental backups. DBMaster has the following constraints when backing up information using multiple backup paths:

When a database system attempts to backup files, it will try to store files in the paths one by one for each file. For example, when storing a file to backup directory 1 and the directory does not

have enough space to store the file, then the file is shunted to backup directory 2, and so on. If all backup directories are full an error message will be returned.

Only one backup directory can be used to backup files on the slave sites.

FOs must backup in the first backup directory.

The maximum number of backup paths is 32.

➡ Example:

When setting multiple backup paths DBMaster conforms to the following structure:

```
DB_BKDIR = <BKDIR 1> <SIZE 1> <BKDIR 2> <SIZE 2> <BKDIR 3> <SIZE 3>...
```

```
<BKDIR n> : the n's backup path
```

```
<SIZE n> : the size of the n's backup path ( 4k per unit )
```

So when setting multiple backup paths for the database DB1 you need to set the paths in DB\_BKDIR.

```
DB_BKDIR = /home/usr/dbmaster/bk 5000 /home2/backup 1000
```

When the available space in home/usr/dbmaster/bk is full the database will backup at home2/backup.

## 2.4 Backup Server

Backup Server runs in the background and performs online full and incremental backups on a regular schedule, as journal files become full, or both. This flexibility is possible because Backup Server and the database server communicate to determine when a backup should occur, the type of incremental backup to perform, and which backup options to change. Backup Server starts at the same time as the database server, and continues running until you either stop it or shut down the database server.

When performing full backups, Backup Server will copy the last full backup from the backup directory to the old directory. Then, it will copy all database files including journal files and **dmconfig.ini** to the backup directory, over writing the previous full backup.

When performing incremental backups, Backup Server will copy necessary journal files to the backup directory and you can set the time when you want to backup a database by keyword [DB\\_BkTim](#) or specify the backup time interval by keyword [DB\\_BkItrv](#).

If backup is success, you can see the file '**dmBackup.his**' which contain the complete backup history and .BB, .DB, .SBB, .SDB, .JNL files. Other more if you set the keyword [DB\\_BkOdr](#), it specifies the directories where the backup server puts the pervious version of full backup files. Once backup is fail, you can also get the information from the **dmBackup.his** and look up the dmconfig.ini to check whether the **DB\_BkSvr**, **DB\_BkDir** is right.

There are several options used to configure Backup Server. You can get the more detailed about related keyword for example [DB\\_BkSvr](#). These options control the filename format of the backup files, the location of the backup directory, the location of the old directory, the schedule Backup Server uses to perform backups, the amount a journal file must fill before Backup Server performs an incremental backup, and the way Backup Server saves backup files.

Backup Server also allows backup-related configuration settings to be made during the run time with the dmSQL SetSystemOption stored procedure.

## 2.5 Compressing Backup Files

Database files can become very large and a large amount of free space is required to store backup files. DBMaster now supports compressing backup files. You can set the keyword [DB\\_BKZIP](#) in the dmconfig.ini to enable or disable this feature.

- **DB\_BKZIP = 1:** Compresses the backup files
- **DB\_BKZIP = 0:** Backup files are not compressed (default)

The compression format is GZIP, so you can use any GZIP-compatible tool to read the compressed file.

**NOTE** *FO files are not compressed, even if you set DB\_BKZIP to enable compressing backup files.*

## 2.6 JServer Manager

JServer Manager has a wide variety of features that help you to effectively manage your database. Such as : Creating Databases, Starting and Shutting Down a Database, Deleting Databases, Backup Databases, Restore Databases, Integrated User Interface. In chapter 3 we will focus on Backup and Restore Databases.

### Backup Databases

- Perform full backups while the database is on-line or off-line
- Perform full backups to disk or to tape
- Perform incremental backups

### Restore Databases

- Restore crashed databases from disk or from tape

## 2.7 JConfiguration Tool

JConfiguration Tool spares users the task of searching for the keywords that represent these parameters. Instead, parameters are grouped into categories dependent on how they affect the database, and each parameter is given a descriptive tag to help users easily identify and understand how the parameter affects the database. Users can select backup options in JConfiguration Tool.

## 2.8 Key words in DMconfig.ini

When a user wants to backup a database, DBMaster must initialize several parameters to configure itself. These parameters are read from an ASCII text configuration file named **dmconfig.ini**. Chapter 5 lists the keywords and corresponding values that will be used by for

configuration. Since this file is in ASCII format, a DBA can edit it with a text editor to change the parameters as required.

## 2.9 JData Transfer Tool

The Data Transfer Tool provides a user-friendly interface for transferring data in and out of the database. It is another way to backup database. The tool performs the following types of data transformation:

- Importing from text
- Importing from XML
- Importing from ODBC
- Exporting to text
- Exporting to XML

Each type of data transformation is performed through a wizard. Each wizard guides the user through every step in the data transformation process, and gives descriptive information on the purpose of each step and the effect of different choices on the result. Consult chapter 6 to get the detail.

## 2.10 Load/Unload

The Load command is a tool provided by dmSQL, it is used to transfer a database object, already unloaded to a text file, into the database. There are seven options: load database, load table, load schema, load data, load project, load module, and load procedure. Only load the file that is unloaded in the same option. For example, load a database from the text file that is unloaded with database option.

Unload is a tool provided by dmSQL used to transfer the contents of a database to an external text file. After the unload procedure succeeds, dmSQL will produce two files. One stores the script, with extension name s0, to establish the database object and the other stores the BLOB data, with the extension name bn.

## 2.11 Dmsql command

The Export command facilitates the extraction of data from database tables and inserts the data into text files. There are two configurations used. The export command interface is used for specifying command options. The description file is used for specifying the export file format.

The Import command is used for extracting data from a text file and then inserting the data into database tables. The import command interface is used for specifying command options. The description file is used for specifying the import file format.

## 2.12 XTT/XTM

The XML transfer mapping (XTM) tool allows you to pass XML data to a database using XSL transformations. The XML Transfer mapping tool consists of three parts: an XML schema part,

which displays the schema of the XML file(s) that you are using as source data; an SQL database part, which displays the database tables; and an XTM part, which displays the mapping from the XML schema to the database tables.

The purpose of the XML Transfer Template (XTT) tool is to provide a customizable bridge between database data and XML documents. The bridge takes the form of a template file, the XML Transfer Template (XTT). The XTT file determines which database tables and columns to map to which XML elements and attributes. You determine the mapping using drag-and-drop operations in the XTT tool. The XTT tool ensures that XTT syntax is correct, and also aids in performing tasks such as generating schema documents (XSD) or document type definitions (DTD).

## 2.13 Rollover

This is a command-line tool for backup/restore. Under DBMaster 5.1\bin, you can see the file: rollover.exe.

The rollover is a text mode tool for backup-restore.

Please consult chapter 10 for details.

## 2.14 DMRestoreTB

This is a new tool for backup/restore in DBMaster 5.1. Under DBMaster 5.1\bin, you can see the files: dmrestortb.exe.

The DMRestoreTB is a command-line tool and we must setup environment and give it some parameters in advance.

Please consult chapter 11 for details.



## 3. Server Manager

The following sections briefly describe some of the features of JServer Manager that related to backup and restore a database. Please take a moment to review this chapter. Section 3.1 describes the ways JServer Manager Backup a database. Section 3.2 explains how to restore a database.

### 3.1 Backup a Database

DBMaster allows the use of different options for backup a database. In addition to periodic incremental or full backups made by the backup daemons, executing a backup at any time the database is on or off-line is supported.

On-line full backups increase the load on a system's processor and storage, and should only be done when client resource demands are low.

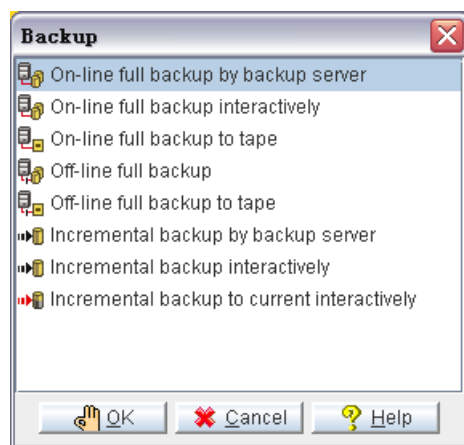


Figure 3-1 The Backup window

You can use the following backup methods.

- **On-Line Full Backup by Backup Server:** JServer Manager automatically backs up the database to the location specified in the dmconfig.ini file. All data in the database is copied to the backup location. The database must be started and the Backup Server activated. Clients may be connected to the database while on-line backup is being performed. File objects are also backed up when this method is used.
- **On-Line Full Backup Interactively:** The destination of backup files may be specified. File objects are not backed up.

- **On-line Full Backup to Tape:** All data in the database is copied to a single tape. File objects are not backed up.
- **Off-Line Full Backup:** All data in the database is copied to the backup location. The database must not have been started before off-line full backup is performed. File objects are not backed up.
- **Off-line Full Backup to Tape:** All data in the database is copied to a single tape. File objects are not backed up.
- **Incremental Backup By Backup Server:** All journal blocks are copied to the incremental backup location that is specified in the dmconfig.ini file. Incremental backups can be performed while the database is on-line and clients are connected to the database.
- **Incremental Backup Interactively:** All journal blocks are copied to an incremental backup location that may be specified at the time of the incremental backup. Incremental backups can be performed while the database is on-line and clients are connected to the database.
- **Incremental Backup to Current Interactively:** The data in the database is backed up from the point of the most recent backup to the end of the current journal file. The advantage of performing an incremental backup to current is that you can better protect the database against crashes.

## 3.1.1 ON-LINE FULL BACKUP BY BACKUP SERVER

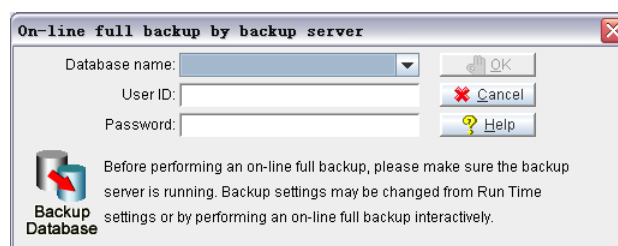
A full backup may be performed quickly and easily while the database is started using the On-line Full Backup by Backup Server. On-line full backups performed by this method are made to the location specified in the configuration file. The backup directory should be located on a disk separate from the disk the database is stored on to prevent loss of data in the case of media failure by the key word '**DB\_BkDir**'.

On-line full backup by backup server can be performed on a remote server, and may be used to perform file object backup; on-line full backup interactively is not capable of these functions.

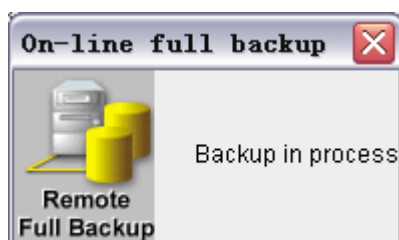
Be sure that the backup server has been activated before using this backup method by the key word '**DB\_BkSvr**'. If an error message "backup server doesn't exist" appears, shut the database down and activate the backup server when restarting. For instructions on starting the backup server, refer to '*JServer manager Tool User's Guide*' in chapter 4.2 and 4.3.

➡ To Perform an On-Line Full Backup by Backup Server:

1. Select **Backup Database** from the main console or the **Database** menu.
2. Select **On-line Full Backup by Backup Server** from the **Backup** window and click **OK**. The **On-line Full Backup by Backup Server** window will open.
3. Select a database from the **Database Name** menu. Enter a user name and password (must be a user with DBA authority or higher).



4. Click **OK**. The **On-Line Full Backup** message box will appear while the backup is in process.



5. The **On-line full backup** message box is replaced by a confirmation dialog box when the backup is complete. Any old backup files will be overwritten. If a directory for previous full backup files has been specified (this must be set from the **Backup** page of the **Start Database Advanced Settings** window), then the old backup files will be copied to the directory of previous backup files.

### 3.1.2 ON-LINE FULL BACKUP INTERACTIVELY

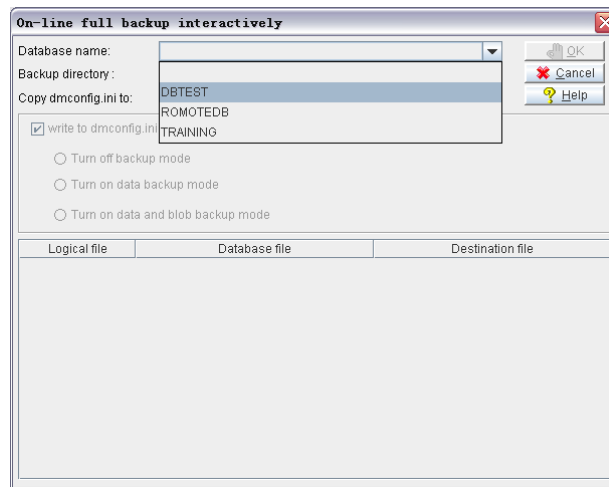
You can perform a backup to a database that has already been started. Performing an on-line backup involves specifying a location for the backup files. You should choose a backup directory location on a separate disk to minimize the risk of loss of data through media failure. You can also change the following incremental backup settings when making an on-line full backup.

BACKUP MODE	DESCRIPTION
Turn off backup mode	Disables the incremental backup daemon. When backup mode is disabled, journal files are not backed up.
Turn on data backup mode	All data is written to the journal but the incremental backup daemon only backs up non-BLOB data in the journal files.
Turn on data and BLOB backup mode	All data is written to the journal and the incremental backup daemon backs up all journal files.
Write to dmconfig.ini	Saves changes in the incremental backup mode to the <b>dmconfig.ini</b> file. The settings will be the same the next time the database is started.

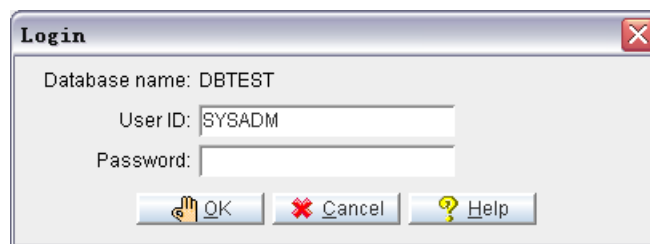
The incremental backups supplement the full backups

- To perform an On-line Full Backup Interactively:

1. Select **On-line Full Backup** from the **Backup** window. The **On-line Full Backup Interactively** window appears.
2. Select a database from the **Database Name** menu.



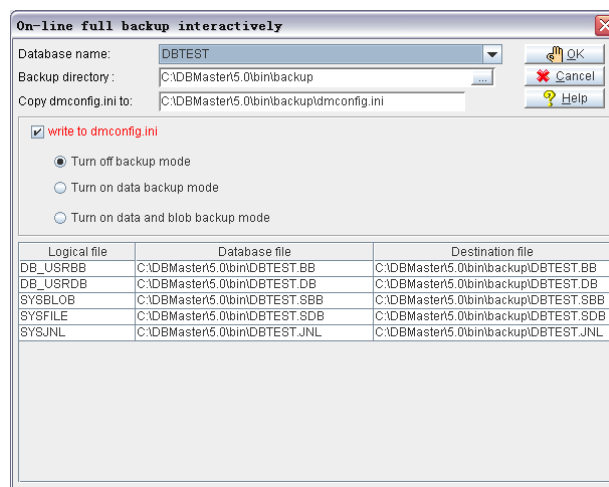
3. The Login dialog box is displayed.



4. Enter your user ID in the User ID field.



**NOTE** Any user with DBA security privilege may back up the database.

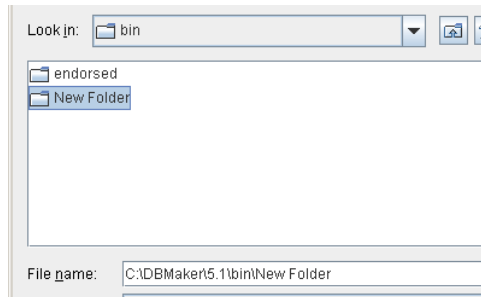
5. Enter a password in the Password field.
6. Click OK. A connection to the database is established. The **On-line Full Backup** window opens again with the list of operating system files to be backed up. The destination file location is the default backup directory specified in the configuration file, **dmconfig.ini**.



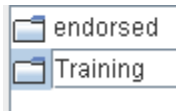
7. To select a new path for the backup directory.

- a) Click on the browse button . The **Select Path** window is displayed.

- b) The default database directory path is *DBMaker5.1\bin\*. A new directory can be created for the database. To create a new database directory, first use the *Up One Level* button  and/or the **Look in** menu to select the root for the database directory. Then click on the **Create New Directory**  button to make a directory called **New Folder**. The new folder will appear as follows and appears in the **File Name** text field.



- c) To change the directory name, type over the original name. Note that these changes are made directly to the operating system (i.e., use caution when changing the names of existing directories). After typing the new directory name it appears as follows:

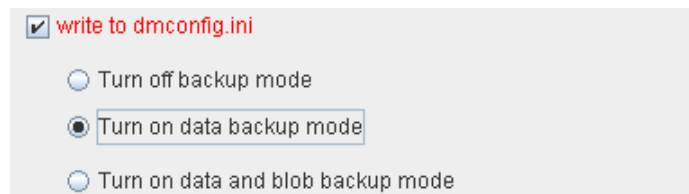


- d) Press the enter key to complete the creation of the new directory. The list will reappear in alphabetical order. Note that new name will appear in the **File name** field.



**NOTE** The name should appear in the directory list in a position according to alphabetical order.

8. Change the incremental backup settings.



- To turn on data backup mode, make sure that the **Turn on data backup mode** option button is selected.
  - To turn off backup mode, select the **Turn off backup mode** option button.
  - To turn on data and blob backup mode, select the **Turn on data and blob backup mode** option button.
9. Click OK to save all files to the backup directory. If files with the same name already exist in the backup directory, they will be overwritten. If a directory for previous full backup files has been

specified (this must be set from the Backup page of the Start Database Advanced Settings window), then the old backup files will be copied to the directory of previous backup file

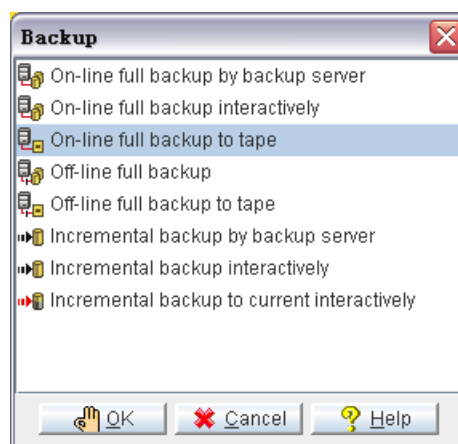
### 3.1.3 ON-LINE FULL BACKUP TO TAPE

When a database is started, you can perform an on-line full backup of your database files to a tape device. You can also change the following incremental backup settings when making an on-line full backup.

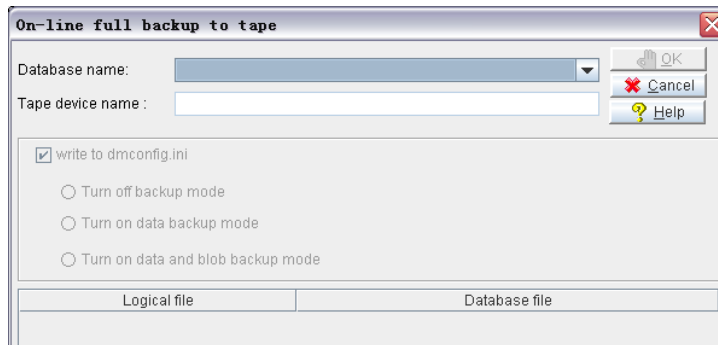
BACKUP MODE	DESCRIPTION
Turn off backup mode	Disables the incremental backup daemon. When backup mode is disabled, journal files are not backed up.
Turn on data backup mode	All data is written to the journal but the incremental backup daemon only backs up non-BLOB data in the journal files.
Turn on data and BLOB backup mode	All data is written to the journal and the incremental backup daemon backs up all journal files.
Write to dmconfig.ini	Saves changes in the incremental backup mode to the <b>dmconfig.ini</b> file. The settings will be the same the next time the database is started.

➤ To perform a full backup to tape:

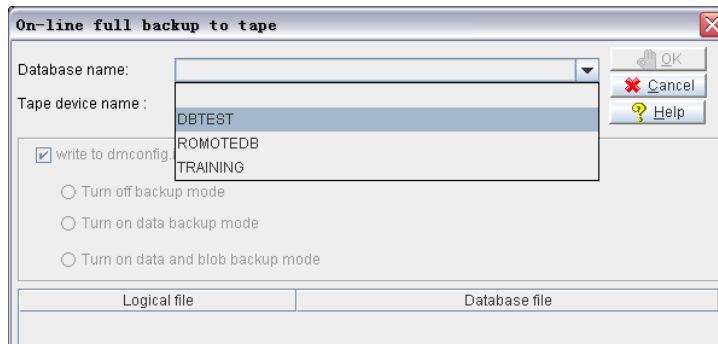
1. Select **Backup Database** from the main console. A list of different backup options is displayed.



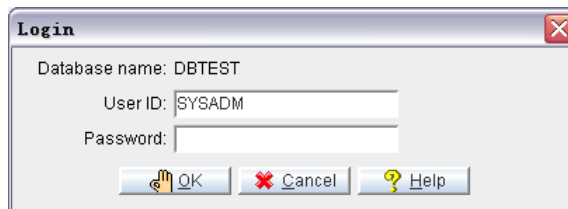
2. Select **On-line Full Backup to Tape** from the **Backup** window. The **On-line Full Backup to Tape** window opens.



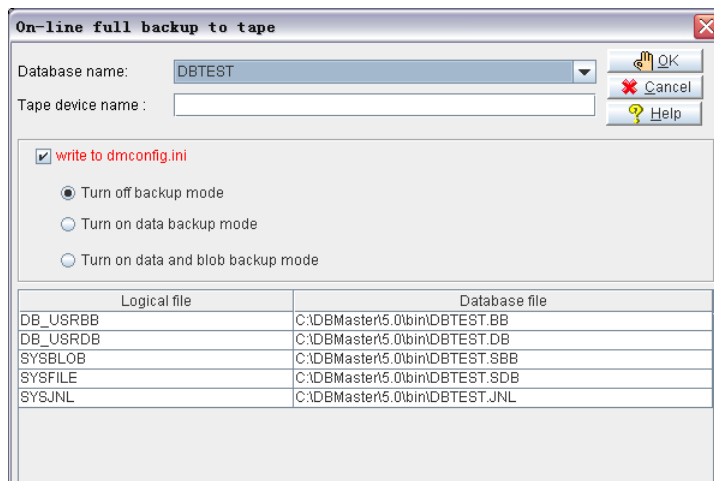
3. Select a database by clicking on the field next to **Database Name**. A drop-down list of databases available on the server will appear.



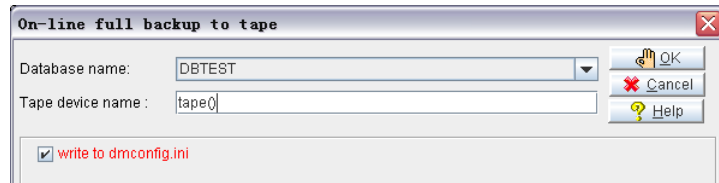
4. JServer manger will prompt you to log on to the database.



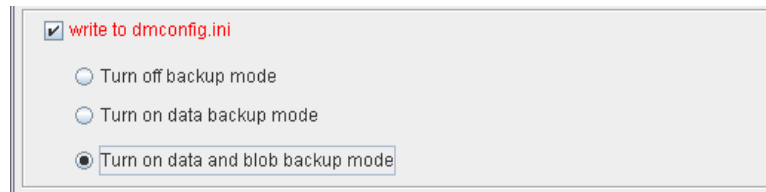
5. Enter a User ID and password into the appropriate fields.
6. Click **OK**. The **On-Line Full Backup to Tape** window will reopen.
7. All files to be backed up will appear in the **Database File** list.



8. Enter the path of the tape device in the **Tape Device Name** field.



9. Change the incremental backup settings:



- To turn on data backup mode, make sure that the **Turn on data backup mode** option button is selected.
- To turn off backup mode, select the **Turn off backup mode** option button.
- To turn on data and blob backup mode, select the **Turn on data and blob backup mode** option button.

10. Click **OK**. The database will be copied to tape.

### 3.1.4 OFF-LINE FULL BACKUP

Offline full backups use operating system commands to back up the database. DBMaster provides this option, however, backup server is recommended. Offline full backups necessitate the database be shut down, furthermore, managing the backup sequence is a more complex process.

To perform an offline full backup, you must have read permission on the database files from the operating system, and write permission on the backup directory from the operating system. If you have to shut down the database first, you must have DBA or SYSADM security privileges.

You can perform an offline full backup regardless of the backup mode; the database may be running in NON-BACKUP, BACKUP-DATA, or BACKUP-DATA-AND-BLOB mode (in chapter 5.12 DB\_BMode). Using an offline full backup, you can restore the database to the point in time the database was shut down.

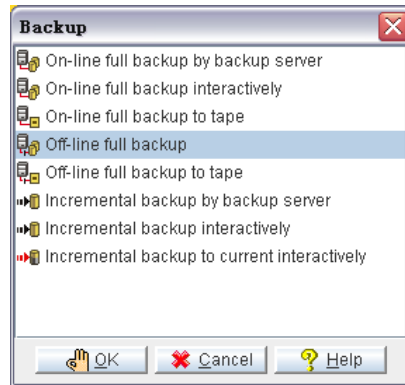
Note the follow section that offline full backup using JServer Manager does not back up file objects. File objects must be copied manually. Be sure to exactly replicate the file and directory structure if restoring a database from an offline full backup.

You can perform a backup to a database that is not yet started. Performing an off-line backup involves specifying a location for the backup files. You should choose a backup directory location on a separate disk to minimize the risk of loss of data through media failure.

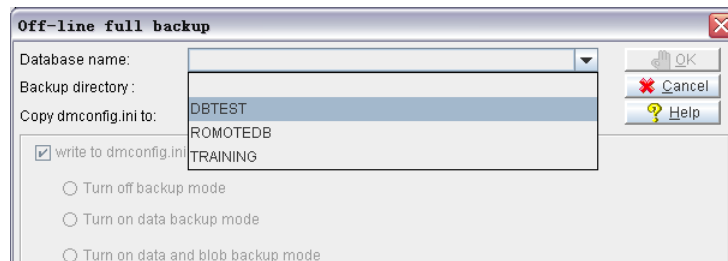
- To perform an Off-line Full Backup:



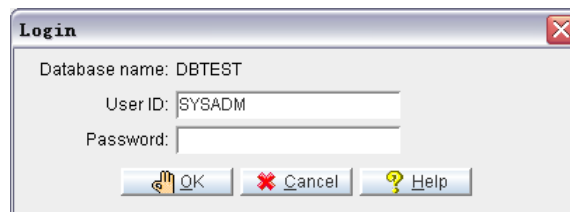
1. Select **Backup Database** from the main console. A list of different backup options is displayed.



2. Select **Off-line Full Backup** from the **Backup** window. The **Off-line Full Backup** window appears.
3. Select a database from the Database Name menu.



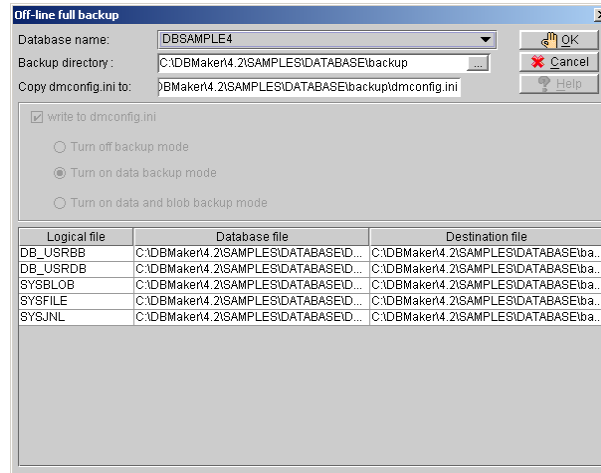
4. JServer Manager will prompt you to log onto the database.



5. Enter your User-ID in the User ID field.

**NOTE** Any user with the DBA security privilege can back up the database.

6. Enter a password in the Password field.
7. Click OK. A single user connection is established to the database. The **Off-line Full Backup** window is displayed with a list of operating system files to be backed up. The destination file location is the default backup directory specified in the configuration file, **dmconfig.ini**.



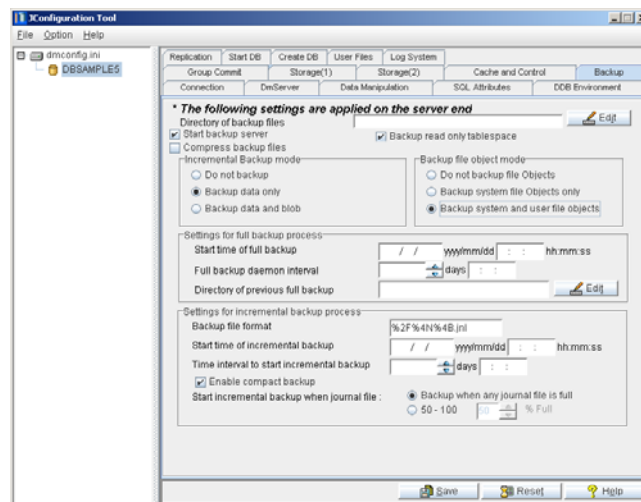
8. To select a new path for the backup directory, Please refer to chapter 3.1.2 for more information about how to create a new backup directory:

## Offline Full Backup using dmSQL

- ➔ To perform an offline full backup using dmSQL:
  1. Notify all users that the database will be shut down at a specified time and ask them to disconnect before that time.
  2. If the database is running, shut it down using the TERMINATE DB command. If there are any errors while shutting down the database, restart the database, correct the problem, and shut it down again.
  3. Examine the **dmconfig.ini** file and list all relevant files and directories, including the file object directory, which require backup.
  4. Use operating system commands or utilities to copy the database files, including data files, journal files, file objects, and the **dmconfig.ini** file, to the backup directory or backup device.

## Backup schedule with JConfiguration

You can use the JConfiguration Tool to set the backup schedule yourself.



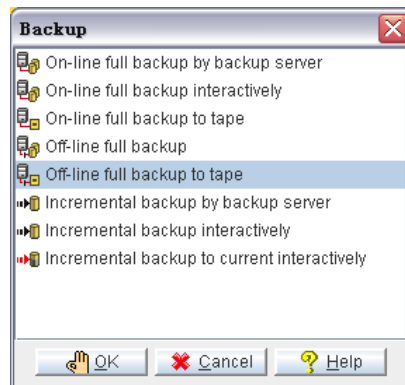
You can choose the way you need, such as backup data only or backup data blob, and then fill the setting in blank depend on the situation. The related configuration keywords are betrayed in chapter 5.

### 3.1.5 OFF-LINE FULL BACKUP TO TAPE

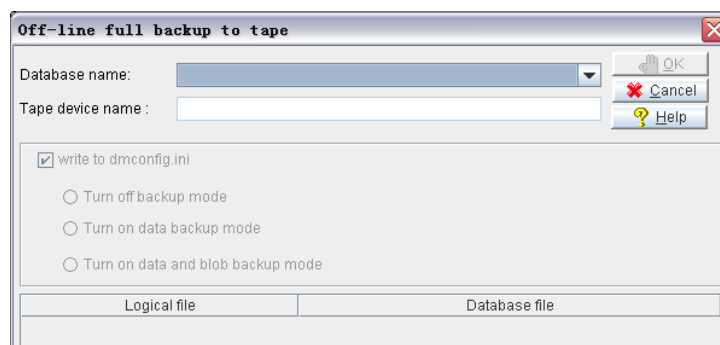
As well as backup your database to another file location, you can back it up to tape or tapes. Performing an off-line backup involves specifying a location for the backup files. Performing backups to tape minimizes the risk of loss of data through media failure.

➔ To perform an off-line full backup to tape:

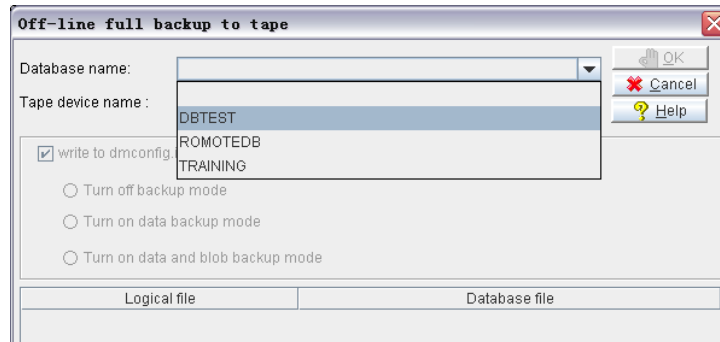
1. Select **Backup Database** from the main console. A list of different backup options is displayed.



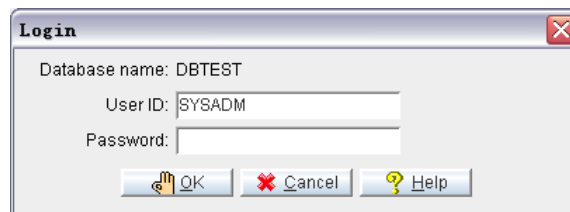
2. Select **Off-line Full Backup** to Tape from the **Backup** window. The **Off-line Full Backup** to Tape window opens.



3. Select a database from the Database Name menu.



4. The Login window is displayed.

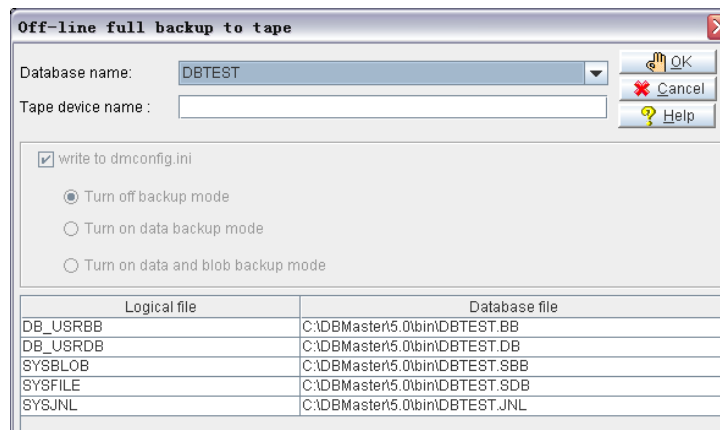


5. Enter your user ID in the User ID field.

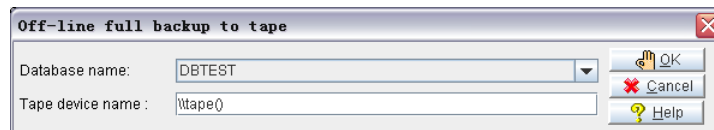
**NOTE** Any user with DBA security privilege may back up the database.

6. Enter your password in the Password field.

7. Click OK. The **Off-line Full Backup to Tape** window opens again with the list of operating system files to be backed up.



8. Enter the device name in the Tape Device Name field.



9. Click OK to execute the full backup.

## 3.1.6 INCREMENTAL BACKUP BY BACKUP SERVER

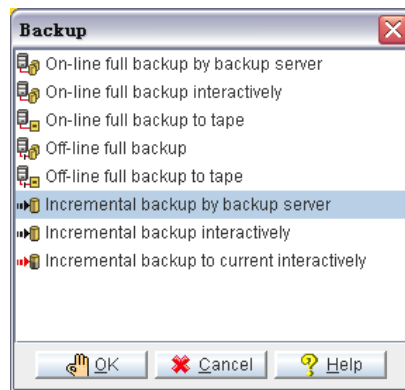
Incremental backups differ from full backups in that they only copy full journal files to the backup destination directory. To allow a database to recover its files, it is necessary to perform a full

backup before an incremental backup. It refers to keyword **DB\_BMode** in chapter 5.12, users must select a mode before backup database.

An incremental backup may be performed quickly and easily while the database is started using the On-line Incremental Backup by Backup Server. Incremental backups performed by this method are made to the location specified in the configuration file. The backup directory should be located on a disk separate from the disk the database is stored on to prevent loss of data in the case of media failure.

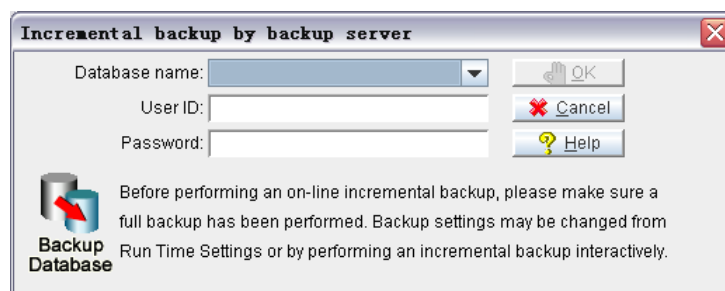
➔ To Perform an Incremental Backup by Backup Server:

1. Select **Backup Database** from the main console or the **Database** menu. The Backup window will appear.

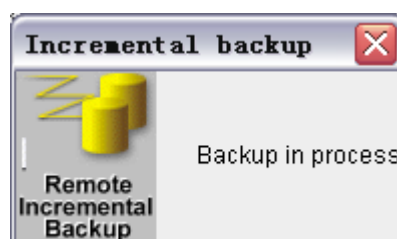


**NOTE** Be sure that the backup server has been activated before using this backup method. If an error message “backup server doesn’t exist” appears, shut the database down and activate the backup server when restarting. For instructions on starting the backup server, refer to sections 4.2 and 4.3 in ‘Server Manager User’s Guide’.

2. Select **Incremental Backup by Backup Server** from the **Backup** window and click **OK**. The **Incremental Backup by Backup Server** window will open.
3. Select a database from the **Database Name** menu. Enter a user name and password (must be a user with DBA authority or higher).



4. Click **OK**. The **Incremental backup** message box will appear while the backup is in process.



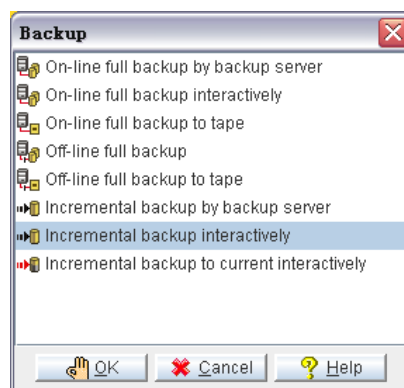
- The **Incremental backup** message box is replaced by a confirmation dialog box when the backup is complete.

## 3.1.7 INCREMENTAL BACKUP INTERACTIVELY

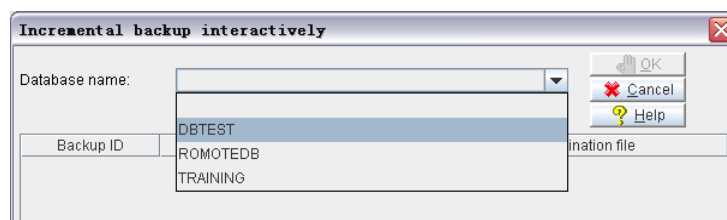
Incremental backups differ from full backups in that they only copy full journal files to the backup destination directory. To allow a database to recover its files, it is necessary to perform a full backup before an incremental backup then when you have changed objects in database you can use incremental Backup after you start the database. The incremental backup daemon can be set to automatically copy journal files when they have been filled to a set capacity. In this way, it handles all the journal files itself and makes sure that the required data is backed up. It is possible to change the destination file location for manually performed incremental backups (See 见下方), but not recommended. Backup journal files are stored in a location indicated on the Backup page of the Advanced Settings windows, and ideally should be stored in the same directory as the full backup. If you have not started the backup server, shut the database down and restart it with this setting enabled. For more information, refer to Start Backup Server of *JServer Manager Tool User's Guide*, or refer to the '*Database Administrators Guide*'.

➔ To perform an incremental backup:

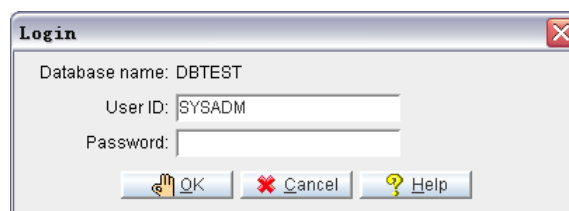
- Select Backup Database from the main console. A list of different backup options is displayed.



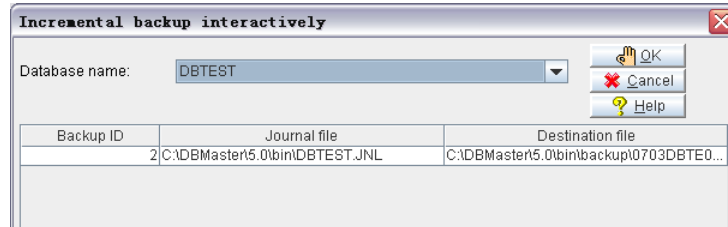
- Select **Incremental Backup** from the **Backup** window. The **Incremental Backup** window appears.
- Select a database from the Database Name menu.



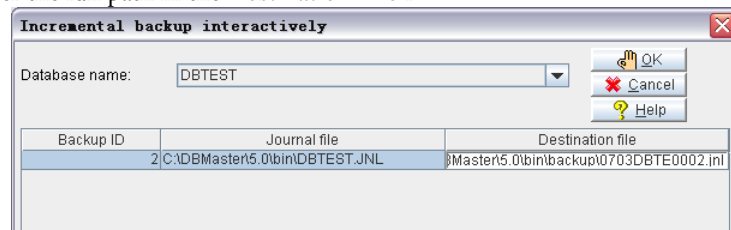
- The Login window appears.



5. Enter your user ID in the User ID field.
6. Enter your password in the Password field.
7. Click OK. A single-user connection is established to the database. The Incremental Backup Interactively window displayed displaying the journal file and destination file locations.



8. To specify a different location for the Destination File:
  - e) Click on the **Destination File** field.
  - f) Enter the full path in the Destination File f



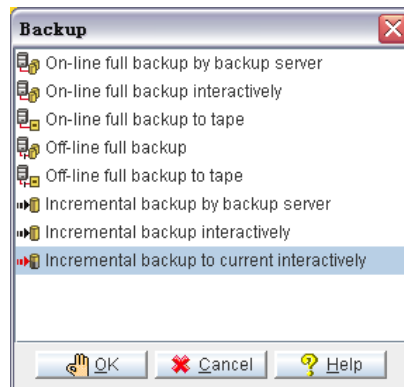
- g) Press ENTER to ensure that the new destination file path is selected.
9. Click **OK** to execute the incremental backup.

### 3.1.8 INCREMENTAL BACKUP TO CURRENT INTERACTIVELY

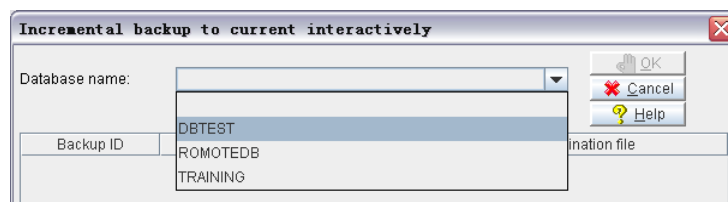
Incremental backup to current copies all journal files to the backup directory, including the journal file that is currently being used. Incremental backups differ from full backups in that they only copy full journal files to the backup destination directory. To allow your database to recover its files, perform a full backup before an incremental backup. The incremental backup daemon can be set to automatically copy journal files when they have been filled to a set capacity. In this way, it handles all the journal files itself and makes sure that the required data is backed up. It is possible to change the destination file location for manually performed incremental backups, but not recommended. Backup journal files are stored in a location indicated on the Backup page of the Advanced Settings windows, and optimally are stored in the same directory as the full backup. For more information, refer to Start Backup Server in '*JServer Manager Tool User's Guide*', or refer to the *Database Administrators' Guide*.

To perform an incremental backup to the current journal file:

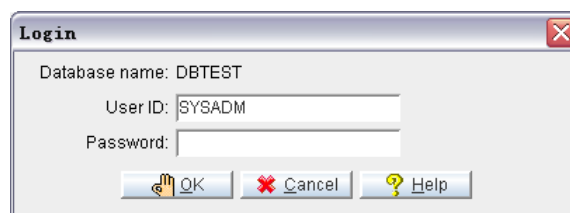
1. Select **Backup Database** from the main console. A list of different backup options is displayed.



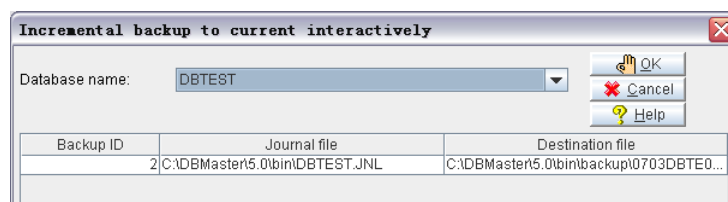
2. Select **Incremental Backup to Current Interactively** from the **Backup** window. The **Incremental Backup to Current Interactively** window is displayed.
3. Select a database from the **Database Name** menu.



4. The Login window appears.

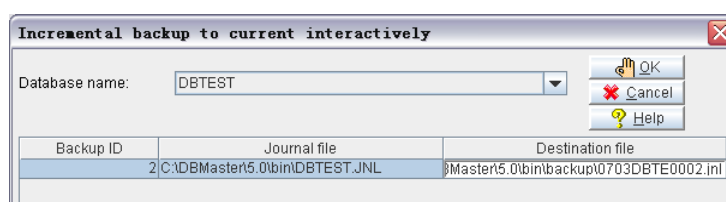


5. Enter your user ID in the **User ID** field.
6. Enter your password in the **Password** field.
7. Click **OK**. A single-user connection is established to the database. The Incremental Backup to Current Interactively window opens displaying the journal file and destination file locations.



8. To enter a new location of the database file from the default, choose a location.

- h) Click the **Destination File** field.
- i) Enter a destination file path in the Destination File field.





- j) Press ENTER to ensure that the new destination file path is selected.
9. Click **OK** to execute the incremental backup.

### **3.1.9 MULTIPLE TAPE BACKUP**

---

Originally DBMaster supports tape backup for only one tape. The tape backup is operated through JServer Manager GUI. However, as time goes by, the request for large database support has been increased and often users need to have backup supports for more than on tape. DBMaster provide an API to JServer Manager not for users. Note that the same as before, for tape, only full backup will be supported.

The newly added APIs will let users do the following things.

- Eject Tape: When backup is full in one tape, allow users to unload the tape.
- Change Tape: After ejecting tape, users load another tape to continue tape backup/restore. When used in restoring data from tape, the change tape utility function is able to check whether the loaded tape is the correct tape.

This function is used after a tape is full when backing up files into tapes. During the time when backup operation is running, if the tape is full, the tape will be ejected and another tape will be loaded. This function will rewind the loaded replacing tape.

This function is used after a tape is restored into file and there are still tapes to be restored. During the time when restore operation is running, if the content of a tape is all written out to a file and there are still more data to be restored, the tape will be ejected and another tape will be loaded. This function will rewind the loaded replacing tape.

Call this function to load restore tape. If the user put wrong tape in tape drive, this function will return error and the tapeno argument will be assign the correct number of tape to be restored.

Users should be allowed to change tape if wrong tape is loaded into tape drive. If wrong tape is loaded into tape drive, the interface for tape restore should tell users to load correct tape. Also, the correct tape number must be returned.

## 3.2 Restore a Database

The database administrator may find it necessary to restore all data from backup files if an unrecoverable error has occurred in the database. You can perform a database restoration from disk or from tape. Restore a database from tape will recreate the database as it existed at the time of the most recent full backup. If you restore the database from disk, you can restore the database to the time of the last incremental backup.

**NOTE:** For more information on database restoration, refer to the '*Database Administrators Guide*'.

The following subsections give procedures for both restoration methods.

### 3.2.1 RESTORE A DATABASE FROM DISK

You can restore a database that has been backed up to another disk location on your computer or on the network.

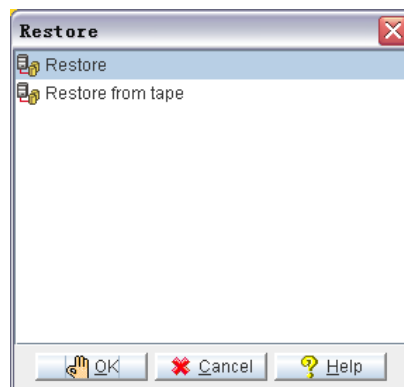
First you must use the on line full backup in JServer Manager before the database has some changes, then backup the database by incremental before restore the database. User must set key words at first of all: **DB\_BMODE**, **DB\_BKSVR**, **DB\_BKFOM** (if there are file objects need to be backup).

**NOTE:** To restore a database you must have DBA privilege.

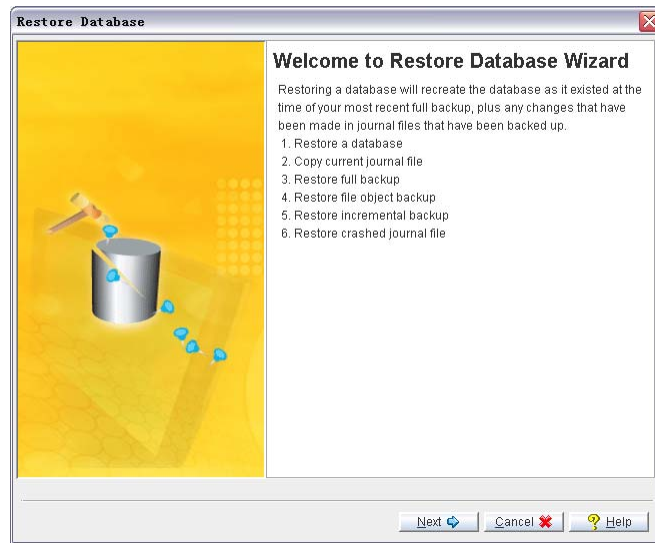
➔ To restore a database from disk:

1. Select Restore Database from the main console.

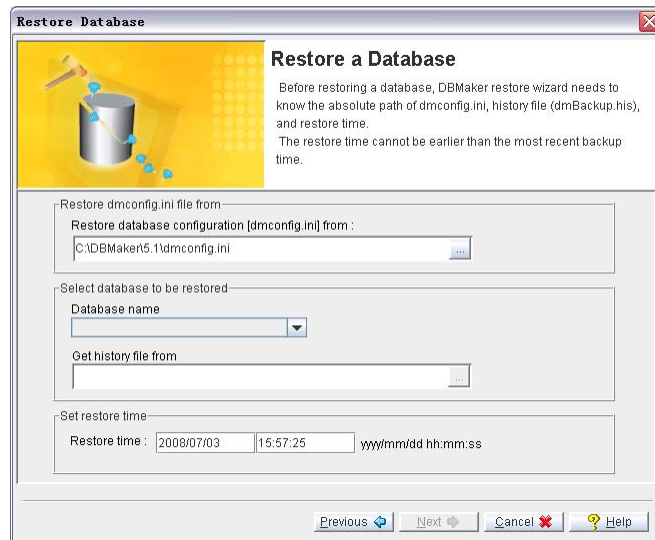
**NOTE** You can also select *Restore Database* from the *Database* menu.



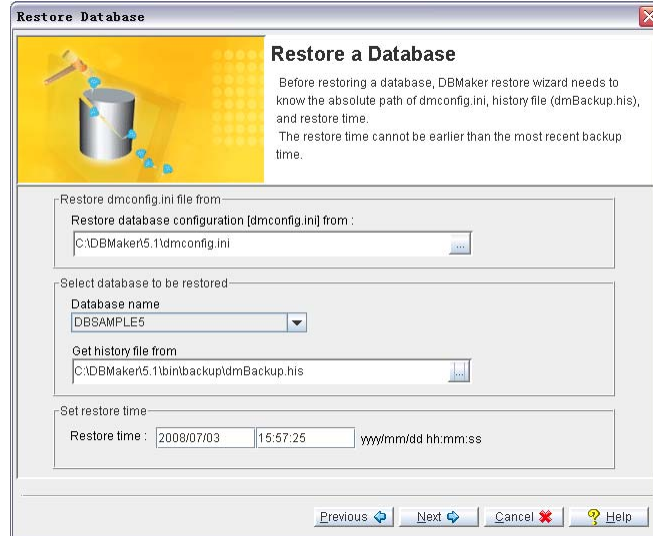
2. Make sure that Restore is selected from the Restore window.
3. Click OK. The Restore Database window is displayed.




4. Click Next. The Restore Wizard window is displayed.



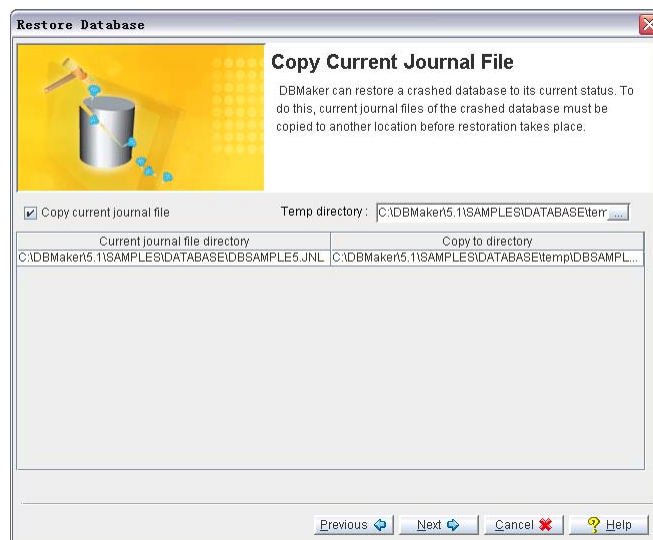
5. Select a database by clicking on the field under **Database Name**. A drop-down list of databases available on the server appears.
6. Select a database. The Restore Database Configuration [dmconfig.ini] from field shows the default location of the configuration file. If you moved the dmconfig.ini



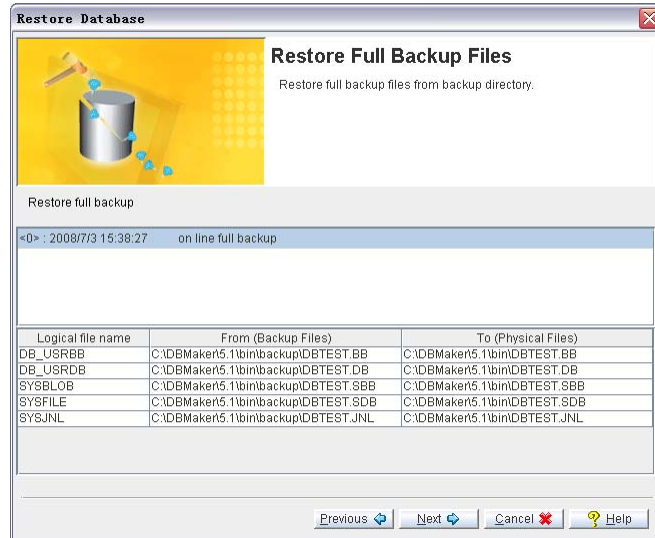
7. If you moved the backup history file to a new location, then enter the new path or click on the browse button  in the Get History File From field.

**NOTE** After the database to be restored has been chosen, the location of the history log file should appear in the Get History File From field.

8. Click Next. The Copy Current Journal File page is displayed. The current journal file is then copied to a temporary directory so that the database can be restored to the condition it was in just before shutdown or failure. If the current storage media is unstable, you can specify another location for storing the current journal file in the Copy to Directory column.



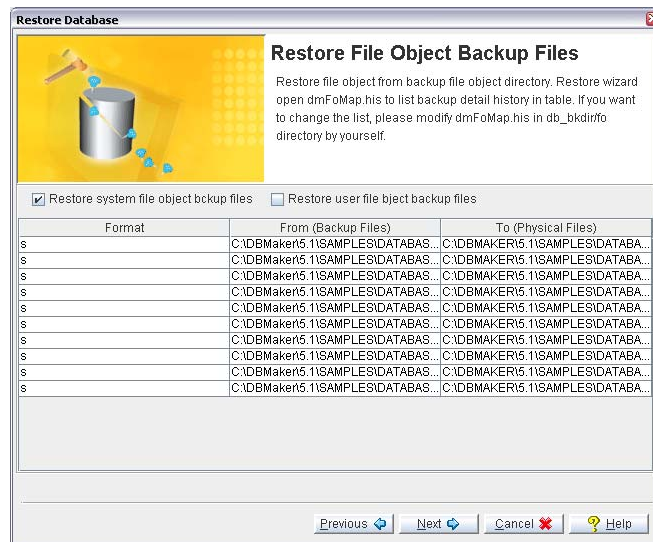
9. Make sure that the correct temporary directory location for the current journal file is displayed in the Temp Directory field.
- 10 Select a backup that contains the files you wish to restore from the top list. The logical file names, backup files, and physical files are displayed in the bottom list.



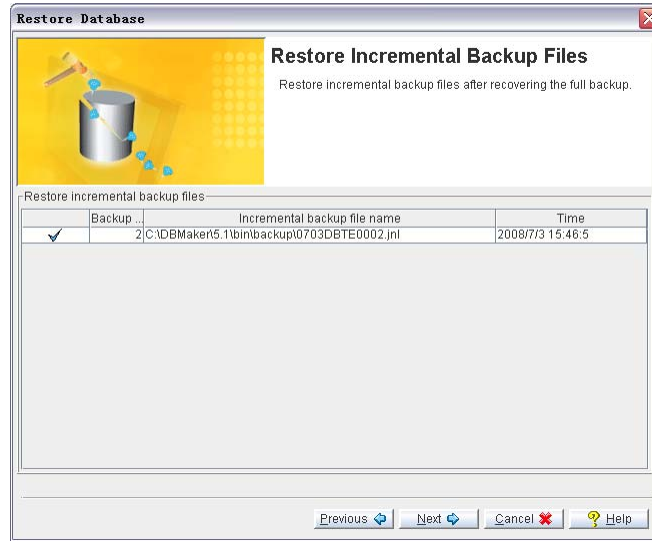
11. Click Next. The Restore File Object Files window will open.

**NOTE** *The Restore File Object Files window only opens if file objects were previously backed up. Otherwise, the Restore Incremental Backup Files page will open*

12. Select whether to restore system file objects or both system and user file objects by checking the appropriate box. File objects that will be restored appear black in the list; file objects that will not be restored appear gray.

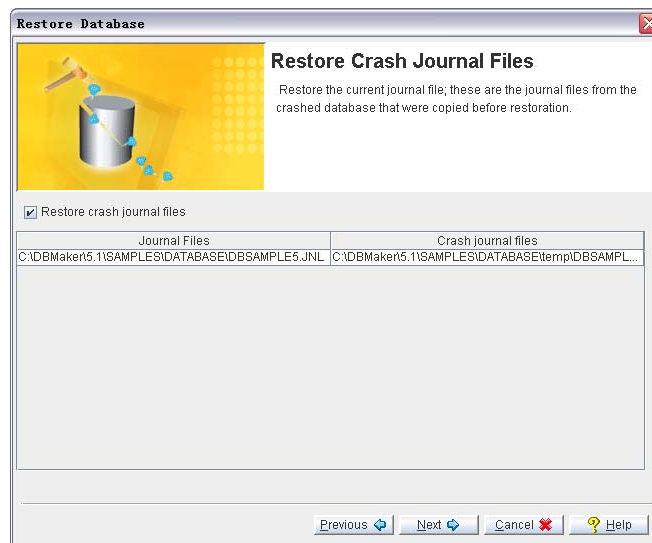


13. Click Next. The Restore Incremental Backup Files page will open.

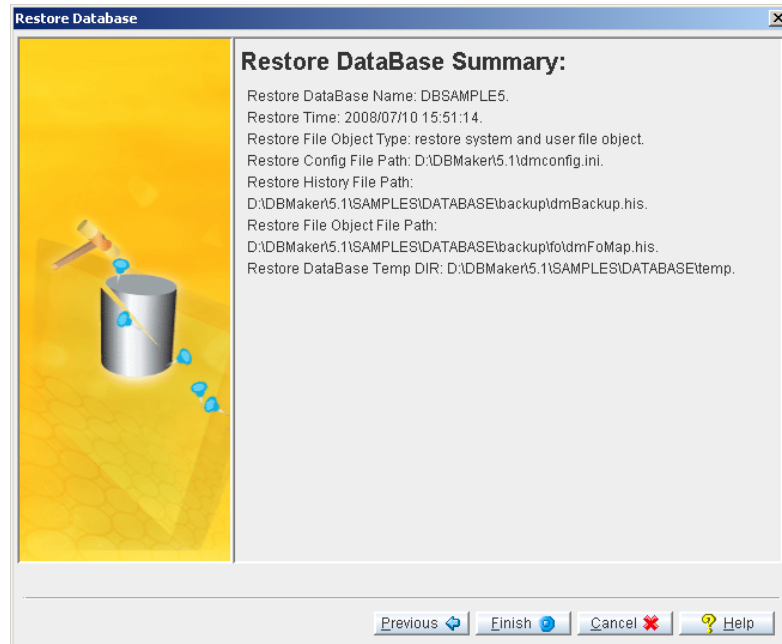


14. The Incremental Backup File Name field shows all the incremental backup files. If you need to change the incremental backup file path, enter a path in the Incremental Backup File Name field. You can edit the file path but must not skip restoration of any of the files.

15. Click Next.



16. If you need to restore journal files created after the last incremental backup was made to the database, make sure that the Restore Crash Journal Files check box is selected. Clearing this check box will prevent you from restore the journal files after the last incremental backup.
17. Click **Next** button, the **Restore Database Summary** window appears, from this window, user can easily browse the restoration informations. If all the information is satisfying, click the **Finish** button to accomplish the restoration.



**NOTE** *The current journal files that were copied into a temporary directory are used to restore the database to its condition just prior to shutdown.*

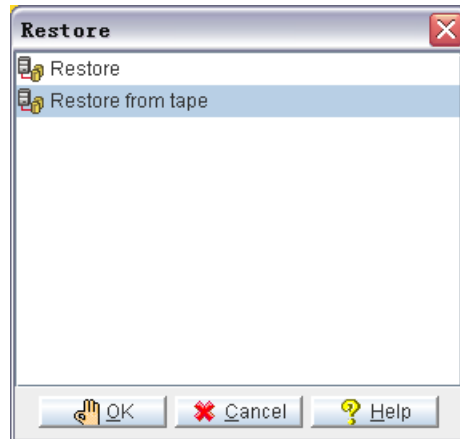
18. Click Finish. Restoration is complete.

### 3.2.2 RESTORE A DATABASE FROM TAPE

You can restore a database that has been backed up to a tape device. You may choose to restore the backup history log and **dmconfig.ini** file from tape. Restore the backup history log from tape will overwrite the current history log. All records of incremental backups made since the last full restoration to tape will be lost. It will not be possible to restore the database to a more recent status using the incremental files after the backup history log is copied from tape. To restore a database, you must have DBA authority or higher.

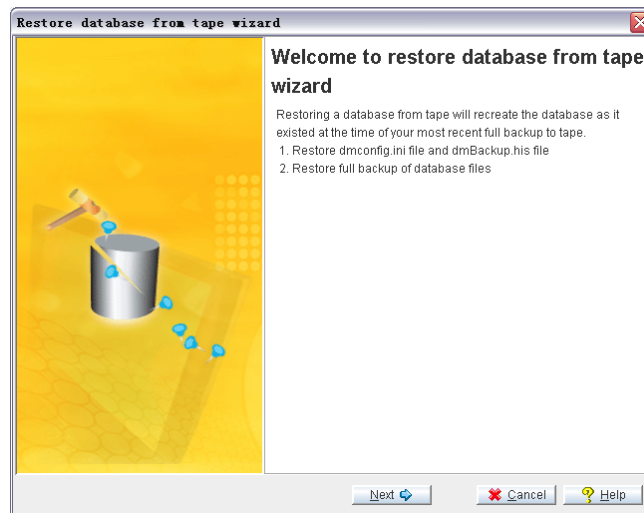
- To restore a database from tape:

1. Select Restore Database from the main console. The Restore window is opened.



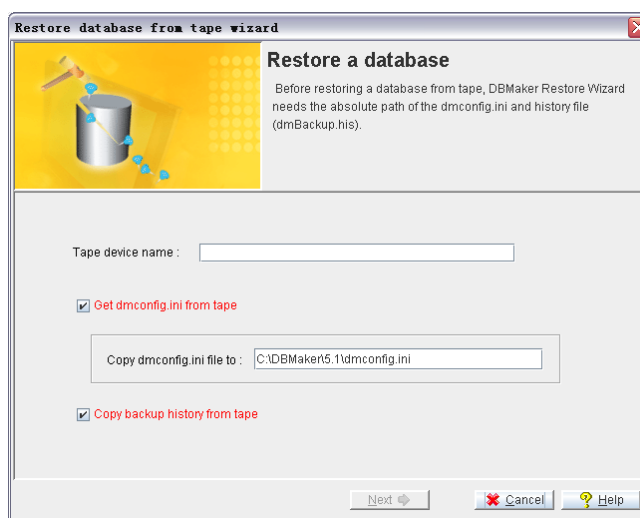
**NOTE** You can also select Restore Database from the drop-down menu. The Restore window opens.

2. Select Restore from Tape from the Restore window and click OK. The Restore Database from Tape Wizard starts.





3. Click Next. The Restore a Database page is displayed.
4. Enter the name of the tape device in the Tape Device Name field.
5. Insert the tape from which the database is to be restored.



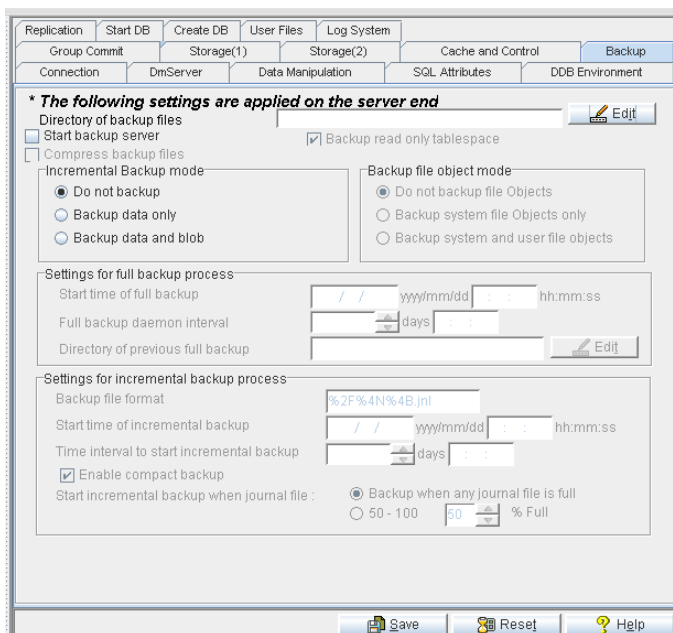
6. To get the **dmconfig.ini** file from tape, make sure that the Get dmconfig.ini from Tape check box is selected.
7. To get the **dmconfig.ini** file from disk, clear the check box.
8. To copy the **dmconfig.ini** file to different location, enter it in the Copy dmconfig.ini File to field.
9. To copy the backup history file from tape, make sure that the Copy Backup History from Tape check box is selected.
10. To copy the Backup History file from another source, clear the Copy Backup History from Tape check box.
11. Click Next. The name of the database and a list of the files in the database appear. The logical files should map to physical destination locations.
12. Click the Finish button to restore the database.
13. Related Topics: Restore a Database from Tape

## 4. JConfiguration Tool

Users can use the JConfiguration Tool which is a GUI Tool to select backup options as a DBA need. The descriptive tags and fields in JConfiguration tool represent the keywords in the configuration file. It helps users easily identify and understand how the parameter affects the database, users can set the time when to backup the database, and then DBMaster will execute automatically.

Selecting the Backup tab opens the Backup page, which displays the available incremental backup and full backup options. The settings on this page determine the actions of the backup server daemon and specify the location to store backup files. The settings on this page are only applied to the server.

For more in-depth information on backup modes and procedures, refer to the *Database Administrator's Guide*.



### 4.1.1 INCREMENTAL BACKUP MODE SETTINGS

The settings in the **Incremental Backup Mode** field specify the incremental backup mode for a database. The Backup Server must be started in order for modes other than Do not backup to function. Selecting 'Do not backup enables NON-BACKUP mode', which causes the journal file's oldest records to be overwritten when the journal file is full. Selecting Backup Data Only enables BACKUP-DATA mode, which allows for a full recovery from an instance failure, and full recovery of non-BLOB data in a media failure. Selecting Backup Data and BLOB enables BACKUP-DATA-AND-BLOB mode, which allows for full recovery of all data. These settings apply only to the

operation of journal files, and therefore to the operation of the incremental backup process. Selecting Backup Data Only or Backup Data and BLOB will allow the user to access settings in the Settings for Incremental Backup Process field. These settings must be configured properly for the incremental backup process to work (see *Settings for Incremental Backup Process*). For more in-depth information on backup modes, refer to Chapter 14 of the *Database Administrator's Guide*. This setting corresponds to the [DB\\_BMode](#) keyword in the **dmconfig.ini** file. The default mode is *No Backup*.

### 4.1.2 BACKUP FILE OBJECT MODE

---

The settings under the **Backup File Object Mode** effect how file objects are copied during the full backup process. Selecting **Do Not Backup File Objects** disables file backup during the full backup process. Selecting **Backup System File Objects Only** will result in system file objects being backed up during automatic full backups. Selecting **Backup System and User File Objects** will result in both system file objects and user file objects being copied to the backup directory during automatic full backups. This setting corresponds to the [DB\\_BkFoM](#) keyword in the **dmconfig.ini** file. The default mode is *No Backup*.

### 4.1.3 SETTING THE BACKUP FILE DIRECTORY

---

The Directory of Backup Files field shows the directory where the backup server puts all full backup files and incremental backup (journal) files. You should create a backup directory on a different disk from the database files to prevent the loss of both the database and the backup files in the event of a media error. Multiple backup paths can be specified. The default path for backup files is *(Database Directory)\Backup* and is automatically created by DBMaster. The total length of any backup directory path must not exceed 256 characters in length. Users should enter a new path for the backup file directory by typing the new path into the text field, or by clicking on *Edit* button next to the text field. This setting corresponds to the [DB\\_BkDir](#) keyword in the **dmconfig.ini** file. When setting multiple backup paths you should keep the following in mind:

When a database system backs up files, it will try to store files in the paths one by one for each file. For example, if you want to store a file to backup directory DIR\_1 but DIR\_1 does not have enough space to store the file, then DBMaster will try to store it to backup DIR\_2 or DIR\_3 and so on. Files are directed to backup paths by file size. For example, two directories exist dir1 and dir2. The file sizes are as follows db\_bkdir = dir1 33 dir2 44. Backup files larger than 132K[33\*4] are backed up into dir2.

Only the first backup directory can accept slave site backup files.

File objects must be backed up in the first backup directory.

The maximum number of backup paths that can be specified are 32.

**NOTE** *The backup directory file size is calculated as follows: The file size number multiplied by 4K. For example setting the file size to 44 means the physical file size is 44 x 4K = 176K.*

### 4.1.4 STARTING BACKUP SERVER

---

Enabling the Start Backup Server check box activates the full backup server daemon. All data, system, and journal files are periodically copied directly to the backup directory when the backup server daemon is activated. Users also can access settings in the Settings for Full Backup Process field. These settings must be configured properly for the full backup process to work. The

default setting is *disabled*. This setting corresponds to the [DB BkSvr](#) keyword in the `dmconfig.ini` file.

#### 4.1.5 ENABLING COMPRESS BACKUP FILES

---

This is a new function in DBMaster 5.1. Enabling the Enable compress backup file check box activates compress backup files. When we full-backup the database files we can compress the backup files that may reduce the requirement of free space. When BkServer/ JServer Manager want to compress the full-backup files, we call this function to compress the files. The default setting is *disabled*. This setting corresponds to the [DB BKZIP](#) keyword in the `dmconfig.ini` file.

#### 4.1.6 ENABLING BACKUP READ ONLY TABLESPACE

---

Enabling the check box next to the Enable backup read only tablespace permits BKSERVER always to backup read-only tablespace files. Disabled the check box, the BKSERVER will not backup the read-only tablespace files. The default setting is enabled. This setting corresponds to the [DB BKRTS](#) keyword in the `dmconfig.ini` file.

#### 4.1.7 SETTINGS FOR FULL BACKUP PROCESS

---

The following settings configure the full backup process and must be set for the backup daemon to function properly.

##### Setting the Start Time for Full Backups

To set the time at which the first full backup will start being processed for the database, enter the date in the yyyy/mm/dd field and the time in the hh:mm:ss field. Reenter the numbers if they appear incorrectly the first time; JConfiguration Tool automatically enters values into the first two spaces of the yyyy field and the first space of the hh field if values are entered into other fields. This setting corresponds to the [DB FBkTm](#) keyword in the `dmconfig.ini` file. There is no default value.

**NOTE** *The full backup is processed only if the full backup start time is set.*

##### Setting the Full Backup Daemon Interval

The number in the combo box specifies the time interval at which the Full Backup Daemon (See *Starting Backup Server*) is activated in days. Next to the combo box is a field for time input, which specifies the time interval in hours, minutes, and seconds. The total time interval is determined by adding the two values together, so inputting 1 into the days field and inputting 12:00:00 into the hh:mm:ss field would cause the Full Backup Daemon to be activated every day and a half. The number of days can be manually entered into the combo box, or increased or decreased by clicking the arrows to the right; the number of hours, minutes, and seconds are manually entered. This setting corresponds to the [DB FBkTv](#) keyword in the `dmconfig.ini` file.

##### Specifying the Previous Full Backup Directory

All old backup data residing in the backup directory at the time of backup is rewritten to the Directory of Previous Full Backup. This previous full backup directory corresponds to the [DB BkOdr](#) keyword. Old backup information already in the Directory of Previous Full Backup will be overwritten in the instance of a full backup unless the directory name is changed or the old backup file names are changed. As with the Directory of Backup Files, the Directory of Previous

Full Backup should reside on a disk or system separate from the *错误! 未找到引用源。* to ensure that data can be recovered in the event of a media failure. The total length of any backup directory path must not exceed 256 characters in length. Users must enter a path for the backup file directory by typing the path into the text field, or by clicking on *Edit* button next to the text field. DBMaster will not copy the previous full backup files if no Directory of Previous Full Backup is specified. This setting corresponds to the [DB\\_BkDir](#) keyword in the **dmconfig.ini** file.

When setting multiple backup paths you should keep the following in mind:

When database system backs up files, it will try to store files in the paths one by one for each file. For example, if you want to store a file to backup directory DIR\_1 but DIR\_1 does not have enough space to store the file, then DBMaster will try to store it to backup DIR\_2 or DIR\_3 and so on. If all backup directories are full, an error message will be returned stating that all backup directories are full.

Only one backup directory at a time is used to backup files in the slave site.

File objects must be backed up in the first backup directory.

The maximum number of backup paths that can be specified are 32. This is defined in the **MAX\_BKPATH** keyword in **dmconfig.ini** file.

#### 4.1.8 SETTINGS FOR INCREMENTAL BACKUP PROCESS

---

The following settings become available only when **Start Backup Server** has been enabled and the user has selected Backup Data Only or Backup Data and BLOB. The following settings are needed for the Incremental Backup Process to function properly.

##### Setting the Journal Backup File Format

The backup filename format allows you to specify the format Backup Server will use to name incremental backup files. The backup filename format may include both text constants and must include format sequences (escape sequences) that represent special character strings.

An incremental backup file name must consist of at least three special character strings: the full backup id, the database name, and the backup identification number. Backup Server assigns a full backup ID when naming incremental files in a backup sequence. When restoring a database, DBMaster uses the full backup ID to correctly recreate the backup sequence. The database name correctly identifies the database to which an incremental backup file belongs. The backup identification number identifies the relative position of the incremental backup file in the backup sequence.

Format sequences have three parts: the escape character, the length value, and the format character. Valid format sequences are:

**%[x]F**—The full backup ID. The variable *x* may have values 1-4 where the values represent the following formats;

1: full backup id shown as YYYYMMDD, e.g. 20010917

2: full backup id shown as MMDD, e.g. 0917

3: full backup id shown as MMDDhhmm, e.g. 09171305

4: full backup id shown as DDhhmmss, e.g. 17130558

**%[n]B**—The backup identification number.

**%[n]N**—The name of the database the journal file belongs to.

➤ Example

```
DB_BkFrm = %3F%n.%B
```

If the database name is test1, the incremental backup files will be named *09171305test1.1.jnl*, *09171305test1.2.jnl*...

For more in-depth information, refer to the section “*Setting the Backup Filename Format*” in Chapter 14 of the *Database Administrator's Guide*. This setting corresponds to the [DB\\_BkFrm](#) keyword in the **dmconfig.ini** file. The default file name format is %2F%4N%4B.jnl.

## Setting the Start Time for Incremental Backups

To set the time at which the first incremental backup will start being processed for the database, enter the date in the yyyy/mm/dd field, and the time in the hh:mm:ss field. Reenter the numbers if they appear incorrectly the first time. JConfiguration Tool automatically enters values into the first two spaces of the yyyy field and the first space of the hh field if values are entered into other fields. This setting corresponds to the [DB\\_BkTim](#) keyword in the **dmconfig.ini** file. There is no default value.

## Setting the Incremental Backup Daemon Interval

The number in the combo box specifies the time interval at which the Incremental Backup occurs in number of days. Next to the combo box, there is a field for time input, which specifies the time interval in hours, minutes, and seconds. The total time interval is determined by adding the two values together, so inputting 1 into the day's field and inputting 12:00:00 into the hh:mm:ss field would cause an incremental backup to occur every day and a half. The number of days can be manually entered into the combo box, or increased or decreased by clicking the up and down arrows to the right of the field. The number of hours, minutes, and seconds can only be manually entered. This setting corresponds to the [DB\\_BkIty](#) keyword in the **dmconfig.ini** file.

## Enabling Compact Backup

Checking the Enable Compact Backup checkbox specifies whether Backup Server will backup entire journal files or only full journal blocks when it performs an online incremental backup. If the **Enable Compact Backup** checkbox is enabled, then the Backup server will only backup journal blocks not previously backed up. Not every journal block contains data needed to restore a database, so Backup Server will only copy the necessary journal blocks when it performs a backup. This allows the user to save storage space on the backup media, but it also means that restoring a database may take more time. This setting corresponds to the [DB\\_BkCmp](#) keyword in the **dmconfig.ini** file. The default value is *enabled*.

## Incremental Backup Initiation Value

The user may want to allow DBMaster to create an incremental backup before the journal file is completely full. The journal trigger value specifies the percentage a journal file must fill before Backup Server will perform an online incremental backup. The journal trigger value and the backup schedule can be combined to backup a database on a regular schedule and when journal files fill to a specified percentage.

Selecting Backup when any Journal File is **Full** sets the Incremental Backup Process to activate any time the journal file is full. Selecting 50-100 allows the user to input a value into the % full

combo box. Users can manually input a value between 50% and 100% or use the up and down arrows to the right of the field to adjust this value. The journal file will be backed up any time it is filled to the percentage specified. This setting corresponds to the [DB\\_BkFu](#) keyword in the `dmconfig.ini` file. The default value is *full*.

## 5. Keywords in DMconfig.ini

This chapter provides some keywords you can use them to change settings before backup or restore a database.

DBMaster configuration parameters are stored as keywords in a configuration file, **dmconfig.ini**.

In this chapter we mostly betray relative keywords in DMconfig, it must be amended yourself. The following sections are these keywords' declaration, please read it carefully.

### 5.1 DB\_BkSvr

`DB_BkSvr=<value>`

This keyword specifies whether or not a backup server will be started when a database is started. Setting this value to 1 will start a backup server for that database before backup by backup server.

**Default value:** 0

**Valid range:** 0, 1

**See also:** `DB_BkCmp`, `DB_BkDir`, `DB_BkFul`, `DB_BkTim`, `DB_BkItv`

**Where to use:** server side

### 5.2 DB\_BkDir

`DB_BkDir=<string>`

This keyword specifies the directories where the backup server puts the database backup files. These directories must already exist and can be different from **DB\_DbDir**.

**Valid range:** string with length < 256

**See also:** `DB_BkSvr`, `DB_BMode`

**Where to use:** server side

### 5.3 DB\_BkOdr

`DB_BkOdr=<string>`

This keyword specifies the directories where the backup server puts the pervious version of full backup files.

**Default value:** none.



**Valid range:** string with length < 256

**See also:** `DB_BkSvr`, `DB_BMode`, `DB_FBkTm`, `DB_FBkTv`

**Where to use:** server side

## 5.4 DB\_BkFoM

`DB_BkFoM=<value>`

This keyword specified the file object (FO) backup mode. File objects will only be backed up during a full backup of the database. **DB\_BkFoM** has three possible values; 0, 1, and 2. Setting **DB\_BkFoM** equal to zero disables the FO backup feature; file objects will not be backed up during a full backup. Setting **DB\_BkFoM** equal to one enables system file objects to be backed up during a full backup. Setting **DB\_BkFoM** equal to two enables both system file objects and user file objects to be backed up.

If database have file object, before you backup or restore a database, you must set it.

**Default value:** 0

**Valid range:** 0: File objects are not backed up

1: System file objects are backed up

2: System and user file objects are backed up.

**See also:** `DB_BkSvr`, `DB_FBkTm`, `DB_FBkTV`, `DB_BkDir`.

**Where to use:** server side

## 5.5 DB\_BkFrm

`DB_BkFrm=<value>`

The keyword allows you to specify the format Backup Server used to name incremental backup journal files. The backup filename format may include both text constants and format sequences (escape sequences), that represent special character strings.

You can use the format sequences to represent the year, month, or date the backup was performed, the name of the database, or the backup identification number. You may combine format sequences and text constants in any way, provided the result is a valid filename supported by the operating system. The format sequences have three parts: the escape character, the length value, and the format character. The valid format sequences are:

**%[n]Y**—The backup year of the journal file was.

**%[n]M**—The backup month of the journal file was.

**%[n]D**—The backup day of the journal file was.

**%[n]B**—The backup identification number.

**%[n]N**—The name of the database the journal file belongs to.

## 5.6 DB\_BkFul

`DB_BkFul=<value>`

This keyword specifies the percentage that all journal files must be filled to before the backup server is triggered to do an incremental backup. Setting this value to 0 will trigger the backup server whenever a journal file is full. Setting this value is between 50 and 100 will trigger the backup server whenever the total space used in all of the journal files exceeds the specified percentage. For example, if there are two journal files of 500 journal blocks each and DB\_BKFUL is set to 80, then after every  $500 \times 2 \times 0.8 = 800$  blocks are used, the backup server will automatically do an incremental backup.

**Default value:** 0

**Valid range:** 0, 50 ~ 100

**See also:** DB\_BkSvr, DB\_BkTim, DB\_BkItv

**Where to use:** server side

## 5.7 DB\_BkItv

`DB_BkItv=<string>`

This keyword specifies the backup time interval. Please refer to DB\_BkTim described later.

**Default value:** none (no backup schedule if DB\_BkItv is not set)

**See also:** DB\_BkSvr, DB\_BkTim, DB\_BMode

**Where to use:** server side

## 5.8 DB\_BkCmp

`DB_BkCmp=<value>`

This keyword specifies whether the compact backup mode is used. Not every journal block in a journal file is needed to perform a backup. If this keyword is set to 1 the backup server will only back up those journal blocks that require back up to save disk space. Also see the chapter Database Backup, Recovery, and Restoration.

**Default value:** 1

**Valid range:** 0, 1

**See also:** DB\_BkSvr

**Where to use:** server side

## 5.9 DB\_BKRTS

`DB_BKRTS=<value>`

This keyword specifies whether the backup server backs up the read-only tablespace files when doing full-backup. The default value is 1 and will backup the read-only tablespace files. If you

have already backed up the read-only tablespace files and you don't want the backup server backups it again, you can assign 0 to the DB\_BKRTS. If you assign 0 to the keyword, please note.

## 5.10 DB\_BkTim

`DB_BkTim=<string>`

This keyword along with **DB\_BkItv** specifies the schedule of the backup server. **DB\_BkTim** specifies the first time a backup server will perform an incremental backup. Incremental backup will then be performed after every time interval specified in **DB\_BkItv**. You can change it to set the time you want to backup, and then DBMaster will backup the database automatically.

## 5.11 DB\_BkZIP

`DB_BkZip=<value>`

This keyword specifies whether a backup server compresses the backup files when performing full backups.

Setting the keyword to 1 that compresses files during a full backup. Setting the keyword to 0 does not compress files during a full backup.

**Default value:** 0

**Valid range:** 0, 1

**See also:** **DB\_BkSvr, DB\_BkCmp, DB\_BkDir, DB\_BkFul, DB\_BkTim, DB\_BkItv**

**Where to use:** server side

## 5.12 DB\_BMode

`DB_BMode=<value>`

This keyword specifies the backup mode of a database. Setting the value to 0 enables NON-BACKUP mode, 1 enables BACKUP-DATA mode, and 2 enables BACKUP-DATA-AND-BLOB mode. Also see the chapter Database Backup, Recovery, and Restoration.

**Default value:** 0

**Valid range:** 0 ~ 2

**See also:** **DB\_BkSvr**

**Where to use:** server side

## 5.13 DB\_FBkTm

`DB_FBkTm=<string>`

This keyword combined with **DB\_FBkTv**, specifies the schedule of the Backup Server to perform an on-line full backup. **DB\_FBkTm** specifies the first time the Backup Server will perform a full backup. On-line full backup will be performed after every time interval specified in **DB\_FBkTv**.

The keywords **DB\_FBkTm** and **DB\_FBkTv** are only used with the Backup Server.

**Default value:** none

**See also:** DB\_FBkTv, DB\_BkSvr, DB\_BkOdr

**Where to use:** server side

## 5.14 DB\_FBkTv

**DB\_FBkTv=<string>**

This keyword specifies the full backup time interval. Refer to DB\_FBkTm for more information.

Default value: none (no full backup schedule if DB\_FBkTv is not set)

**See also:** DB\_BkSvr, DB\_FBkTm, DB\_BkOdr

**Where to use:** server side

## 5.15 DB\_RTime

**DB\_RTime=<string>**

This keyword specifies the target time for a database to be restored from a backup. When performing a database restoration, DBMaster will roll forward on the backup files from the earliest time in the backup files to the time specified by **DB\_RTime**. If **DB\_RTime** is not given, DBMaster will restore the database to the latest time in the backup files, which is the time the backup was performed, you can set the time you want to restore the database.

If the **DB\_RTime** is later than the backup time, the backup time will be used as the value for **DB\_RTime**.

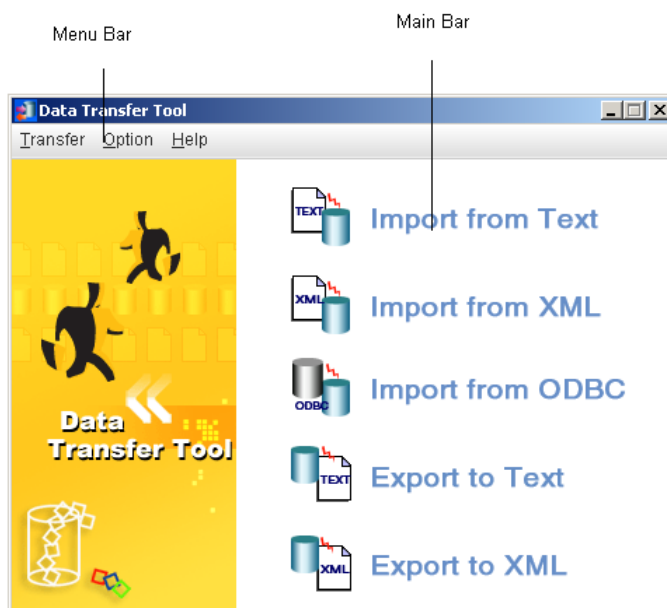
The format for **DB\_RTime** is **yy/mm/dd hh:mm:ss**.

**Default value:** 0 (70/1/1 00:00:00)

**Where to use:** server side

## 6. JData Transfer Tool

The Data Transfer Tool is a separate application, you can view it as extension to backup database; it may be opened from windows start>programs>DBMaster 5.1>JData Transfer, or opened from within JDBC Tool. It consists of a main console and a menu bar, as illustrated in Figure. The main console provides five options: import from text, import from XML, and import from ODBC, export to text, and export to XML. The Menu Bar consists of three menus: the Transfer menu, the Option menu, and the Help menu. The Transfer menu provides the same transfer functions as the main console, with the addition of the batch transfer function. The option menu can be used to change the language that the UI is displayed in; currently English, Japanese, and Chinese (traditional) are the supported languages. The help menu provides access to the help system for JDBC Tool.



➡ To open the Data Transfer Tool:

1. Start JDBC Tool and connect to the database that data is to be transferred to or from.
2. Select **Data Transfer** from the Tool menu. The Data Transfer tool window will open.

### 6.1 Importing data from Text

The ability to import table data from a text file is an important feature in a database, and made the Data Transfer Tool easy to use. Text data must be properly formatted to be acceptable for import. Data may be imported to the database only from a properly formatted text file. This section describes the types of formatting that Data Transfer Tool supports, and then describes how to import text to the database using the tool.

Before attempting to import data from a text file, you should be sure that the file is in a format that will result in coherent structured data within the database. Certain programs output data in a fixed text format, if this is the case, check the format of the output file that you want to import. Some important settings to consider in the format of a text file include the following:

- **Row Delimiter:** Determines the type of character that signifies a break between the rows of a table. Possible characters: {CR/LF} (Carriage return / line feed. In Windows applications, a new line in the text is normally stored as a pair of CR LF characters. In UNIX applications, a new line is normally stored as a LF character. Some applications use only a CR character to store a new line), {CR}, {semicolon} (;), {comma} (,), {tab}, {vertical bar} (|), {semicolon} {LF}, or {comma} {LF}.
- **Column Delimiter:** Determines the type of character that signifies a break between columns in each row. Possible characters: semicolon, comma, or vertical bar.
- **Text Qualifier:** Determines how each tuple of any data type except BINARY, LONGVARBINARY, or numeric data types (integer, smallint, serial, decimal, double, float) is enclosed. Possible values: none, single quote, or double quote.
- **Binary Qualifier:** Determines how each tuple of BINARY or LONGVARBINARY type data is enclosed. Possible values: none, single quote, or double quote.
- **Binary Padding:** Binary type data may have a character appended to it.
- **Fixed Field:** Instead of using a row delimiter, the text file may be formatted with fixed fields. This means a number of spaces, or fields, defines each column.
- **Include column name:** The first line in a text file may be used to define the column names. The format is "*column1*".*"table name"*.*"owner name"*;*"column2"*.*"table name"*. *"owner name"*; etc. In this case the column delimiter is set to semicolon (;).
- **Include table schema:** The first line in a text file may be used to define the column schema (or the second line if the first line was used to define column names). The format is *data type(scale,precision)*;*data type(scale,precision)* etc. In this case, the column delimiter is set to semicolon (;).
- **Use NULL to display null data:** Columns that contain no data display "NULL".
- **File link name for FILE type data:** The file name for system or user file objects is displayed.
- **Use escape character "/":** This character is used when qualifiers or delimiter characters appear in the data. If the data contains a reserved character, the reserved character will be enclosed by an escape character (/) so that the text import engine recognizes it as data, not a qualifier or delimiter.
- **Use temp files to store LONGVARBINARY or LONGVARCHAR type data column content:** BLOB data is stored as a separate, linked file (as a file object), and the name of the file containing the BLOB is displayed.

Data may be imported to a new table or an existing table. Three options are available when importing data to an existing table. The destination table may be replaced, removing the schema and data of the original table. The rows of the destination table may be replaced, meaning schema is retained, but data is removed. The last option is to append rows to the destination table, in which case the table's original schema and data is preserved, and the data from the text file is appended as new rows. In the last two cases, be aware that the schema of the destination table must be such that it is able to accept the imported data. For example, a set of integer data can be

imported into a column of CHAR type data, but not vice-versa. Refer to the *SQL Command and Function Reference* for more information on acceptable formats for supported data types in DBMaster.

➤ Example

The following text file contains owner, table, and column information in the first line, and the table's schema in the second line. Subsequent lines contain raw data. Semicolons delimit columns. Single quotes qualify non-binary data; double quotes qualify binary data. The LONGVARCHAR and LONGVARBINARY columns display temporary file names (the data has been stored as file objects), and binary data is padded with "9".

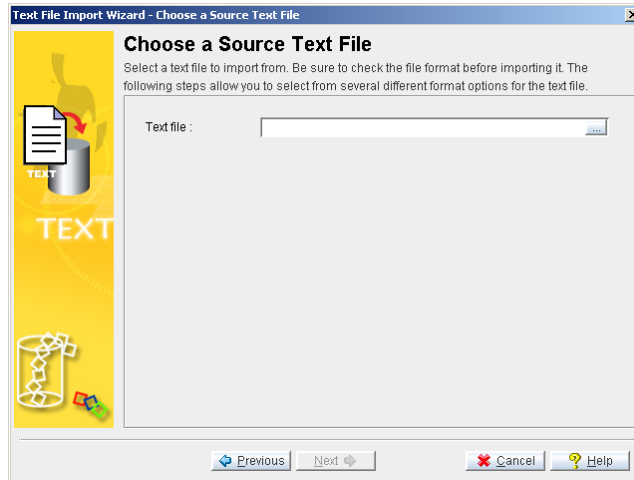
```
"SYSADM"."EXPORT"."LOGINID";"SYSADM"."EXPORT"."REQUEST";"SYSADM"."EXPORT"."REQUESTTIME";"SYSADM"."EXPORT"."ATTACHMENT";"SYSADM"."EXPORT"."BINARY_C";"SYSADM"."EXPORT"."DECIMAL_C"
SQL_CHAR(20);SQL_LONGVARCHAR;SQL_TIMESTAMP;SQL_FILE;SQL_BINARY(10);SQL_DECIMAL(10,3)
'A_HOWARD          '; 'blobtmpdir2\bltmpf0.txt'; '2001-09-09
12:47:05.000'; 'C:\DBMASTER\5.1\BIN\WEBDB\F0\ZZ0000B.GIF'; "10000000000000000000"
9;10.250
'A_HOWARD          '; 'blobtmpdir2\bltmpf1.txt'; '2001-09-22
10:14:21.000'; 'C:\DBMASTER\5.1\BIN\WEBDB\F0\ZZ0000C.GIF'; "20000000000000000000"
9;13.550
'A_HOWARD          '; 'blobtmpdir2\bltmpf2.txt'; '2001-10-04
16:22:06.000'; 'C:\DBMASTER\5.1\BIN\WEBDB\F0\ZZ0000D.GIF'; "30000000000000000000"
9;27.333
```

➤ To import a text file to a database

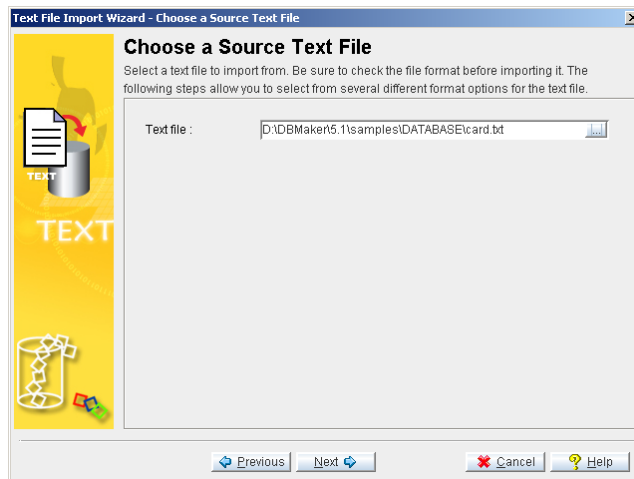
1. Open the Data Transfer Tool.
2. Select **Import Text File** from the main console or the **Transfer** menu. The Welcome to Import from Text File Wizard window will open, displaying a summary of the steps to be taken in the wizard



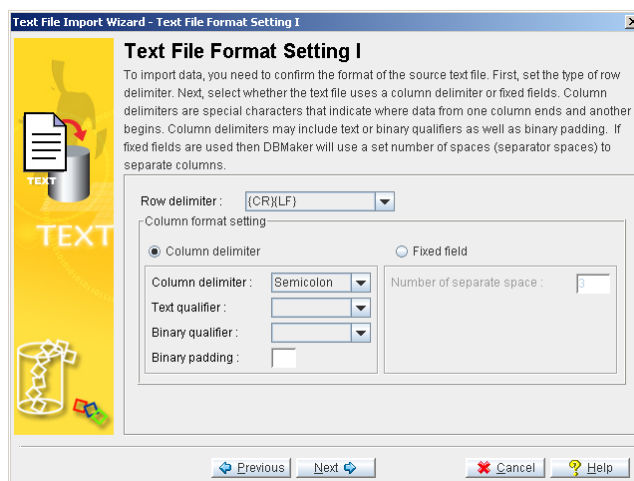
3. Click **Next**. The **Choose a Source Text File** window will open.



4. Enter the full path of a text file to import or click the browse button to search for a text file.

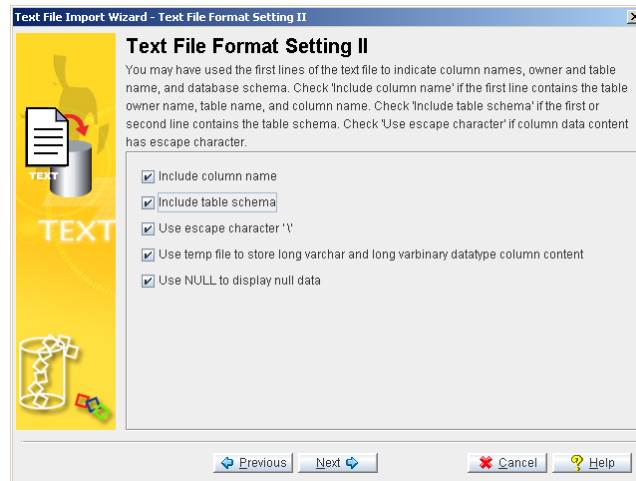


5. After you have selected a text file, click **Next**, the **Text File Format Setting 1** window will open.

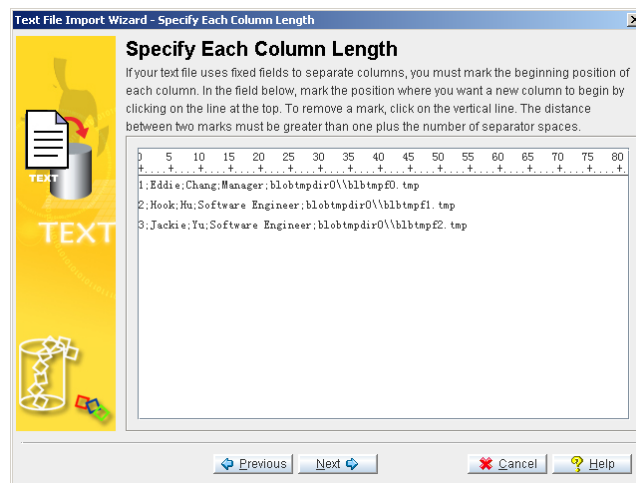




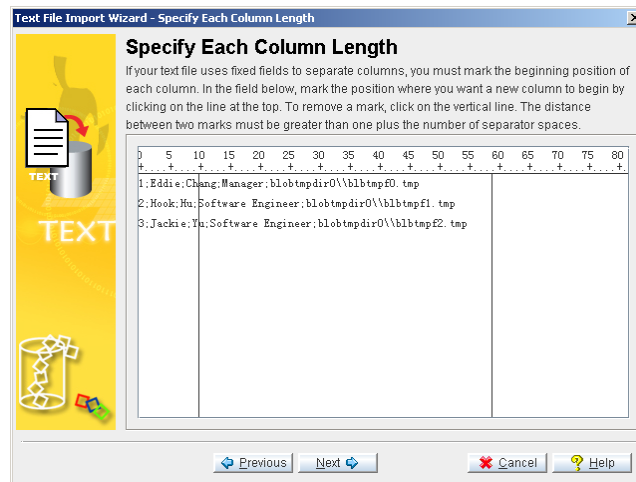
6. Open the text file in a text editor to check the format of the data.
7. Select the appropriate settings for the format of the text file you are importing.
8. Click **Next**, the **Text File Format Setting 2** window will open.



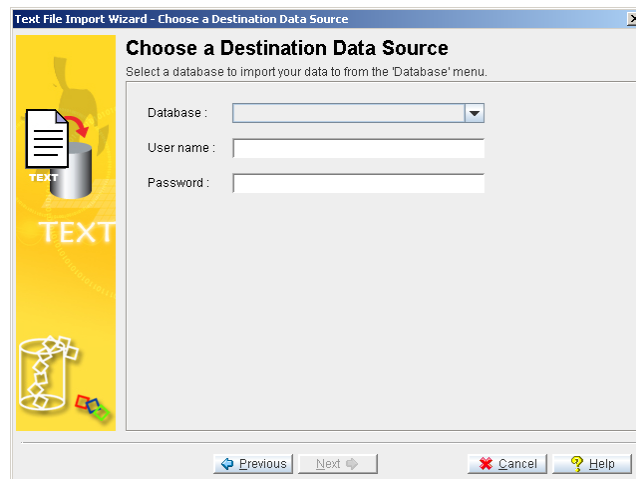
9. Finish selecting the appropriate settings for the format of the text file you are importing. Click **Next**.
10. If you selected **Fixed field** from the **Text File Format Setting 1** window, the **Specify Column Length** window will open. If you selected **Column Delimiter**, then proceed to step 12.



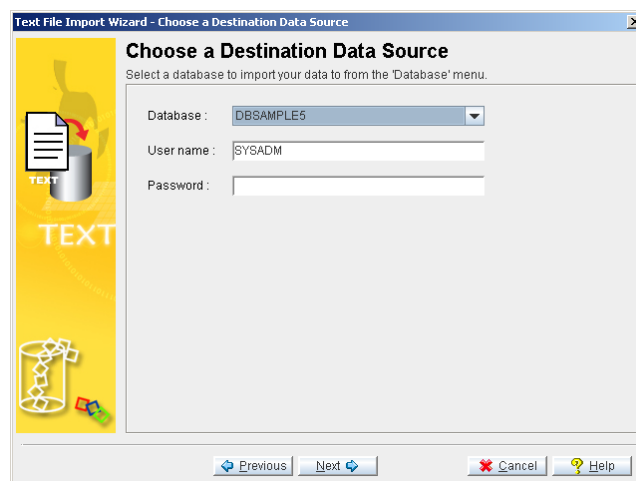
- Click the horizontal line where you want to indicate the beginning of a column. A vertical line will appear, marking the column break. Use the scroll bar at the bottom to advance to other columns. Click **Next**.



- The **Choose Destination Data Source** window will open.



- Select the database to import data to from the **Database** menu.
- Enter a user name and password into the appropriate fields.



**NOTE** *A user must have **INSERT** privilege to import a text file.*

15. Click **Next**. The **Transfer Setting** window will open.



16. Enter a new table name into the **Table Name** field, or select a table from the menu. Selecting a table from the menu will allow you to choose to replace the destination table, delete rows in the destination table, or append new rows to the destination table. Click **Execute** to import the text file. A confirmation dialog box will appear.

17. Click OK

## 6.2 Importing data from XML

XML files may also be imported into the database. XML tags may first be defined in a *Document Type Definition (DTD)* file before being imported into the database. Furthermore, the DTD may define the schema in a way that is acceptable to the database.

It is important to consider the structure of the XML file you wish to import. To ensure that the structure of the XML file and associated DTD have compatible structure, examine the structure of XML files produced by the Data Transfer Tool: Export to XML File wizard. Examples may be found in section 6.5, *Exporting Data to XML*. Files produced using the Export to XML wizard always can be imported; however, the extent to which a table's schema is reproduced varies. The setting that influences table schema the most is the Column as Element / Attribute setting. Make sure the destination database is started.

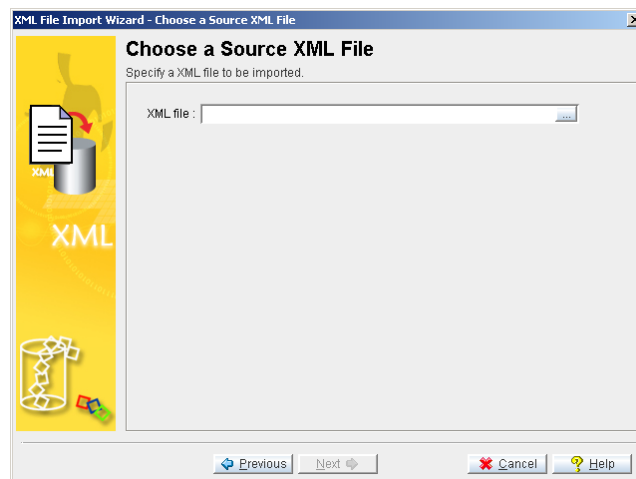
- Column as Element: Stores data items in elements. If table schema information exists as element attributes (data type, column name, length, etc.) in the DTD, then columns will be created with names and of the appropriate data type and length. Columns are child elements, and the table is represented as the parent element. File objects must be referenced as entities in the DTD file if Column as Element is chosen.
- Column as Attribute: Stores data in an attribute of an element. Each element is a record. If column names are represented as attributes of the root element (the table) in the DTD, and each tag in the XML file represents one record, then Column as Attribute should be chosen.

➤ To import data from an XML file:

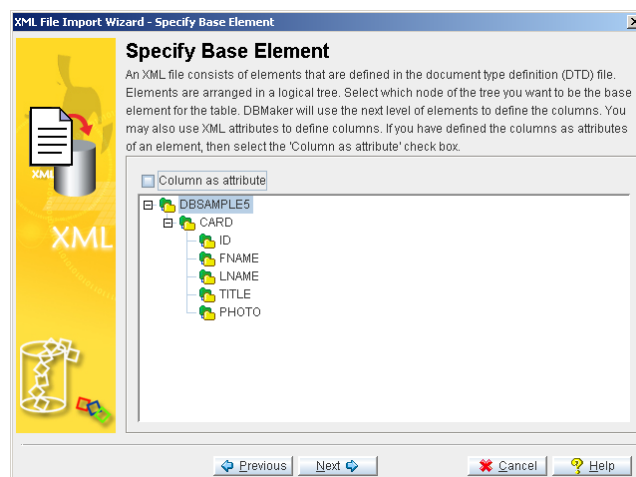
1. Open the Data Transfer Tool.
2. Select **Import XML File** from the main console or the **Transfer** menu. The **Welcome to Import from XML File Wizard** window will open, displaying a summary of the steps to be taken in the wizard.



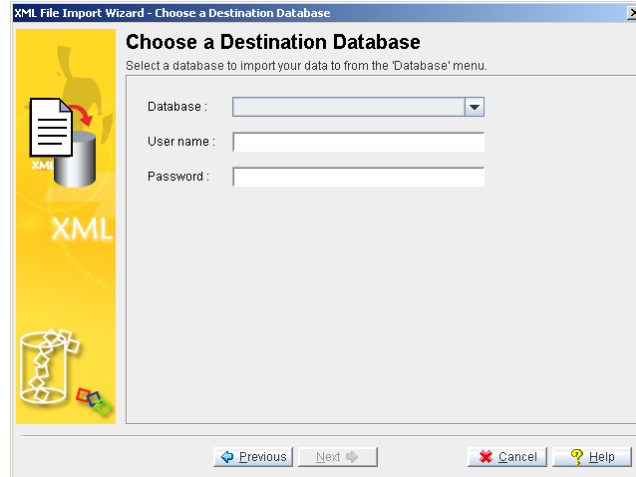
3. Click **Next**. The **Choose a Source XML File** window will open.



4. Enter the full path of a text file to import or click the browse button to search for a text file.
5. Click **Next**. If the XML file has a structure acceptable to DBMaster's parser, the **Specify Base Element** window will open.



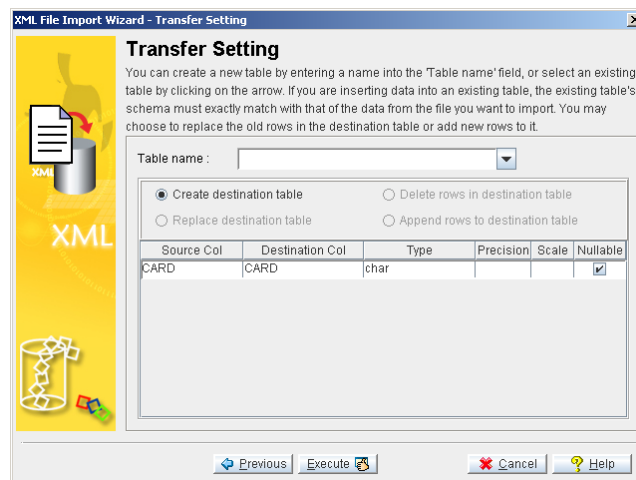
6. The nodes of the tree structure represent the elements in the XML file. Click the nodes on the tree until they are fully expanded. Select a parent element to be the table name. The child elements will become the columns of the table. Check Column as attribute if appropriate.
7. Click **Next**. The **Choose a Destination Data Source** window will open.



8. Select the database to import data to from the **Database** menu.
9. Enter a user name and password into the appropriate fields.

**NOTE** *DBA authority or higher is required to import a text file.*

10. Click **Next**. The **Transfer Setting** window will open.



11. Enter a new **table name** into the Table Name field, or select a table from the menu. Selecting a table from the menu will allow you to choose to replace the destination table, delete rows in the destination table, or append new rows to the destination table. Click **Execute** to import the XML file. A confirmation dialog box will appear.
12. Click **OK**

## 6.3 Importing data from ODBC

A large number of software developers have developed applications to be Open Database Connectivity (ODBC) compatible. ODBC is an industry standard for sharing data between diverse data sources. DBMaster can import data from any ODBC compliant data source through the Import from ODBC wizard.

Data may be imported by three methods:

- Directly from tables.
- Writing one or more SQL SELECT statements.
- Importing through an XML batch file.

Furthermore, you may specify the mapping of column data through the transformation function. The transformation function supports direct column-to-column mapping or mapping through SQL SELECT and SQL INSERT statements.

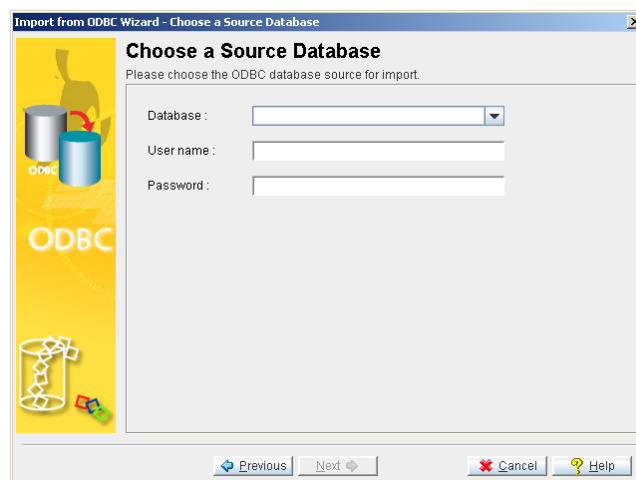
When importing data directly from tables or through SQL SELECT statements it is possible to save a 'map' of the data transformation to an XML batch file. The saved XML batch file is a well-formed XML document with a form that can be parsed by the data transfer tool. Batch files may be used to import table schema from a data source to multiple DBMaster databases.

☞ To import data from an ODBC database:

1. Open the Data Transfer Tool.
2. Select **Import XML File** from the main console or the **Transfer** menu. The **Welcome to Import from XML File Wizard** window will open, displaying a summary of the steps to be taken in the wizard.



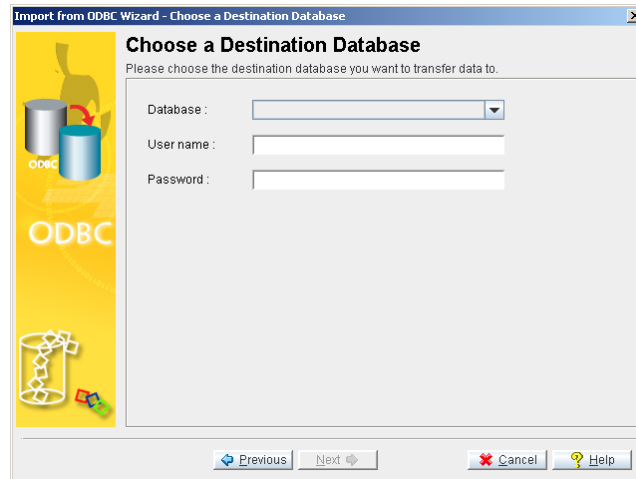
3. Click **Next**. The **Choose a Source Database** window will open.



4. Select the database to export data from in the **Database** menu.
5. Enter a user name and password into the appropriate fields.

**NOTE** *DBA authority or higher is required to export a text file.*

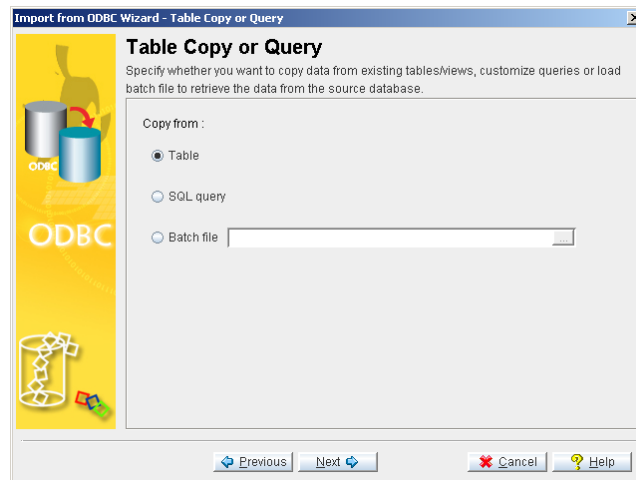
6. Click **Next**. The **Choose a Destination Data Source** window will open.



7. Select the database to import data to from the **Database** menu.
8. Enter a user name and password into the appropriate fields.

**NOTE** *DBA authority or higher is required to import a text file.*

9. Click **Next**, the **Table Copy or Query** window will open.



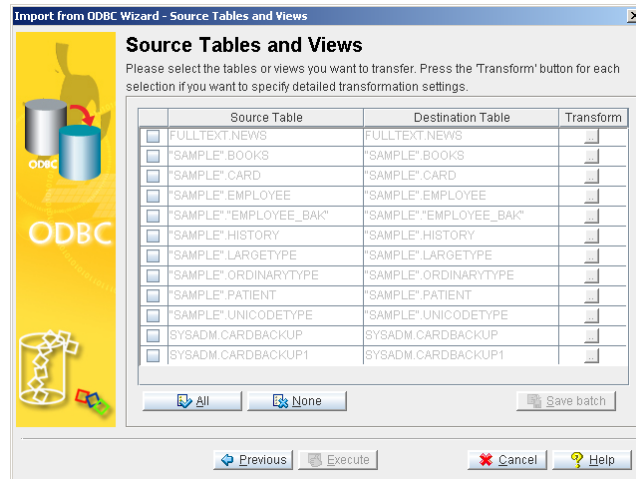
10. Select from one of the three methods for data transfer:

- To import data from a list of tables, select **Table**.
- To import data using a series of SQL SELECT statements, select **SQL query**.
- To import data through an XML file, select **Batch file**

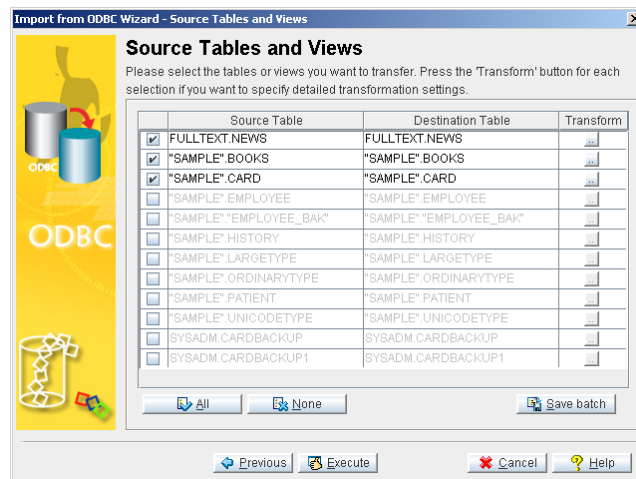
### 6.3.1 IMPORTING ODBC DATA FROM TABLES

ODBC data may be imported from variety of sources by selecting tables directly from the source database. The Import from ODBC Wizard provides an intuitive graphical interface for selecting tables and setting how data should be transformed during the import process.

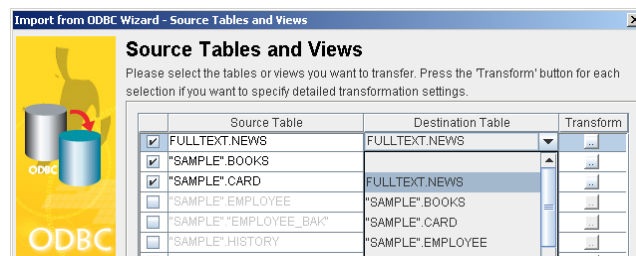
- To import ODBC data from a list of tables
  1. In the Import from ODBC wizard, select a source and destination database, and then select **Table** from the **Table Copy or Query** window.
  2. Click **Next**. The **Source Tables and Views** window will open.



3. All tables from the source database will appear in the **Source Table** column. Check the box to the left of each table to import.

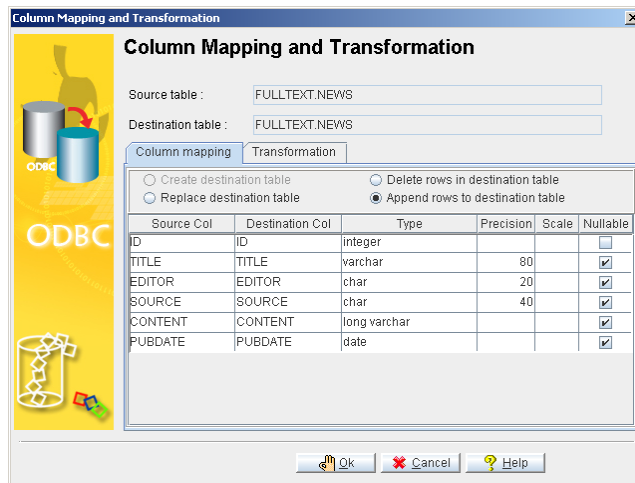


4. For each source table or view selected, click the **Destination Table** field. If desired, change the name of the destination table by selecting a new table from the menu or entering a new name.





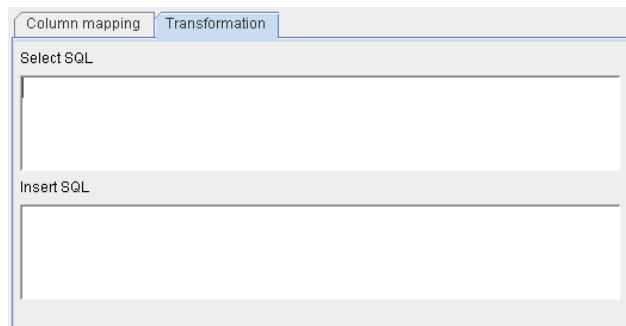
- You may modify column mapping or the result set to import by clicking on the **Transform** button of the corresponding source and destination table.



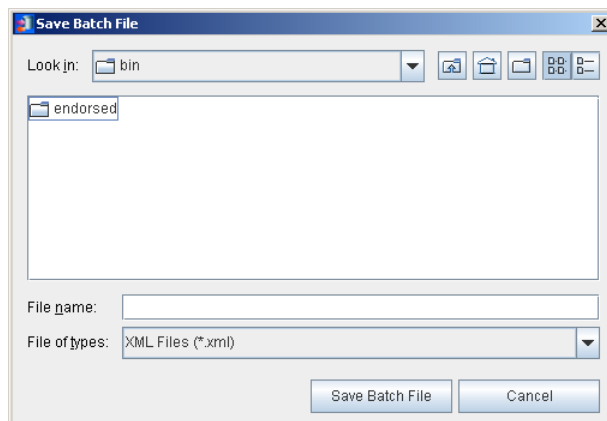
- Change the name of the destination column by selecting a new column from the menu or entering a new name.

Source Col	Destination Col	Type	Precision	Scale	Nullable
ID	ID	integer			<input type="checkbox"/>
TITLE		varchar	80		<input checked="" type="checkbox"/>
EDITOR	ID	char	20		<input checked="" type="checkbox"/>

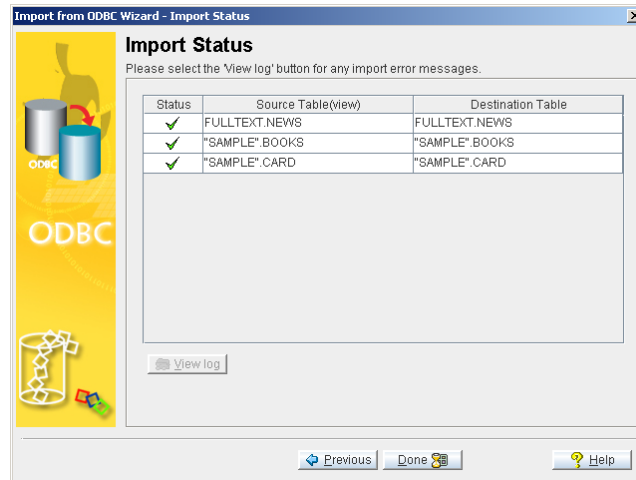
- Click the **Transformation** tab to specify constraints on the result set. Enter a valid SQL SELECT statement into the **Select SQL** field and a valid SQL INSERT statement into the **Insert SQL** field.



- Click **OK** to return to the **Source Tables and Views** window.
- You may also choose to save the map of the import ODBC schema to an XML file by clicking **Save batch**. The **Save Batch File** will open.



10. Select or create an XML file to save the imported ODBC map schema to. Click **Save Batch File** to create the XML file. The **Source Tables and Views** window will reappear.
11. Click **Execute** to import the source data. The Import Status window will appear.

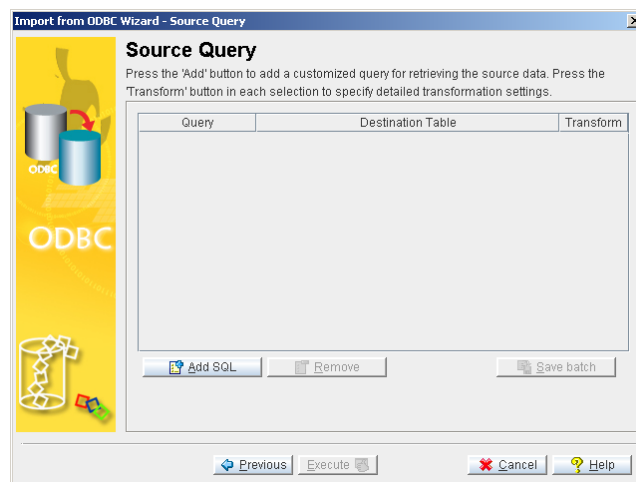


12. If errors appear, click **View log** and scroll to the bottom to see the error message. If no errors occurred, click **Done**.

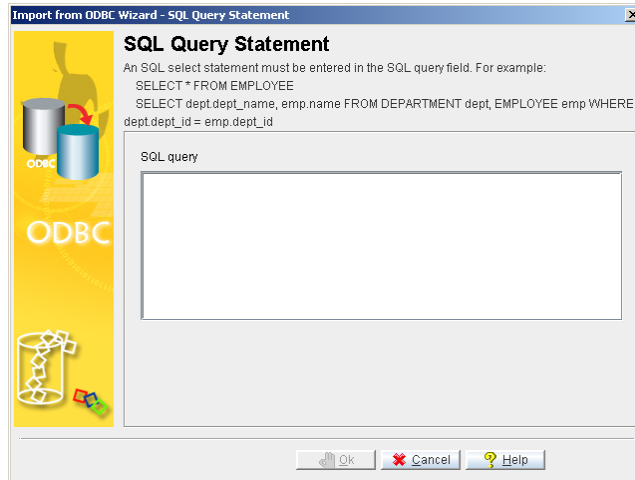
## 6.3.2 IMPORTING ODBC DATA USING SQL SELECT STATEMENTS

Data may also be imported from ODBC sources by creating a series of SQL SELECT statements. If you have knowledge of the schema of tables you want to import, this may be a faster option.

- To import ODBC data using a series of SQL SELECT statements
  1. In the Import from ODBC wizard, select a source and destination database, and then select Table from the Table Copy or Query window.
  2. Click **Next**. The **Source Query** window will open.

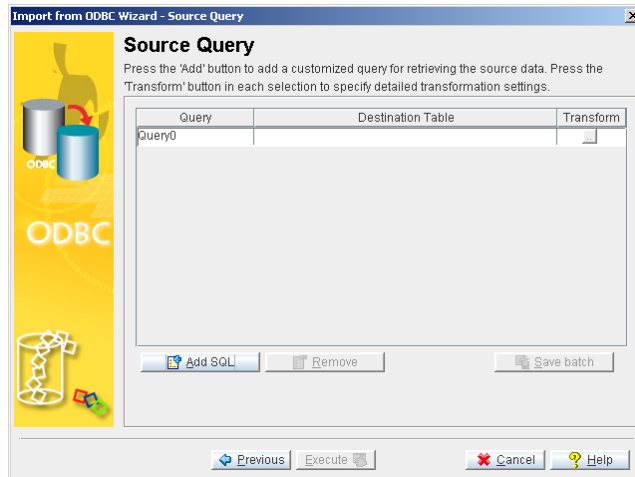


3. Click **Add SQL**. The **SQL Query Statement** window will open.

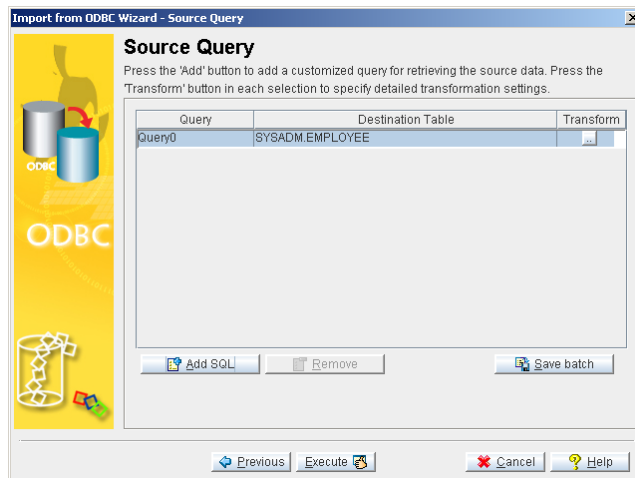


4. Enter a valid SQL SELECT statement into the SQL Query field.

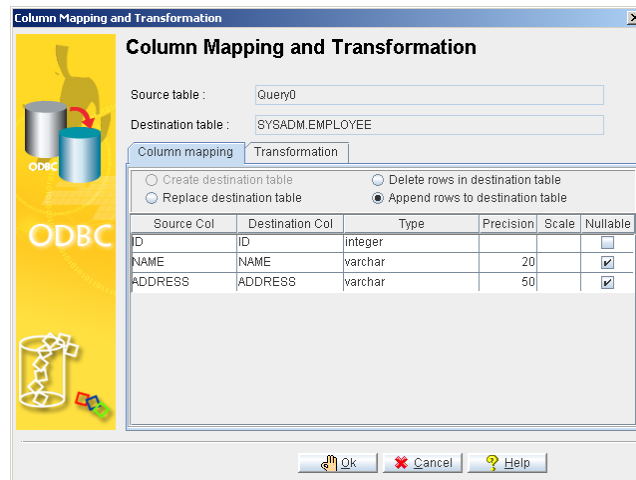
5. Click **OK**. The **Source Query** window will reappear.



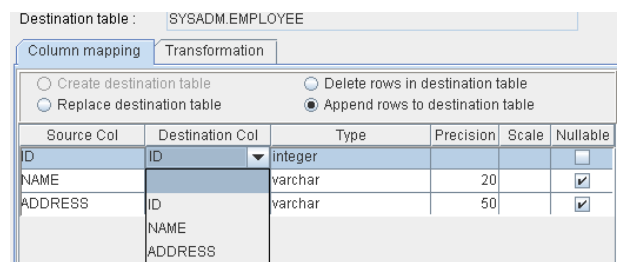
6. Select or create a destination table from the **Destination Table** column.



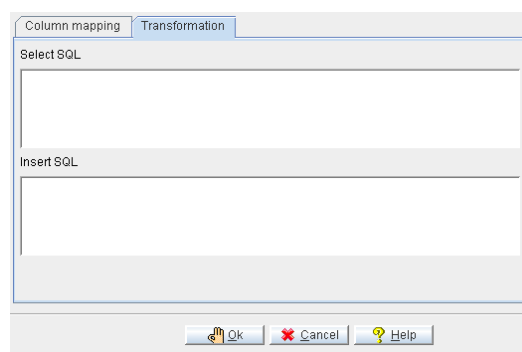
7. You may add more SQL query statements by clicking **Add SQL**, or modify the mapping of source and destination columns by clicking the **Transform** button.



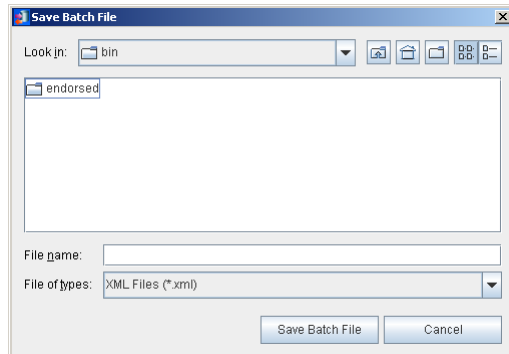
8. Change the name of the destination column by selecting a new column from the menu or entering a new name.



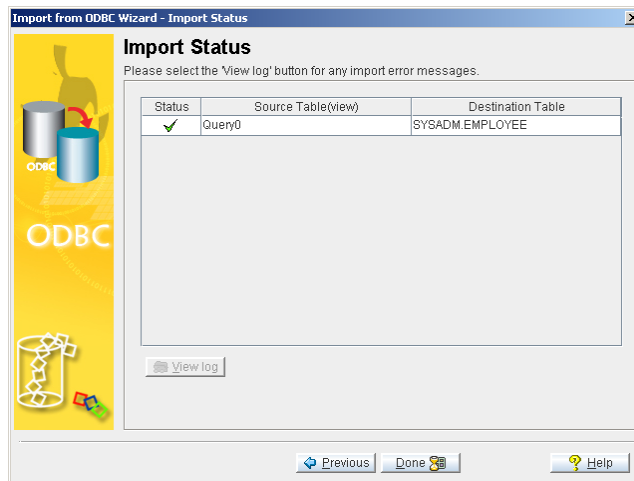
9. Click the **Transformation** tab to specify constraints on the result set. Enter a valid SQL SELECT statement into the **Select SQL** field and a valid SQL INSERT statement into the **Insert SQL** field.



10. Click **OK** to return to the Source **Tables and Views** window.
11. You may also choose to save the map of the imported ODBC schema to an XML file by clicking **Save batch**. The **Save Batch File** will open.



12. Select or create an XML file to save the import ODBC map schema to. Click **Save Batch File** to create the XML file. The **Source Tables and Views** window will reappear.
13. Click **Execute** to import the source data. The Import Status window will appear.



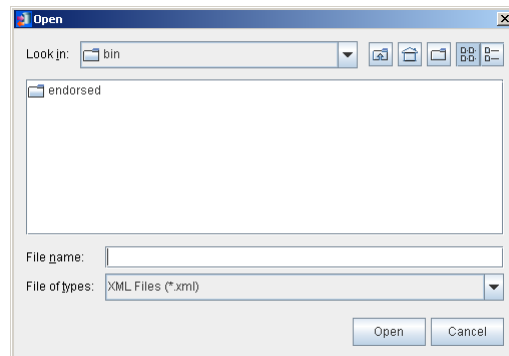
14. If errors appear, click **View log** and scroll to the bottom to see the error message. If no errors occurred, click **Done**.

### 6.3.3 IMPORTING ODBC DATA THROUGH AN XML BATCH FILE

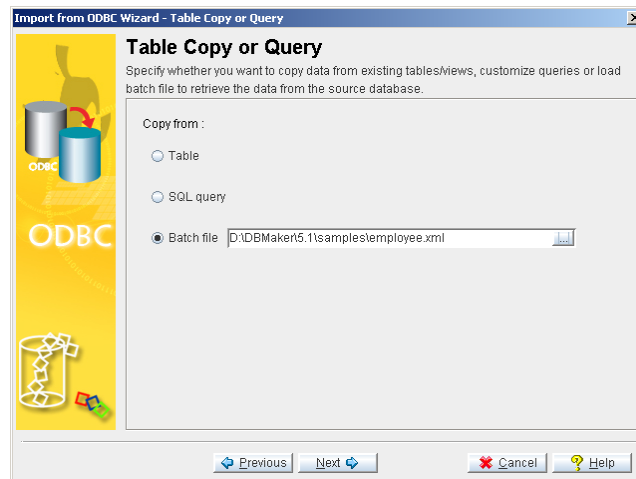
It is also possible to use an XML Batch file to specify which tables are to be imported. Users have the option to create an XML batch file when importing ODBC data from tables or with SQL Select statements. Batch files may be used to import table schema from a data source to multiple DBMaster databases

- ➡ To import data through an XML batch file

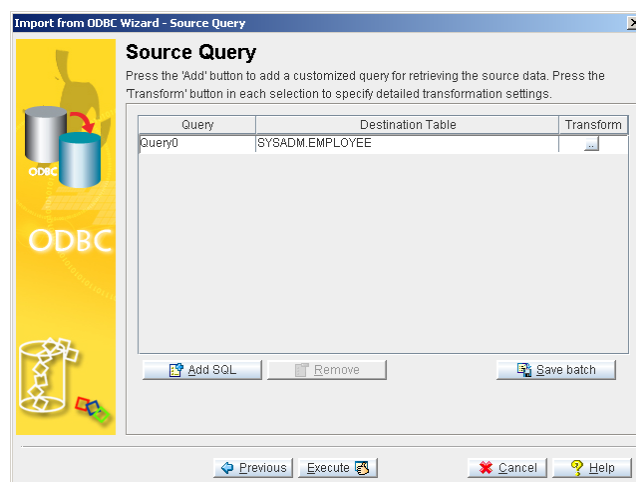
1. In the Import from ODBC wizard, select a source and destination database, and then select **Batch file** from the **Table Copy or Query** window. The **Open** window will open.



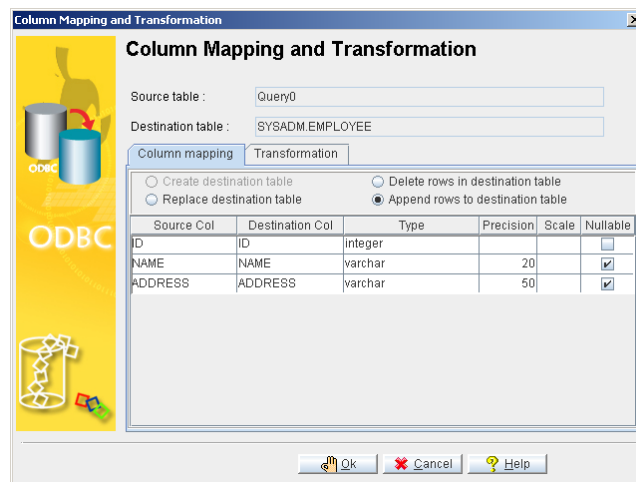
2. Select an XML file from which to import the ODBC map schema.
3. Click **Open**. The **Table Copy or Query** window will reappear.



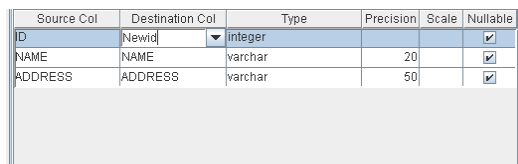
4. Click **Next**. The **Source Tables and Views** window will open, displaying a mapping schema according to the XML file.



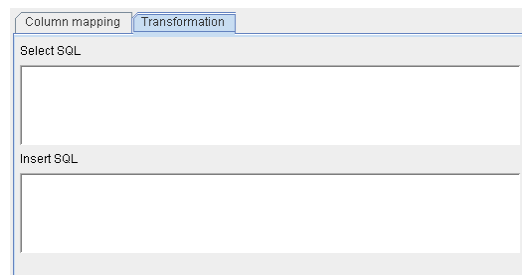
5. For each source table or view selected, click the **Destination Table** field. If desired, change the name of the destination table by selecting a new table from the menu or entering a new name.
6. You may modify column mapping or the result set to import by clicking on the **Transform** button of the corresponding source and destination table.



7. Change the name of the destination column by selecting a new column from the menu or entering a new name.

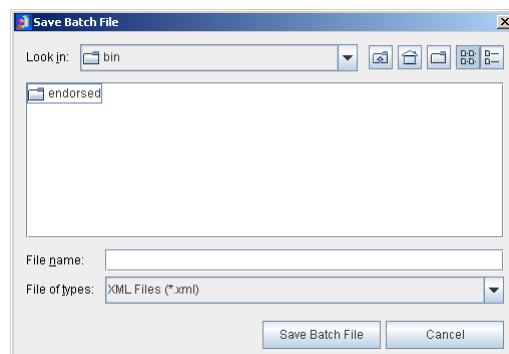


8. Click the **Transformation** tab to specify constraints on the result set. Enter a valid SQL SELECT statement into the **Select SQL** field and a valid SQL INSERT statement into the **Insert SQL** field.



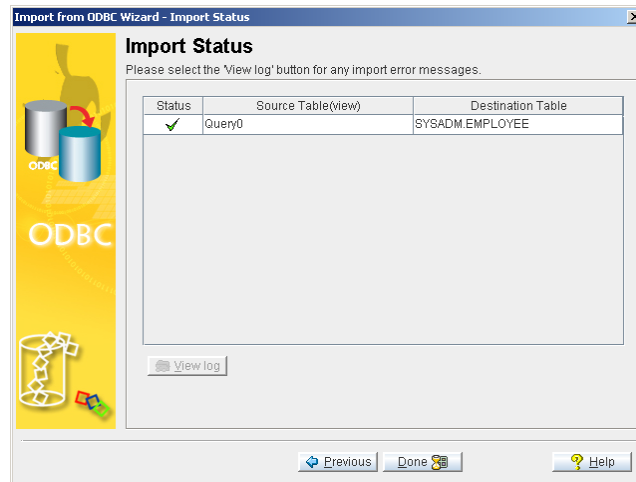
9. Click **OK** to return to the **Source Tables and Views** window.

10. You may also choose to save the map of the import ODBC schema to an XML file by clicking **Save batch**. The **Save Batch File** will open.



11. Select or create an XML file to save the imported ODBC map schema to. Click **Save Batch File** to create the XML file. The **Source Tables and Views** window will reappear.

12. Click **Execute** to import the source data. The Import Status window will appear.



13. If errors appear, click **View log** and scroll to the bottom to see the error message. If no errors occurred, click **Done**.



## 6.4 Exporting Data to Text

Data may be exported from the database to form a structured text file. This section describes the different text file formats that the Data Transfer tool can produce. Consider the following settings affecting the format of the text file before exporting data.

- **Row Delimiter:** Determines the type of character that signifies a break between the rows of a table. Possible characters: {CR/LF} (Carriage return / line feed. In Windows applications, a new line in the text is normally stored as a pair of CR LF characters. In UNIX applications, a new line is normally stored as a LF character. Some applications use only a CR character to store a new line), {CR}, {semicolon} (;), {comma} (,),{tab}, {vertical bar} (|), {semicolon}{LF}, or {comma}{LF}.
- **Column Delimiter:** Determines the type of character that signifies a break between columns in each row. Possible characters: semicolon, comma, or vertical bar.
- **Text Qualifier:** Determines how each tuple of any data type except BINARY, LONGVARBINARY, or numeric data types (integer, smallint, serial, decimal, double, float) is enclosed. Possible values: none, single quote, or double quote.
- **Binary Qualifier:** Determines how each tuple of BINARY or LONGVARBINARY type data is enclosed. Possible values: none, single quote, or double quote.
- **Binary Padding:** A character appended to each tuple containing binary data.
- **Fixed Field:** Instead of using a row delimiter, the text file may be formatted with fixed fields. This means a number of spaces, or fields, defines each column.
- **Include column name:** The first line in a text file may be used to define the column names. The format is "*column1*". "*table name*". "*owner name*"; "*column2*". "*table name*". "*owner name*"; etc. In this case the column delimiter is set to semicolon (;).
- **Include table schema:** The first line in a text file may be used to define the column schema (or the second line if the first line was used to define column names). The format is *data type(scale,precision);data type(scale,precision)* etc. In this case the column delimiter is set to semicolon (;).
- **Use NULL to display null data:** Columns that contain no data output "NULL".
- **File link name for FILE type data:** The file name for system or user file objects is displayed.
- **Use escape character "/":** This character is used when qualifiers or delimiter characters appear in the data. If the data contains a reserved character, the reserved character will be enclosed by an escape character (/).
- **Use temp files to store LONGVARBINARY or LONGVARCHAR type data column content:** BLOB data is stored as a separate, linked file (as a file object), and the name of the file containing the BLOB is displayed.

Data can be exported by selecting individual columns from a table, or through a valid SQL SELECT statement. For more information on SQL syntax, refer to the *SQL Command and Function Reference*.

➤ Example

The following is text produced by the Data Transfer Tool Export to Text wizard. The table's name and schema will be displayed on the first two lines. The data has been exported with fixed fields to delimit columns and CR to delimit rows. File link names appear where LONGVARCHAR data once stored as BLOBs in the table have been exported as file objects.

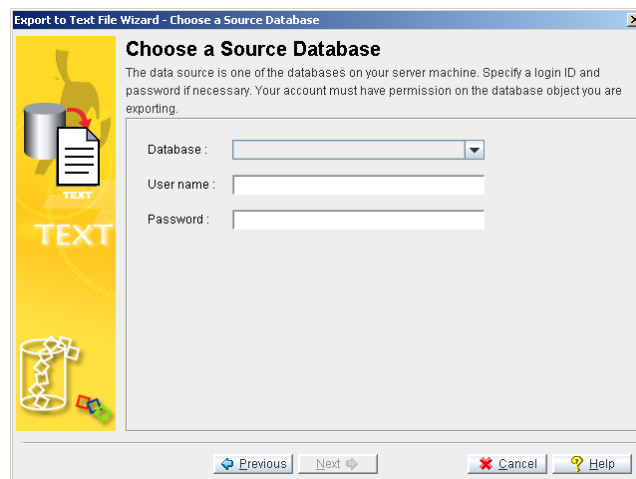
SQL_CHAR(10)	SQL_TIMESTAMP	SQL_LONGVARCHAR	SQL_DECIMAL(10, 3)
"SYSADM"."SUPPORTQUERIES"."LOGINID"	"SYSADM"."SUPPORTQUERIES"."REQUESTTIME"		
"SYSADM"."SUPPORTQUERIES"."ATTACHMENT"	"SYSADM"."SUPPORTQUERIES"."DECIMAL_C"		
A_HOWARD	2001-09-09 12:47:05.000	C:\WEBDB\FO\ZZ000000.GIF	10.250
A_HOWARD	2001-09-22 10:14:21.000	C:\WEBDB\FO\ZZ000001.GIF	13.550
A_HOWARD	2001-10-04 16:22:06.000	C:\WEBDB\FO\ZZ000002.GIF	27.333
A_HOWARD	2001-10-09 17:44:56.000	C:\WEBDB\FO\ZZ000003.GIF	16.140
A_HOWARD	2001-10-12 09:12:38.000	C:\WEBDB\FO\ZZ000004.GIF	88.847
A_HOWARD	2001-10-31 23:16:11.000	C:\WEBDB\FO\ZZ000005.JPG	841.336

➤ To export a table to a text file

1. Open the Data Transfer Tool.
2. Select **Export Text File** from the main console or the **Transfer** menu. The Welcome to Import from Text File Wizard window will open, displaying a summary of the steps to be taken in the wizard.

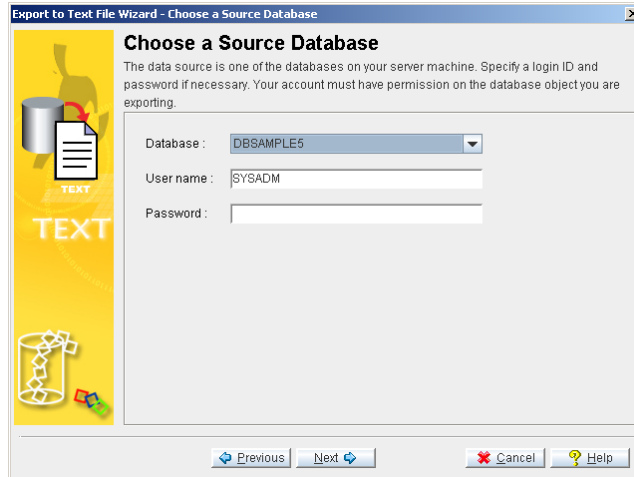


3. Click **Next**. The **Choose a Data Source** window will open.

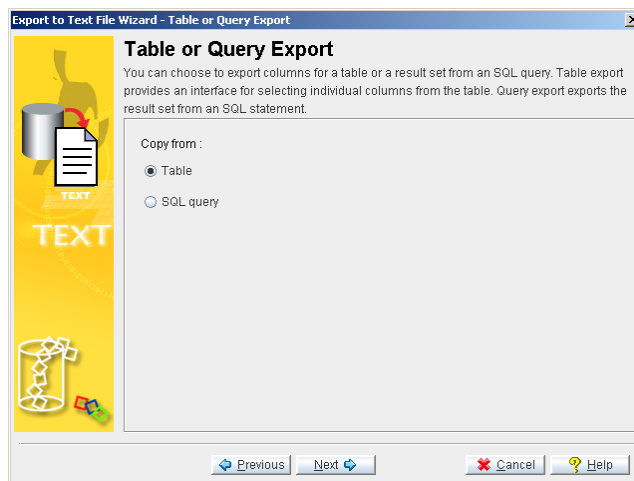


4. Select the database to export data from in the **Database** menu.
5. Enter a user name and password into the appropriate fields.

**NOTE** A user must have *SELECT* privilege to export a text file.



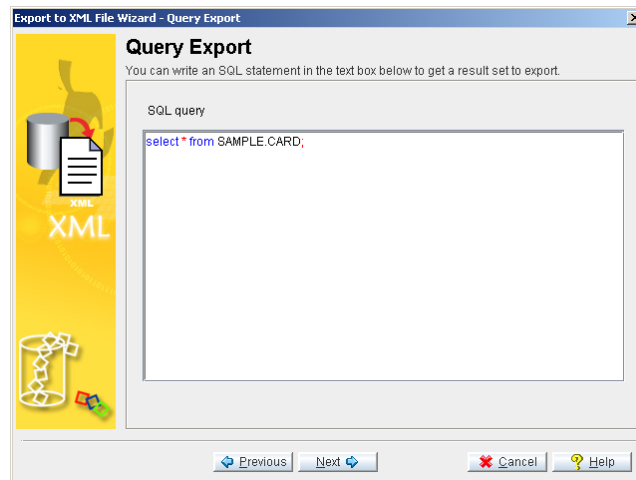
6. After you have selected a database, click **Next**, the Table or Query Export window will open.



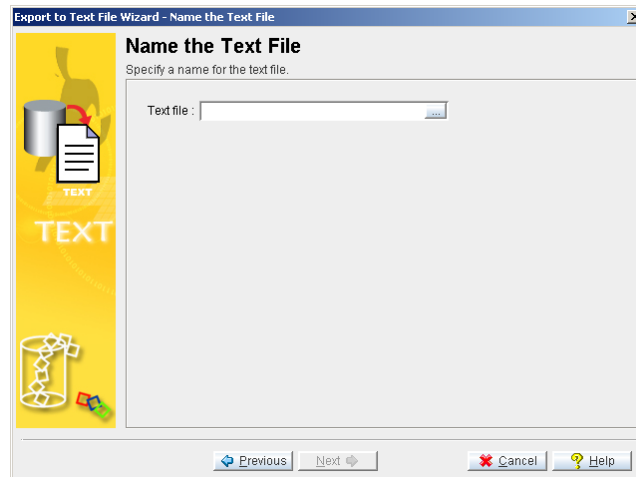
7. Select **Table** to export data from a table. Select **SQL query** to export data from the result set of an SQL SELECT statement.
8. If you selected **Table** from the **Table or Query Export** window, the **Table Export** window will open. If you selected **SQL query**, then proceed to step 13.
9. Click **Next**, the **Table Export** window will open.



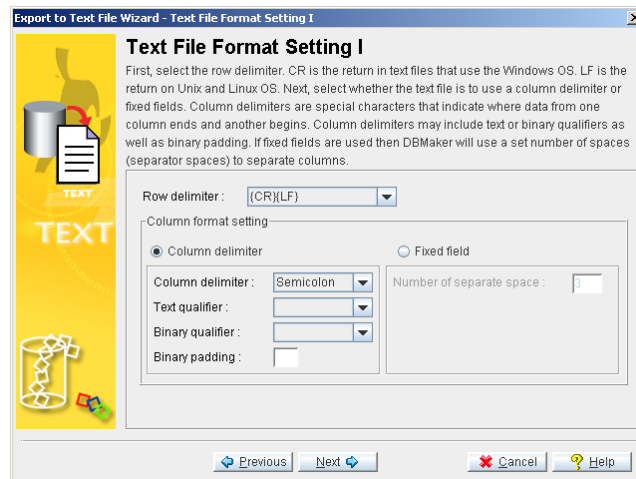
10. Select a table to export from the **Table** name menu. A list of columns in the table will appear in the **Select columns** to export field.
11. Select columns by clicking on the column name and clicking **Add**, or select all columns by clicking **Add All**. Selected column names will appear in the right hand field.
12. Click **Next**. The **Name The Text File** window will appear (proceed to step 16).
13. If you selected **SQL query** from the **Table or Query Export** window, the **Query Export** window will open.
14. Enter a valid SQL select statement into the **SQL query** field.



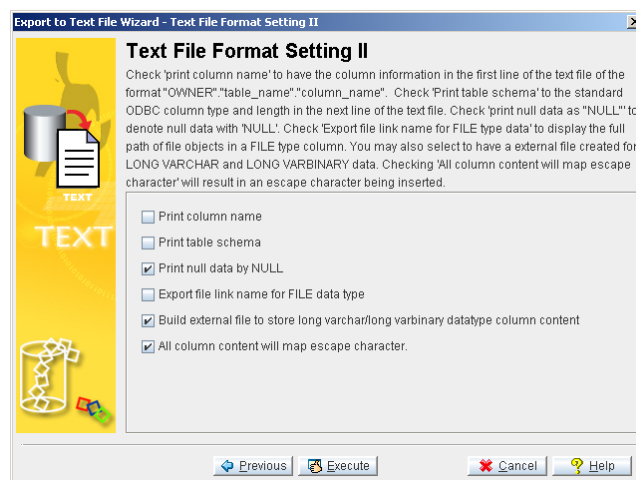
15. Click **Next**. The **Name The Text File** window will appear.



16. Enter the full path of a text file in the **Text file** field, or select one using the browse button.
17. Click **Next**. The **Text File Format Setting 1** window will open.



18. Select the appropriate settings for the format of the text file you will create.
19. Click **Next**. The **Text File Format Setting 2** window will open.



20. Finish selecting the appropriate settings for the format of the text file you are creating. Click **Execute** to export the data to the text file. A confirmation dialog box will appear.
21. Click **OK**

## 6.5 Exporting Data to XML

DBMaster supports the export of data from a table to an XML file. Columns may be stored as individual elements, or as attributes of the table element. When an XML file is created, an associated DTD file is created. The DTD contains information necessary for defining the elements and attributes of the XML file. The structure of both the DTD and XML file will vary depending on whether the columns are stored as attributes or elements.

Consider how the following settings affect the XML file produced by the Export to XML wizard.

- **Column as Element:** If columns are represented as elements in the resultant files, then schema information will be retained as element attributes (data type, column name, length, etc.) in the DTD. Columns are child elements, and the table is represented as the parent element. If the XML file is later imported back into the database, then the table's structure will be exactly replicated. File objects are referenced as entities in the DTD file if Column as Element is chosen.
- **Column as Attribute:** Columns are represented as attributes of the table element in the DTD. There is no record of the table's schema. An element in the XML file represents each record.
- **Export file link name for FILE type data:** The original full path will reference system and user file objects if this option is selected. If this option is not selected, file type data will be treated as Long Varbinary.
- **Translate all tag names to uppercase:** All tag names are converted to uppercase characters.
- **Build temp file to store LONGVARCHAR and LONGVARBINARY data type column constant:** If this option is chosen, BLOB data will be stored in a temporary directory under the directory the XML file resides in. If this option is not selected, BLOB data is stored directly in the XML file.
- **XML file cannot include DTD file reference:** if this option is selected, no DTD is created. No information about the elements will be preserved in the DTD if this option is selected.

### ➡ Example 1

Assume the table 'supportqueries' with columns 'LOGINID' CHAR(10); 'REQUEST' SQL\_LONGVARCHAR; 'REQUESTTIME' SQL\_TIMESTAMP; 'ATTACHMENT' 'SQL\_FILE'; 'BINARY\_C' SQL\_BINARY(10); 'DECIMAL\_C' SQL\_DECIMAL(10, 3). The table has two records. The entire table is exported to an XML file with columns as elements. File link names are exported, temp files are built to store BLOB data, and the DTD is included. The resulting XML file follows:

```
<?xml version="1.0" encoding="BIG5"?>
<!DOCTYPE WEBDB SYSTEM "Support.dtd">
<WEBDB>
  <SUPPORTQUERIES>
    <LOGINID>A_HOWARD          </LOGINID>
    <REQUEST>&BLBTMP_TXT0;</REQUEST>
    <REQUESTTIME>2001-09-09 12:47:05.000</REQUESTTIME>
    <ATTACHMENT>&DBMASTER_FO_0;</ATTACHMENT>
    <BINARY_C>10000000000000000000</BINARY_C>
    <DECIMAL_C>10.250</DECIMAL_C>
  </SUPPORTQUERIES>
```

```

<SUPPORTQUERIES>
  <LOGINID>A_HOWARD          </LOGINID>
  <REQUEST>&BLBTMP_TXT1;</REQUEST>
  <REQUESTTIME>2001-09-22 10:14:21.000</REQUESTTIME>
  <ATTACHMENT>&DBMASTER FO_1;</ATTACHMENT>
  <BINARY_C>2000000000000000000</BINARY_C>
  <DECIMAL_C>13.550</DECIMAL_C>
</SUPPORTQUERIES>
</WEBDB>

```

The associated DTD follows:

```

<!ELEMENT SUPPORTQUERIES (LOGINID, REQUEST, REQUESTTIME, ATTACHMENT, BINARY_C,
DECIMAL_C)>
<!ELEMENT LOGINID (#PCDATA)>
  <!ATTLIST LOGINID
    TYPE CDATA #FIXED "SQL_CHAR"
    NAME CDATA #FIXED "LOGINID"
    LENGTH CDATA #FIXED "20"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >
<!ELEMENT REQUEST (#PCDATA)>
  <!ATTLIST REQUEST
    TYPE CDATA #FIXED "SQL_LONGVARCHAR"
    NAME CDATA #FIXED "REQUEST"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >
<!ELEMENT REQUESTTIME (#PCDATA)>
  <!ATTLIST REQUESTTIME
    TYPE CDATA #FIXED "SQL_TIMESTAMP"
    NAME CDATA #FIXED "REQUESTTIME"
    STORAGE CDATA #FIXED "29"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >
<!ELEMENT ATTACHMENT (#PCDATA)>
  <!ATTLIST ATTACHMENT
    TYPE CDATA #FIXED "SQL_FILE"
    NAME CDATA #FIXED "ATTACHMENT"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >
<!ELEMENT BINARY_C (#PCDATA)>
  <!ATTLIST BINARY_C
    TYPE CDATA #FIXED "SQL_BINARY"
    NAME CDATA #FIXED "BINARY_C"
    LENGTH CDATA #FIXED "10"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >

```

```

>
<!ELEMENT DECIMAL_C (#PCDATA)>
  <!ATTLIST DECIMAL_C
    TYPE CDATA #FIXED "SQL_DECIMAL"
    NAME CDATA #FIXED "DECIMAL_C"
    LENGTH CDATA #FIXED "(10, 3)"
    ISNULL (true|false) 'true'
    xml:space (default|preserve) 'preserve'
  >
<!ENTITY BLBTMP_TXT0 SYSTEM "blobtmpdir0\blbtmpf0.txt">
<!ENTITY DBMASTER_FO_0 SYSTEM "C:\DBMASTER\5.1\BIN\WEBDB\FO\ZZ000000.GIF">
<!ENTITY BLBTMP_TXT1 SYSTEM "blobtmpdir0\blbtmpf1.txt">
<!ENTITY DBMASTER_FO_1 SYSTEM "C:\DBMASTER\5.1\BIN\WEBDB\FO\ZZ000001.GIF">
<!ENTITY BLBTMP_TXT2 SYSTEM "blobtmpdir0\blbtmpf2.txt">
<!ELEMENT WEBDB (SUPPORTQUERIES*)>

```

### ➡ Example 2

Given the same table as example 1, but with the entire table exported with columns as attributes. The resulting XML file follows:

```

<?xml version="1.0" encoding="BIG5"?>
<!DOCTYPE WEBDB SYSTEM "Support.dtd">
<WEBDB>
  <SUPPORTQUERIES
    LOGINID="A_HOWARD&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;"
    REQUESTTIME="2001-09-09 12:47:05.000"
    ATTACHMENT="C:\DBMASTER\5.1\BIN\WEBDB\FO\ZZ000000.GIF"
    BINARY_C="10000000000000000000"
    DECIMAL_C="10.250" />
  <SUPPORTQUERIES
    LOGINID="A_HOWARD&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;&#x20;"
    REQUESTTIME="2001-09-22 10:14:21.000"
    ATTACHMENT="C:\DBMASTER\5.1\BIN\WEBDB\FO\ZZ000001.GIF"
    BINARY_C="20000000000000000000"
    DECIMAL_C="13.550" />

```

The associated DTD follows:

```

<!ELEMENT SUPPORTQUERIES EMPTY>
  <!ATTLIST SUPPORTQUERIES
    LOGINID CDATA #IMPLIED
    REQUESTTIME CDATA #IMPLIED
    ATTACHMENT ENTITY #IMPLIED
    BINARY_C CDATA #IMPLIED
    DECIMAL_C CDATA #IMPLIED
  >
<!ELEMENT WEBDB (SUPPORTQUERIES*)>

```

### ➡ To export a table to an XML file:

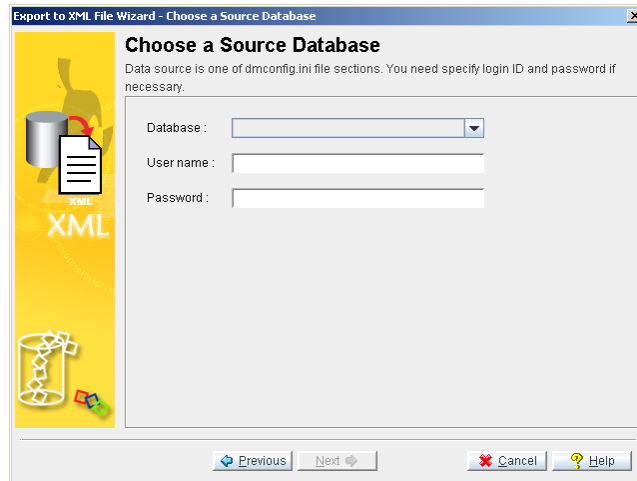
1. Open the Data Transfer Tool.



2. Select **Export to XML** from the main console or the **Transfer** menu. The **Welcome to Export to XML File Wizard** window will appear.

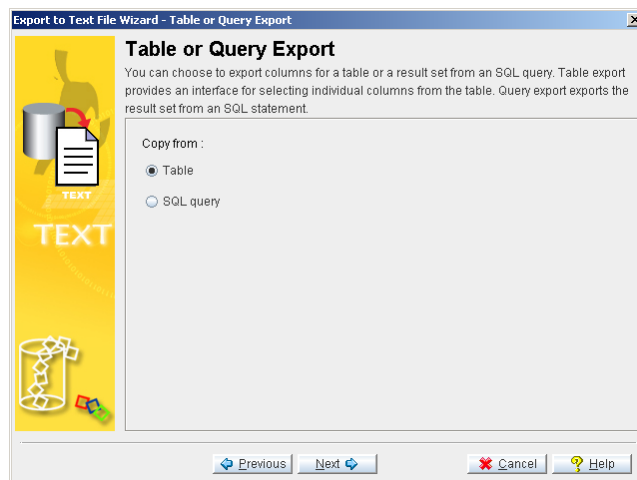


3. Click **Next**. The **Choose a Data Source** window will open.



4. Select a database from the **Database** menu. Enter a user name and password into the appropriate fields.

5. Click **Next**. The **Table or Query Export** window will appear.



6. If you selected **Table** from the **Table or Query Export** window, the **Table Export** window will open. If you selected **SQL query**, then proceed to step 13.

7. Click **Next**, the **Table Export** window will open.



8. Select a table to export from the **Table name** menu. A list of columns in the table will appear in the **Select columns to export** field.

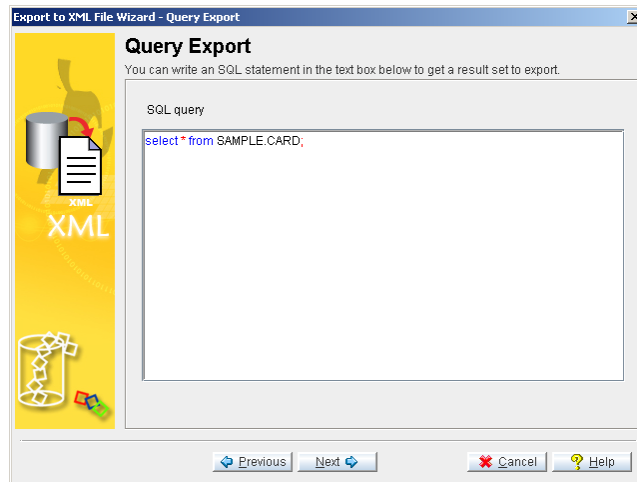
9. Select columns by clicking on the column name and clicking **Add**, or select all columns by clicking **Add All**. Selected column names will appear in the right hand field.



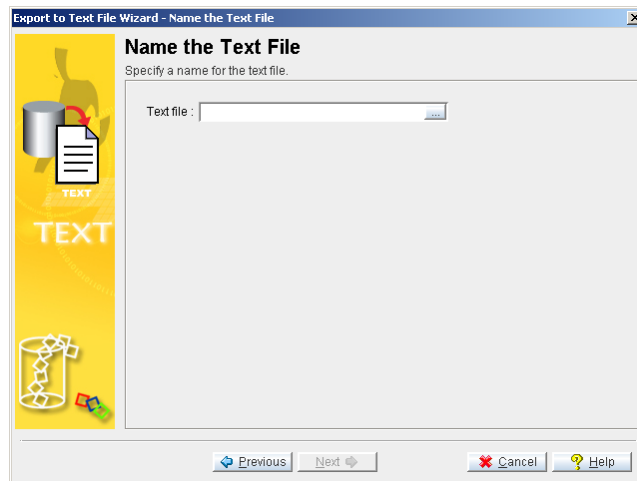
10. Click **Next**. The **Name The XML File** window will appear (proceed to step 16).

11. If you selected **SQL query** from the **Table or Query Export** window, the **Query Export** window will open.

12. Enter a valid SQL select statement into the **SQL query** field.

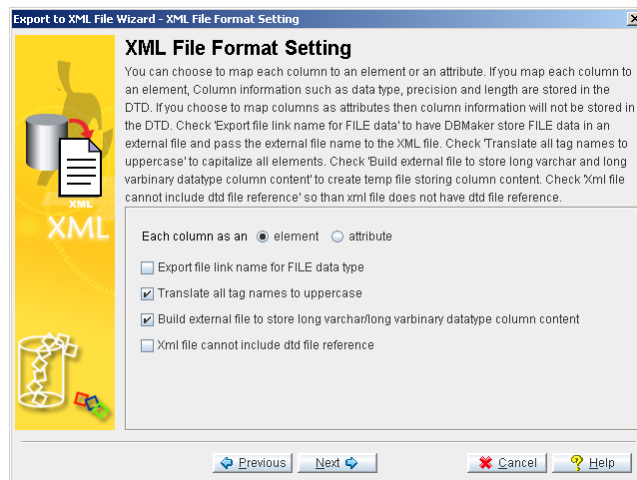


13. Click **Next**. The **Name The XML File** window will appear.



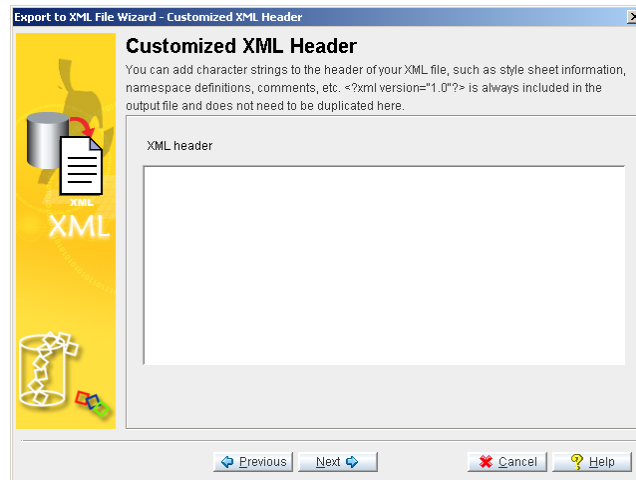
14. Enter the full path of an XML file to export to, or select one by using the browse button.

15. Click **Next**. The **XML File Format Setting** window will open.



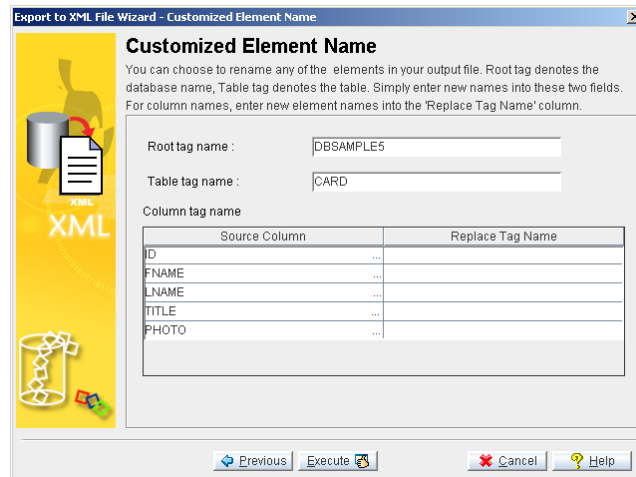
16. Select the appropriate settings for the format of the XML file you will create.

17. Click **Next**. The **Customized XML Header** window will open.



18. Enter appropriate information, such as namespace and style sheet definitions, if relevant.

19. Click **Next**. The **Customized Element Name** window will appear.



20. It is possible to modify the tag definitions. Enter new tag definitions into the **Replace Tag Name** column. The name of the corresponding column will be changed in the resulting XML file.

21. Click **Execute** to export the table to the XML file. A confirmation window will appear.

22. Click **OK**.

## 7. UNLOAD/LOAD

Sometimes the user may need to save database data to an external text file. DBMaster provides the UNLOAD and LOAD commands just for this purpose. UNLOAD and LOAD command can be seen as another way to backup and restore a database.

UNLOAD is a tool provided by dmSQL used to transfer the contents of a database to an external text file. After the unload procedure succeeds, dmSQL will produce two text files. One stores the script, with extension name s0, to establish the database object and the other stores the BLOB data, with the extension name bn. Objects that are unloaded from the database are not removed from the database; they are simply saved as one or more external text files.

The LOAD command is also a tool provided by dmSQL, it is used to transfer a database object, already unloaded to a text file, into the database. When an object is loaded onto a database, the schema of that object is also recreated.

## 7.1 UNLOAD

There are eight options for the unload command: unload database, unload table, unload schema, unload data, unload project, unload module, unload procedure, and unload procedure definition. Only unload the object that you have the select privilege on. For instance, if you have the select privilege on a table, then you can only unload the content of this table. Only a DBA or a SYSADM may unload the database.

To unload tables with names containing wildcards like the escape character "\", or double quotes on the name.

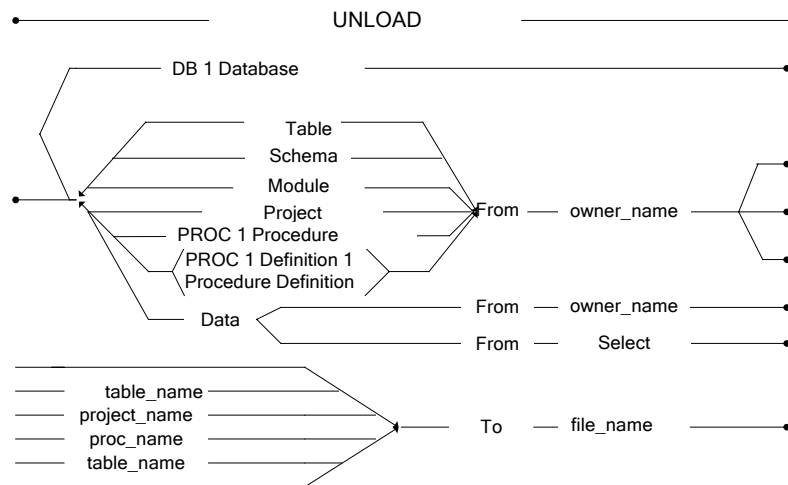


Figure 7-1 UNLOAD syntax

### 7.1.1 UNLOAD DB [DATABASE]

A DBA or a SYSADM may unload the content of a database to an external text file. This file includes information about security, tablespaces, definitions, indices, synonyms, data, etc. For each database, dmSQL will generate at least two external files, one script, and one BLOB data.

The name of the external text file is **empdb**. By default, dmSQL will create these files in the current working directory. In the statement below, there are at least two text files created, empdb.s0 and empdb.b0. If the unloaded BLOB file empdb.b0 exceeds the maximum size allowed by the operating system, dmSQL will generate empdb.b1, empdb.b2 through to empdb.bn sequentially up to a maximum number of 99. dmSQL will always generate one script file emodb.s0, and its maximum size is set to the operating system limitation.

#### ➤ Example

```
dmSQL> unload db to empdb;
```

## 7.1.2 UNLOAD TABLE

Unloads tables to an external file and will record the definition, synonyms, indices, primary key, foreign keys, and data of the table.

Use the wild cards “\_” and “%”, which is similar with “?” and “\*” in DOS, in the owner and table name. The wild card “\_” represents a character, and “%” represents a set of characters.

### ➤ Example 1:

The following will unload the table **e tab** for the current user; if there are any blanks in the table name add double quotes:

```
dmSQL> unload table from "e tab" to empfile;
```

### ➤ Example 2:

The following will unload all tables with the names starting with **emp** for the SYSADM owner, for example, **emptab**, **empname**, &ldots; etc:

```
dmSQL> unload table from SYSADM.emp% to empfile;
```

## 7.1.3 UNLOAD SCHEMA

The usage of this option is very similar with unload table. It can only unload the definition of a table, and does not unload the data in a table. Uses the same wild cards as illustrated in the above unload table option.

### ➤ Example:

The following will unload the schema of all tables with the name **ktab**:

```
dmSQL> unload schema from %.ktab to kfile;
```

## 7.1.4 UNLOAD DATA

This option will unload all data from a table and does not unload the definition of the table. Unload data uses the same wildcards as the previous two options. Only users with the SELECT privilege on the unloaded table may execute the unload data command.

DBMaster 3.6 and later versions support an additional syntax for unloading data: **dmSQL>unload data from (select statement) to file\_name**. If the select statement is a join, the projection columns must be from the same table, the following statement is executable. DDL commands, delete, insert, or updates are not permitted.

### ➤ Example 1

Valid syntax

```
dmSQL> unload data from (select t1.c1, t1.c2 from t1, t2 where t1.c1= t2.c1) to f1;
```

### ➤ Example 2

Illegal syntax

```
dmSQL> unload data from (select t1.c1, t2.c1 from t1, t2 where t1.c1 = t2.c1) to f1;
```

### ➤ Example 3

Illegal syntax, no aggregate or built-in functions are permitted in the projection columns.

```
dmSQL> unload data from (select avg(c1) from t1) to f1;
```

```
dmSQL> unload data from (select now() from t1) to f1;
```

➤ Example 4

Valid syntax, views and synonyms are permitted.

```
dmSQL> unload data from (select * from s1 where c1 > 10) to f1;  
dmSQL> unload data from (select * from v1 where c1 < 10) to f1;
```

## 7.1.5 UNLOAD PROJECT

---

This option allows a user to unload a project to an external text file.

## 7.1.6 UNLOAD MODULE

---

This option allows a user to unload a module to an external file.

## 7.1.7 UNLOAD [PROC | PROCEDURE]

---

This option allows a user to unload the stored procedures to an external file. When user issues unload PROC command in dmSQL tool, the procedure definition will be unloaded into script files. User must also make sure they have installed the required C compiler on the new database server side before load procedure.

Users can get more information from '*Stored Procedure User's Guide*'.

➤ Example:

The statement unloads the stored procedure **select\_employee**. After unloaded, it works out two text files: select\_employee.b0 and select\_employee.s0 to file d:\workdir\sp\_file.

```
dmSQL> unload procedure from select_employee to 'd:\workdir\sp_file'
```

## 7.1.8 UNLOAD [PROC DEFINITION | PROCEDURE DEFINITION]

---

This option allows a user to unload the definition of the stored procedure to an external text file. Users can get more information from '*Stored Procedure User's Guide*'.

In most case, user should use unload proc to unload their procedure. In some case, user already develop their stored procedure in the development environment and want to copy the stored procedure into user's environment that have no C compiler installed.

In this case, user can do as follows:

1. Use 'unload stored procedure definition'.
2. Copy all the stored procedure's dynamic linking library as the directory in the developing environment to user's environment.
3. Load the procedure by LOAD PROCEDURE syntax.

➤ Example

Users who use unload procedure definition can restore the SP which was created last time

The statement unload the stored procedure begin with sp\_, for example sp\_brows, sp\_get1 and so on.

```
dmSQL> unload procedure definition from sysadm.sp_% to d:\sp_file1
```



## 7.2 LOAD

There are seven options: load database, load table, load schema, load data, load project, load module, and load procedure. Only load the file that is unloaded in the same option. For example, load a database from the text file that is unloaded with database option.

When loading a text file, set the number of commands to automatically commit the transaction. The default number is 1000. The size of  $n$  will affect whether the transaction succeeds or not and the speed of loading. The Journal will fill easily with a large  $n$  value and could cause the transaction to fail. A small  $n$  value will increase the commit times and slow down the speed of loading.

If there are errors occurring during the loading procedure, an error messages will be recorded in a log file, which the system will use to undo executed commands. The log file is stored in the same directory as the external text file being loaded and does not stop the loading procedure.

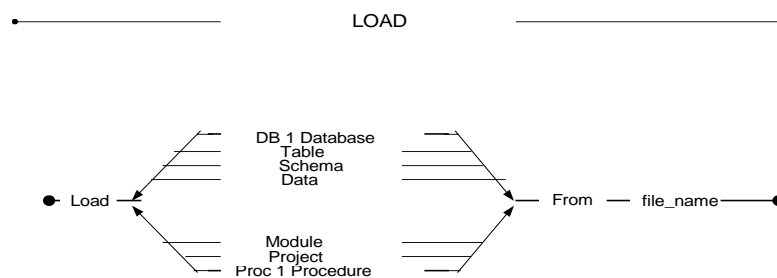


Figure 7-2 LOAD syntax

### 7.2.1 LOAD DB [DATABASE]

Use the command to transfer the contents of a database to a new database. First, unload the database to transfer to an external text file, and then use the "load db" command to load the contents of the database from the text file. Before loading a database, create a new one. The name of the new database can be different from the old one. Only a DBA or a SYSADM may execute this command.

The utility will work in Journal mode if the loaddb is set in safe mode. The load utility will rollback to the last committed command if any error occurs during loading, the error messages will return to screen, and write to the log file of the load utility.

When using the set loaddb in fast mode, the rule for loading the utility in DBMaster versions earlier than 3.6, will make the whole load procedure work under the no Journal mode. Setting loaddb in fast mode will speed up the load utility, but it will make the database shut down in no Journal mode if any error occurs.

For example, suppose that the load file has tablespace creation but it is not specified in the **dmconfig.ini** file. If **loaddb** is set to use the safe option, the following error message, "ERROR (8002): [DBMaster] keyword entry is required for configuration file", will be reported and then the load command will rollback. If **loaddb** is set to use the fast option, then the following error message occurs, "ERROR (30017), [DBMaster] errors occurred on no-Journal mode, shut down database". The default option is "set loaddb safe" in DBMaster Configuration file.

➡ Example1:

The following set option for **loaddb** has been added to versions above DBMaster 3.6.

```
Set loaddb [safe | fast]
```

➡ Example2:

The following command loads the database from a file named "**empdb**", and commits it automatically every **100** commands during loading. The system will generate a log file named "**empdb.log**" in the same directory.

```
dmSQL> load db from empdb 100;
```

## 7.2.2 LOAD TABLE

---

The option permits loading the contents of a table, including schema and data, from a text file. When loading a table from a text file, make sure that the table name is unique.

➡ Example:

The following command will load a table from a file named "**empfile**", and it will commit automatically every **50** commands during loading.

```
dmSQL> load table from empfile 50;
```

## 7.2.3 LOAD SCHEMA

---

The option allows users to load the schema, not including the data, from a table contained in a text file. When loading a table schema from a text file, ensure that the table name is unique.

## 7.2.4 LOAD DATA

---

A corresponding table must exist when loading data from an external text file. In versions earlier than 3.6 when the errors occur during the LOAD DATA procedure, it will rollback to the last committed command.

If *load data skip error*, is set then the following error messages will be skipped during the loading of data:

ERROR (401) unique key violation

ERROR (410) referential constraint violation: value does not exist in parent key

ERROR (6521) table or view does not exist

ERROR (6002) syntax error

ERROR (6015) incomplete SQL statement input

The error will be skipped and the load utility will resume execution of subsequent commands. The above errors are the most common errors to occur during loading of data. When the load data

stop or stop on error is set, the whole load command will rollback if errors occur. The default value for this option is set *load data skip [error]*. All the error messages occurred during the loading of data will be written into the log file.

➡ Example1:

DBMaster 3.6 and later versions support the following options.

```
Set loaddata skip [error] | stop [on error]
```

➡ Example2:

The following command will permit the loading of data from an external data file named “**datafile**” and will commit automatically every **1000** commands using the default setting.

```
dmSQL> load data from datafile;
```

---

## 7.2.5 LOAD MODULE

---

The option allows a user to load a module from an external text file.

---

## 7.2.6 LOAD PROJECT

---

The option allows a user to load a project from an external text file.

---

## 7.2.7 LOAD PROC [PROCEDURE]

---

This option allows a user to load a stored procedure from an external text file.

When user only unload procedure definition, you can use ‘dmSQL> load procedure from file\_name’, the file\_name is same as the file\_name in unload statement.

➡ Example:

The statement load stored procedure from external file of d:\sp\_file1 and use default setup, every 1000 commands commit one time automatically.

```
dmSQL> load procedure from 'd:\sp_file1';
```

## 8. Dmsql Command

In this chapter we introduce the commands: EXPORT and IMPORT, they require the use of DBMaster's DMSQL Tool. These dmsql commands can also be treated as a way about backup/restore through file.

## 8.1 EXPORT

The Export command extract data from tables in database and inserts the data into text files. There are two configurations used. The export command interface is used for specifying command options. The description file is used for specifying the export file format.

### 8.1.1 EXPORT COMMAND INTERFACE

---

The Export command syntax is as follows:

*<data\_file>* This is the target file into which you will insert the data. It should be in full path. If you do not specify *data\_file*, the export file name will be *<table\_name>\_out.txt*.

*TABLE* Please specify the table you want to export.

*[DESCRIPTION <description\_file>]* This is the description file for the data format in the resulting data file. In the description file, users will specify some rules for the resulting data file. Refer to the DESCRIPTION FILE FORMAT section for more information. If the description file is not specified, the description file name will be *<table\_name>\_out.dsc*. If this file does not exist, DBMaster will use the default output format. The default file format will be variable format. That means:

- TAB as column delimiter
- New line character as row terminator
- No quotation marks
- All columns in source table are exported in the same order as they are in the table

*[LOG <log\_file>]* This file logs the errors that occur during the course of unloading data. If this option is not specified, the default log file name, *export.log*, will be used.

*[STOP\_ON\_ERROR]* it specifies that you want to stop unloading data if an error occurs. If this option is not specified, the unloading of data will continue even if an error has occurred.

```
EXPORT
[INTO <data_file>]
TABLE [<owner_name>.<table_name>]
[DESCRIPTION <description_file>]
[LOG <log_file>]
[STOP_ON_ERROR]
```

### 8.1.2 DESCRIPTION FILE

---

You can specify the format of the description file for formatting the unloading result. Two types of format can be used, fixed format and variable format.

#### FIXED FORMAT DESCRIPTION FILE

When the fixed format description file is used, users want each column of the export result to be aligned vertically. The separators used for alignment will be space characters.

*FORMAT = FIXED* This specifies the description file format for fixed length data files.

*[LOB\_FORMAT= INTERNAL | EXTERNAL]* This specifies that when exporting columns of large object types (such as blob, clob, nclob, nblob and other files) external files will be generated. For each column of large object type in each row, an external file will be generated. If this option is not specified, the content of data will be embedded in a data file.

*blobtempdir<m>\blbtmpf<n>.<tmp | txt>*. When naming external files it's important to keep the following in mind: *blobtempdir<m>\blbtmpf<n>.<tmp | txt>*.

*m* specifies the minimum un-used number counted from 1 in the directory.

For example, if there are already directories named blobtempdir1, blobtempdir2 and blobtempdir3, the newly created directory for containing external files will be blobtempdir4.

*n* specifies the minimum un-used number counted from 1 in the directory.

Whether the file extension name is **tmp** or **txt** depends on whether the exported column is BLOB type, FILE type or CLOB type. If the column type is BLOB or FILE, the file extension name will be **tmp**. Otherwise, the column type is **txt**.

*server\_column\_name* This lists the names of the source table columns that are going to be exported from the database. If there are spaces in table name, use double quotes to enclose the column names.

*column\_position* Specifies the column byte position in data file.

*server\_columnname* and *column\_position* are separated by space character(s). *column\_position* is specified by two numbers that are separated by (:). For example a 1:40 means the data loader should look for data from 1<sup>st</sup> byte to 40<sup>th</sup> byte in data file. We will use space characters to align the data field vertically. If the data in the source table exceeds the field length, the data output will be truncated.

```
FORMAT=FIXED
[LOB_FORMAT=INTERNAL | EXTERNAL]
<server_column_name> <column_position>
```

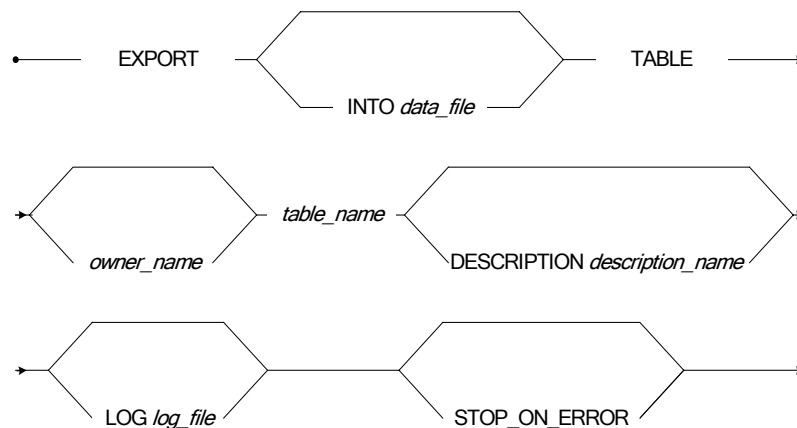


Figure 7-1 EXPORT syntax

## VARIABLE FORMAT DESCRIPTION FILE

When variable format description file is chosen, the fields of resulting data output will be separated by a user specified delimiter.

*FORMAT=VARIABLE* This specifies that the resulting output file is in variable format.

*[COLUMN\_DELIMITER=<delimiter>]* This specifies a character that separates each column in datafile. The character should be single quoted. For example, to indicate that a **SPACE** is used as column delimiter, use ' '. Aside from normal characters, take the following escape sequences that represent special characters.

CHARACTER	ESCAPE SEQUENCE REPRESENTATION
TAB	\t
NEW LINE	\n

For example, if the delimiter is a TAB, users will use '\t' in <delimiter>. If the column delimiter is not specified, we will use TAB (\t) as the column delimiter. Use discretion when choosing a delimiter.

If the number of column delimiters is fewer than the number of target table columns specified by users, NULL will be used for the insert value.

*[ROW\_TERMINATOR=<row\_terminator>]* This string denotes the end of a row.

*[QUOTATION=SINGLE\_QUOTE | DOUBLE\_QUOTE]* This indicates that the output data will be quoted by either single quotes or double quotes. If there is quotation mark in the data, the output will show two consecutive quotation marks.

*[LOB\_FORMAT=INTERNAL | EXTERNAL]*: This specifies that when exporting columns of large object types, such as blob, clob, nclob, nblob and other large files, external files will be generated. For each column of large object type in each row, an external file will be generated. If this option is not specified, the content of the data will be embedded in a data file.

When naming external files it's important to keep the following in mind:

**blobtempdir<m>\blbtmpf<n>.<tmp | txt>**

**m** specifies the minimum un-used number counted from 1 in the directory.

For example, if there are already directories named blobtempdir1, blobtempdir2 and blobtempdir3, the newly created directory for containing external files will be blobtempdir4.

**n** specifies the minimum un-used number counted from 1 in the directory.

Whether the file extension name is **tmp** or **txt** depends on whether the exported column is BLOB type, FILE type or CLOB type. If the column type is BLOB or FILE, the file extension name will be **tmp**. Otherwise, the column type is **txt**.

This variable lists the names of columns of a server table which are to be exported. The order of these names represents the order of column export. If there is no such list, all the columns in source table will be export in the same order as that of table columns.

```
FORMAT=VARIABLE
[COLUMN_DELIMITER=<delimiter>]
[ROW_TERMINATOR=<row_terminator>]
```

```
[QUOTATION=SINGLE_QUOTE | DOUBLE_QUOTE]
[LOB_FORMAT=INTERNAL | EXTERNAL]
[<server_column_name>]
```

## IMPORT/EXPORT DATA RULES

The following table outlines the rules that must be applied when attempting to import or export data to or from a file.

DATA TYPE	IMPORT/EXPORT FORMAT	EXAMPLE
BINARY	Use HEX format	To import the binary number "0x004D2", use 004D2 in datafile
CHAR	Characters are used exclusively	To import the word "inception", use inception in the datafile
VARCHAR	See CHAR data type	
DATE	The format YYYY/MM/DD will be used for exporting	import the date "2003/07/25", use 2003/07/25 in the datafile
TIME	Export and import will use the format HH:MM:SS	To import the time "14:30:25", use 14:30:25 in the datafile
TIMESTAMP	The combination of DATE format and TIME format forms the format of TIMESTAMP	To import the timestamp "2003/07/25 14:30:25", use 2003/07/25 14:30:25 in data file
DECIMAL	Use numeric data representation	To import the number "36.82", use 36.82 in data file
DOUBLE	Use numeric data as described in DECIMAL or scientific notation of numbers	To import the number "13e+12", use 13e+12 in datafile
FLOAT	See DOUBLE	
INTEGER	Use integer data	To import the integer "576", use 576 in datafile
LONG	Two formats can be used:	(1) embedded format:



DATA TYPE	IMPORT/EXPORT FORMAT	EXAMPLE
VARBINARY	<p>embedded or external file format.</p> <p>For embedded format, HEX characters are used.</p> <p>For external file format, the URL is provided.</p> <p>Use description flag LOB_FORMAT to indicate your option. For details see description file specifications.</p>	<p>The format used will be the same as BINARY.</p> <p>Document\GRAPH.G IF (2) external file format:</p> <p>For example, if users want to import a binary file whose full path is "c:\My Document\GRAPH.G IF". The URL provided will be c:\My Document\GRAPH.G IF</p>
LONG VARCHAR	<p>Similar to the case for LONG VARBINARY, two formats can be used. The input data will be in ASCII string instead of HEX string.</p>	<p>(1) embedded format: Same as CHAR format.</p> <p>(2) external file format: Same as LONG VARBINARY.</p>
FILE	<p>For FILE type, import/export will adopt the same rule for LONG VARBINARY.</p>	
OID	<p>Same rule as INTEGER</p>	
SERIAL	<p>Same rule as INTEGER</p>	
SMALLINT	<p>Same rule as INTEGER</p>	
NULL data	<p>For variable format, NULL data is recognized by the fact that there's nothing between two consecutive delimiters.</p> <p>For fixed format, NULL data is recognized by the fact that there are all space characters between columns.</p>	

## 8.2 IMPORT

The Import command is used for extracting data from a text file and then inserting the data into database tables. The import command interface is used for specifying command options. The description file is used for specifying the import file format.

### 8.2.1 IMPORT COMMAND INTERFACE

The Import Command Interface provides you with several options for importing data. Options include controlling the stoppage criteria for data loading, the logging of errors and the data encoding of source data files. The format, of source data files, is described in the description file.

*[<owner\_name>.<table\_name>* This identifies the table to be loaded from the datafile. If you do not specify the *<owner\_name>*, the current connection user will be assigned as the owner.

*[FROM <data\_file>]* This is the actual file that contains data to be loaded. If you do not specify *data\_file*, the datafile name will be *<table\_name>.in.txt*. For example, if the import table name is **t1** and datafile name is not specified in command, the datafile name will be **t1\_in.txt**.

*[DESCRIPTION <description\_file>]* This is the description file for describing the data format in the data file. If this option is not specified, the description file name will be assigned as *<table\_name>.in.dsc*. For example, if the import table name is **t1** and description file is not specified, the description file name will be assigned as **t1\_in.dsc**. If this file is not found, a default description file format will be used, variable description file format.

*[LOG <log\_file>]* This identifies the log file, which logs any errors during the course of data loading. It will show the content of the record, which triggers the error as well as the corresponding error message. If you do not specify this option, the default log name will be *import.log*.

*[STOP\_ON\_ERROR]* The loading of data will stop if an error occurs during the import process if this variable is set. If it is not specified, the loading will continue even when an error occurs.

```
IMPORT [<owner_name>.<table_name>
[FROM <data_file>]
[DESCRIPTION <description_file>]
[LOG <log_file>]
[STOP_ON_ERROR]
```

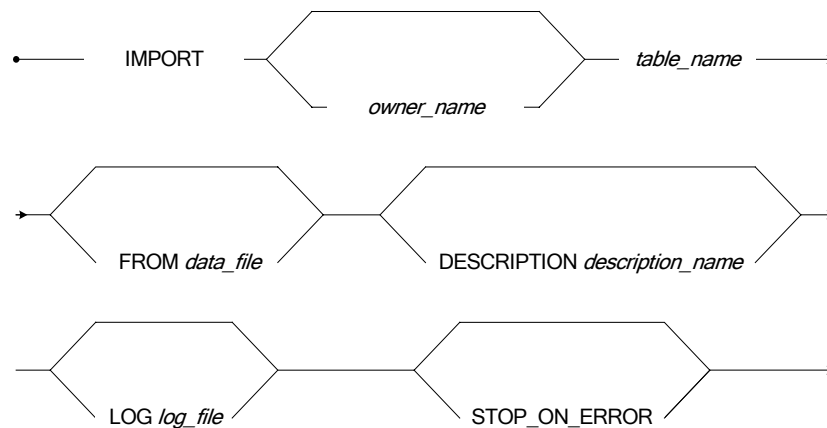


Figure 7-2 IMPORT syntax

## 8.2.2 DESCRIPTION FILE

Two types of description file are used. One is fixed format and the other is variable format. Parse errors in the description file will be shown as clearly as possible. You will know why the error has happened by checking the error message. The error message will display the problem that occurred when parsing a specific word.

### FIXED FORMAT DESCRIPTION FILE

**FORMAT=FIXED** When the format is set to fixed this means the description file describes the format for fixed length data files.

**[START\_WITH\_ROW=<row\_number>]** You can specify from which record you want to start loading data. The default number is 1, if you do not specify this option. If **START\_WITH\_ROW** is greater than total rows of data in data file, no data will be loaded. The **row\_number** is must be a positive number.

**[NUMBER\_OF\_ROWS\_FOR\_EACH\_TRANSACTION=<number>]** This lets you specify the interval of the rows of records loaded between each commit-transaction. If this option is not specified, DBMaster will commit transaction for every 5 rows. If the variable is set at **-1**, there will be no commit. In this case you must commit transaction manually if you want the load to be effective. If the variable is set at **0**, the entire import is seen as a single transaction. The system will then issue a commit after the loading is finished.

The number of rows committed will still count a record even if an error occurs when loading the record.

For example, you set **NUMBER\_OF\_ROWS\_FOR\_EACH\_TRANSACTION=10** and an error occurs when the 4<sup>th</sup> record is loaded. The 1<sup>st</sup> to 3<sup>rd</sup> records and 5<sup>th</sup> to 10<sup>th</sup> records will still be committed and the 1<sup>st</sup> to 10<sup>th</sup> records still seen as one transaction unit. Of course, when **STOP\_ON\_ERROR** is specified, the 5<sup>th</sup> record to 10<sup>th</sup> record won't be committed at all only the 1<sup>st</sup> to 3<sup>rd</sup> records will be committed.

This option is valid only when auto-commit is off.

*[LOB\_FORMAT=INTERNAL | EXTERNAL]* If clob/blob format is internal, the text in data file is seen as the data that is going to be imported. Otherwise, the text is seen as a URL to external files that are going to be imported.

*server\_column\_name* This lists the names of the target table columns that are going to be imported from a data file. If there are spaces or equal signs in the table column name, use double quotes to enclose it.

*column\_position* This is the column byte position in data files.

*server\_column\_name* and *column\_position* are separated by space characters. *column\_position* is specified by two numbers that are separated by (:). For example a 1:40 means the data loader should look for data from 1<sup>st</sup> byte to 40<sup>th</sup> byte in a datafile. Use space characters to align the data field vertically. If the data in the source table exceeds the field length, the rest of row data will be truncated. Each line is terminated by either new line or a carriage return and a new line, depending on whether the loader is a Windows platform. If a line is smaller than the maximum position, spaces will be padded to fill the hole. If a line is longer than the maximum position, the rest of the line is ignored.

```
FORMAT=FIXED
[START_WITH_ROW=<row_number>]
[NUMBER_OF_ROWS_FOR_EACH_TRANSACTION=<number>]
[LOB_FORMAT=INTERNAL | EXTERNAL]
<server_column_name> <column_position>
```

**NOTE** The fields, *server\_column\_name*, and *column\_position* are separated by space characters.

- An example for importing a file with fix format description file is as follows:

The datafile exists as follows:

Davolio Nancy	Sales Representative	Ms.
Fuller Andrew	Vice President, Sales	Dr.
Leverling Janet	Sales Representative	Ms.
Peacock Margaret	Sales Representative	Mrs.
Buchanan Steven	Sales Manager	Mr.
Suyama Michael	Sales Representative	Mr.
King Robert	Sales Representative	Mr.

The description file for this data file may look like this:

```
START_WITH_ROW=1
NUMBER_OF_ROWS_FOR_EACH_TRANSACTION=5
Name 1:20
Position 20:45
Gender 50:54
```

## VARIABLE FORMAT DESCRIPTION FILE

```
FORMAT=VARIABLE
[START_WITH_ROW=<row_number>]
[NUMBER_OF_ROWS_FOR_EACH_TRANSACTION=<number>]
[ {COLUMN_DELIMITER=<delimiter>} ]
[ROW_TERMINATOR=<row_terminator>]
[QUOTATION=SINGLE_QUOTE | DOUBLE_QUOTE]
[ESCAPE_CHAR=YES|NO]
```

```
[LOB_FORMAT=INTERNAL | EXTERNAL]
[<server_column_name> <column_number>]
```

**FORMAT=VARIABLE** This means this file contains the format for variable length description files.

**[START\_WITH\_ROW=<row\_number>]** You can specify from which record you want to start loading data. The default number is 1, if you do not specify this option. If **START\_WITH\_ROW** is greater than total rows of data in datafile, no data will be loaded. The *row\_number* is must be a positive number.

**[NUMBER\_OF\_ROWS\_FOR\_EACH\_TRANSACTION=<number>]** This lets you specify the interval of the rows of records loaded between each commit-transaction. If this option is not specified, DBMaster will commit transaction for every 5 rows. If the variable is set at **-1**, there will be no commit. In this case you must commit transaction manually if you want the load to be effective. If the variable is set at **0**, the entire import is seen as a single transaction. The system will then issue a commit after the loading is finished.

The number of rows committed will still count a record even if an error occurs when loading the record.

For example, you set **NUMBER\_OF\_ROWS\_FOR\_EACH\_TRANSACTION=10** and an error occurs when the 4<sup>th</sup> record is loaded. The 1<sup>st</sup> to 3<sup>rd</sup> records and 5<sup>th</sup> to 10<sup>th</sup> records will still be committed and the 1<sup>st</sup> to 10<sup>th</sup> records still seen as one transaction unit. Of course, when **STOP\_ON\_ERROR** is specified, the 5<sup>th</sup> record to 10<sup>th</sup> record won't be committed at all only the 1<sup>st</sup> to 3<sup>rd</sup> records will be committed.

This option is valid only when auto-commit is off.

**[COLUMN\_DELIMITER=<delimiter>]** This specifies a character that separates each column in data file. The character should be single quoted. For example, to indicate that a **SPACE** is used as column delimiter, use ' '. Aside from normal characters, take the following escape sequences that represent special characters.

CHARACTER	ESCAPE SEQUENCE REPRESENTATION
TAB	\t
NEW LINE	\n

For example, if the delimiter is a TAB, users will use '\t' in *<delimiter>*. If the column delimiter is not specified, we will use TAB (\t) as the column delimiter. Use discretion when choosing a delimiter.

If the number of column delimiters is fewer than the number of target table columns specified by users, NULL will be used for the insert value.

**[ROW\_TERMINATOR=<row\_terminator>]** This is a string that denotes the end of a row. The *row\_terminator* should be double-quoted. The escape sequence rule for column delimiter applies to row terminator. In addition to that, the carriage-return also can be the escape sequence:

CHARACTER	ESCAPE SEQUENCE REPRESENTATION
CARRIAGE RETURN	\r

For example, if a carriage return and a new line character form a row terminator, the `<row_terminator>` should be `"\r\n"`. If no row terminator is specified, a new line character (`'\n'`) will be used as row terminator. The number of characters in row terminator should not be greater than 2.

Note that, no column delimiter should be in `row_terminator`.

`[QUOTATION=SINGLE_QUOTE | DOUBLE_QUOTE]` This indicates whether the alphabetic data in one field of a data source file is quoted. If `SINGLE_QUOTE` is specified, the data enclosed by single quotes is seen as one column of data. If `DOUBLE_QUOTE` is specified, the data enclosed by double quotes is seen as one column of data.

`[ESCAPE_CHAR=YES | NO]` This indicates whether an escape character (`\`) is used or not. The default is `YES`. If the escape character is used, the column delimiter character after escape character is seen as real data. For example, if we specify that a `TAB` be used as the column delimiter, and `ESCAPE_CHAR` is `YES`, a `\TAB` data is seen as `TAB` in data instead of column delimiter. For row terminator, this escape character means the line continues, and the `\n` is seen as real data. This rule also applies to the quotation mark.

`[LOB_FORMAT=INTERNAL | EXTERNAL]` If clob/blob format is internal, the text in the datafile is seen as the data that is going to be imported. Otherwise, the text is seen as a URL to external files that are going to be imported.

`server_column_name` This lists the names of the target table columns that are going to be imported from a data file. If there are spaces or equal signs in the table column name, use double quotes to enclose it.

`column_number` This is the cardinal number of each field in data file.

`server_column_name` and `column_number` are separated by space characters.

**NOTE** Note that if `server_column_name` and `column_number` are not specified, all columns in datafile will be imported into target table columns in the same order as datafile columns. That is to say, the 1st column in datafile will be imported as 1st column in the table, and the 2nd column in datafile will be imported as the 2nd column in table, etc. If the number of columns in datafile is greater than that of the target table, the remaining columns in datafile will be ignored. If, on the other hand, the number of columns in datafile is smaller than that of the target table, the remaining columns in target table will be inserted with `NULL`.

## DEFAULT VARIABLE FORMAT DESCRIPTION FILE

It's optional that users specify the description file for their data file format. If users do not specify the description file, a default description format is assumed. The default format means the following description file is used (On Win32 platform, the `ROW_DELIMITER="\r\n"`):

```
START_WITH_ROW=1
NUMBER_OF_ROWS_FOR_EACH_TRANSACTION=5
COLUMN_DELIMITER="\t"
ROW_TERMINATOR="\n"
```

➡ An example for importing a file with variable format description file is as follows:

A data file exists:

```
Davolio Nancy,Sales Representative,Ms.
Fuller Andrew,"Vice President, Sales",Dr.
Leverling Janet,Sales Representative,Ms.
Peacock Margaret,Sales Representative,Mrs.
```

```
Buchanan Steven,Sales Manager,Mr.
Suyama Michael,Sales Representative,Mr.
King Robert,Sales Representative,Mr.
```

The description file for this data file may look like this:

```
START_WITH_ROW=1
NUMBER_OF_ROWS_FOR_EACH_TRANSACTION=5
COLUMN_DELIMITER=","
ROW_TERMINATOR="\n"
DOUBLE_QUOTE
Name 1
Position 2
Gender 3
```

## IMPORT/EXPORT DATA RULES

The following table outlines the rules that must be applied when attempting to import or export data to or from a file.

DATA TYPE	IMPORT/EXPORT FORMAT	EXAMPLE
BINARY	Use HEX format	To import the binary number "0x004D2", use 004D2 in data file
CHAR	Characters are used exclusively	To import the word "inception", use inception in the data file
VARCHAR	See CHAR data type	
DATE	The format YYYY/MM/DD will be used for exporting	To import the date "2003/07/25", use 2003/07/25 in the data file
TIME	HH:MM:SS Export and import will use the format HH:MM:SS	To import the time "14:30:25", use 14:30:25 in the data file
TIMESTAMP	The combination of DATE format and TIME format forms the format of TIMESTAMP	To import the timestamp "2003/07/25 14:30:25", use 2003/07/25 14:30:25 in data file
DECIMAL	Use numeric data representation	To import the number "36.82", use 36.82 in data file
DOUBLE	Use numeric data as described	13e+12 To import

DATA TYPE	IMPORT/EXPORT FORMAT	EXAMPLE
	in DECIMAL or scientific notation of numbers	the number "13e+12", use 13e+12 in data file
FLOAT	See DOUBLE	
INTEGER	Use integer data	To import the integer "576", use 576 in data file
LONG VARBINARY	<p>Two formats can be used: embedded or external file format.</p> <p>For embedded format, HEX characters are used.</p> <p>For external file format, the URL is provided.</p> <p>Use description flag LOB_FORMAT to indicate your option. For details see description file specifications.</p>	<p>(1) embedded format: The format used will be the same as BINARY.</p> <p>(2) external file format: For example, if users want to import a binary file whose full path is "c:\My Document\GRAPH.GIF". The URL provided will be c:\My Document\GRAPH.GIF</p>



## 9. XTT/XTM

DBMaster includes two Java-based, platform-independent tools for passing data between a database and XML documents. The XML Transfer Template tool and the XML Transfer Mapping tool allow you to create custom templates that determine how data maps from a database to XML files.

The XTT is designed for changing data template through xml format. The XTM maps the relationship between XTT and table. Actually is equal to through xml format importing/exporting data, but the difference is that users can use XTT/XTM establishing xml data format. Thus we can see the Tools as a way to backup/restore, the follow sections describe the tools and introduce how to create or edit XTT/XTM file etc. If you want to know more information about how to use XTT /XTM Tools, please consult '*XTT/XTM Tool user's Guide*'.

## 9.1 XML Transfer Template Tool

XML Transfer Template Tool is XTT for short. The purpose of the XML Transfer Template (XTT) tool is to provide a customizable bridge between database data and XML documents. The bridge takes the form of a template file, the XML Transfer Template (XTT). The XTT file determines which database tables and columns to map to which XML elements and attributes. You determine the mapping using drag-and-drop operations in the XTT tool. The XTT tool ensures that XTT syntax is correct, and also aids in performing tasks such as generating schema documents (XSD) or document type definitions (DTD).

Using The XTT tool to transform data is a four-phase process – creating or importing the XTT structure, linking XTT objects to the database with SQL queries, generating DTD or XSD files if necessary, and finally generating the XML document.

Usually, linking XTT objects will only be necessary if you are importing an existing XML structure from an XML file, XSD, or DTD. Likewise, linking will not be necessary if you are creating an XTT based on the database. Hybrid situations may exist, however; for example, where you have an existing XML structure but need to add new elements for new data.

XTT files define a map from database tables and columns to the elements and attributes of an XML file. An XTT file is a document with syntax similar to a valid XML document. Elements define tables and columns, and attributes define SQL queries, attribute names and values, and element values for XML documents generated using the XTT.

So if you want to backup a database or a table, first you must know the tool and then you can create an xtt file or import an excised file.

➡ Example:

The following is a complete XTT file that maps data from the table CARD. It maps the columns FIRSTNAME, LASTNAME, and TITLE as attributes of the element CARD, and the column NUM as a child element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
  <root>
    <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME, LASTNAME,
TITLE, BMP from SYSADM.CARD">
      <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME" />
      <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME" />
      <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE" />
      <NUM xtt:textvalue="$CARD_SQL0.NUM" />
    </CARD>
  </root>
</xtt:template>
```

If the XTT in the above example is run, the following XML file is generated:

```
<?xml version="1.0" encoding="US-ASCII" ?>
<root>
  <CARD FIRSTNAME="Eddie" LASTNAME="Brown" TITLE="Manager">
    <NUM>1</NUM>
  </CARD>
</root>
```

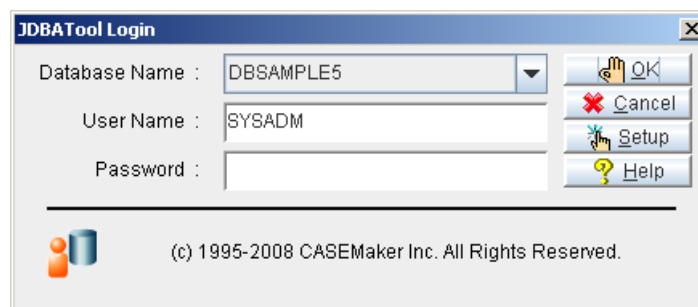
The XTT tool provides a simple user interface for scripting, validating and running XTT files. The following sections describe the user interface, and give procedures to help you quickly learn to start creating your own XTT files.

### 9.1.1 GETTING TO KNOW THE XTT TOOL

This section describes the elements of the XTT Tool user interface and how to log onto the database.

#### Opening the XTT tool and logging into a database

When you open the XTT tool from the Windows start menu you will automatically be prompted to log into the database. Select the database from which you want to export information. You must have an account on the database in order to log in. Be sure to use an account that has access to all the tables that you will need information from.



*Figure 8-1: The login dialog*

- ➔ To open the XTT tool and log into a database:
1. From the Windows Start menu, click **Start > Programs > DBMaster 5.1 > XML Transfer Template**.
  2. In the **Login** dialog, select a database and enter a user name and password
  3. Click **Ok**. The XTT tool will display database tables in the Database Schema Panel.

#### The Main Console

The Main Console can be divided into five logical areas. Refer to figure below.

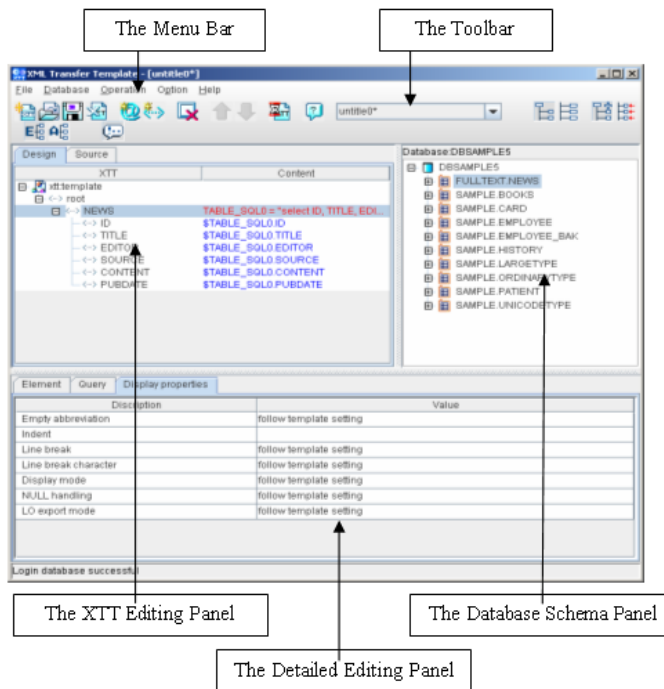


Figure 8-2: Elements of the Main Console

## The Menu Bar

The Menu Bar consists of five menus: **File**, **Database**, **Operation**, **Options**, and **Help**. Menu items are disabled if they cannot be used. Refer to the following sections for each menu item's function.

### File

The **File** menu consists of the following items:

- **New XTT > Empty XTT**: Creates a new empty XTT. Refer to *Creating an empty XTT file* for more information
- **New XTT > With imported DTD**: Creates a new XTT based on a DTD file. Refer to *Creating an XTT from a DTD file* for more information
- **New XTT > With imported XSD**: Creates a new XTT based on an XSD file. Refer to *Creating an XTT from an XSD file* for more information
- **New XTT > With imported XML**: Creates a new XTT based on an XML file. Refer to *Creating an XTT from an XML file* for more information
- **Open XTT**: opens the **Open** dialog with **.XTT** as the default file extension filter.
- **Close**: closes the XTT currently open in the XTT editing panel. If the XTT has been modified a confirmation dialog will ask to save changes.
- **Save**: saves the XTT currently open in the XTT editing panel. If the XTT has not been saved before, the **Save as** dialog will open.
- **Save as**: Opens the **Save as** dialog with **.XTT** as the default file extension.
- **Generate DTD**: Opens the **Save as** dialog with **.DTD** as the default file extension. Refer to *Generating a DTD* for more information.
- **Generate XSD**: Opens the **Save as** dialog with **.XSD** as the default file extension. Refer to *Generating an XSD* for more information
- **Recent files**: displays the most recently opened files

- **Exit:** exits the XML Transfer Template tool

## Database

---

The **Database** menu consists of the following items:

- **Connect:** Opens the **Login** dialog. A list of running databases appears in the drop down menu.
- **Disconnect:** stops the session with the database. The content in the **database schema panel** is cleared.
- **Refresh:** refreshes the **database schema panel** if a session is active.

## Operation

---

The **Operation** menu consists of the following items:

- **Insert > Element:** inserts a new empty element into the XTT object tree. Refer to *Adding New Elements and Attributes* for more information
- **Insert > Attribute:** inserts a new empty attribute into the XTT object tree. Refer to *Adding New Elements and Attributes* for more information
- **Undo:** returns the XTT object tree to the state it was in before the last modification
- **Copy:** copies the selected node of the XTT object tree and all descendants
- **Cut:** cuts the selected node of the XTT object tree and all descendants
- **Paste:** Pastes the last cut or copied node of the XTT object tree and all descendants.
- **Remove:** removes the selected node of the XTT object tree and all descendants
- **Run:** executes the XTT file. Refer to *Generating XML data* for more information.
- **Validate:** checks if variable references for elements exist in the parent element, and checks if SQL commands in elements are valid in the database

## Options

---

The **Options** menu consists of the following items:





- **Preferences:** opens the user preferences dialog
- **Tree operation options:** opens the tree operation options dialog

## The Toolbar

This section shows the toolbar items with their equivalent menu bar operations.



### File Operations




---

-  New empty XTT= menubar > File > New XTT > Empty XTT
-  Open XTT= menubar > File > Open XTT
-  Save= menubar > File > Save
-  Close= menubar > File > Close

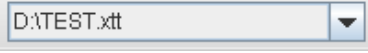
### XTT Tree Operations

---

-  Add Attribute Object = menubar > Operation > Insert > Attribute
-  Add Element Object = menubar > Operation > Insert > Element










-  Remove Tree Node = menubar > Operation . Remove
-  Move Up = Move current selected node before its previous sibling node. The element can be moved before another element but cannot be moved before an attribute.
-  Move Down = Move current selected node after its next sibling node. The attribute can only be moved after another attribute but can't be moved after an element.

## Opened Files

- 

The combo box displays all open XTT filenames. If the file has been edited, then there will be an asterisk (\*) after the filename. When you choose a different filename, the XTT edit panel will reload and show the XTT tree of the newly selected file.

## Operation Options

-  Run Transfer = menubar > Operations > Run
-  Help = menubar > Help > Help
-  Insert as Child = Insert elements as child elements (when performing drag-and-drop operations)
-  Inset as Sibling = Insert elements as sibling elements (when performing drag-and-drop operations)
-  = Add a new node (when performing drag-and-drop operations)
-  = Link by source structure (when performing drag-and-drop operations)
-  = Add a new element below the selected element (follows **Insert as ...** rule)
-  = Add a new attribute within the selected element
-  = Operation options – when selected, causes **Customize** dialog to appear when performing drag-and-drop operations on tables or views.

## The XTT Editing Panel

The XTT editing panel consists of two views: the design view and the source view. The design view contains the XTT object tree and is displayed by default when an XTT is created or opened. The XTT object tree is a logical representation of the XTT file itself, which is a well-formed XML document. The XTT object tree consists of five types of elements, described in Table .

XTT object type	Tree node name	Content description
<xtt:template>	xtt:template	(none)
<root>	root	(none)
<xtt:attribute>	value of attribute 'name'	value of attribute 'value'
user-defined element without query or text value	element's tag name	(none)

XTT object type	Tree node name	Content description
user-defined element with query	element's tag name	value of attribute 'xtt:query' + '=' + value of attribute 'xtt:command'
user-defined element with text value	element's tag name	value of attribute 'textvalue'

Table 8-3: Element types in the XTT object tree.

The five XTT element types as they appear in the design view are illustrated in Table .

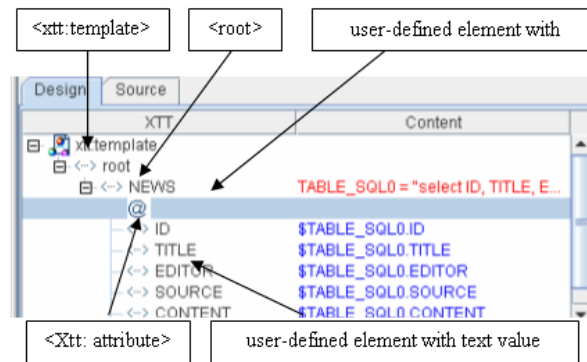


Table 8-4: Design view of element types in the XTT object tree

To select an XTT tree object, left-click on it and it will be highlighted in blue. If you right click on the design view, a pop-up menu will appear depending on the object that is highlighted. Right-clicking on an object will not cause it to be selected. Table summarizes the pop-up menu contents for different selected elements.

Node type	Pop-up menu
<xtt:template>	N/A
attribute	<ul style="list-style-type: none"> <li>● 'Change to ELEMENT' - the selected attribute node will be replaced by an element with the same name. The data in the <b>value</b> attribute will be copied to the <b>textvalue</b> attribute of the new element</li> <li>● Cut – same as ctrl-X</li> <li>● Copy – same as ctrl-C</li> <li>● 'Remove' - removes the current selected node from the tree</li> </ul>

user-defined element	<ul style="list-style-type: none"> <li>● Element – creates a new element below the selected element. Whether the new element is inserted as a sibling or child element depends on whether the <b>Insert as Child</b> or <b>Insert as Sibling</b> option has been selected</li> <li>● Attribute – creates a new attribute for the selected element</li> <li>● 'Change to ATTRIBUTE' – only provided if the object has no sub node (either attribute or element). The selected element will be replaced by an attribute with the same name. The data in <b>text value</b> will be copied to the <b>value</b> attribute of the new attribute</li> <li>● Cut – same as ctrl-X</li> <li>● Copy – same as ctrl-C</li> <li>● Paste – same as ctrl-V</li> <li>● 'Remove' - to remove the current selected node from the tree</li> </ul>
----------------------	---

*Table8-5: pop-up menus available in the XTT editing panel design view*

The **Source** tab displays the source code for the XTT file.

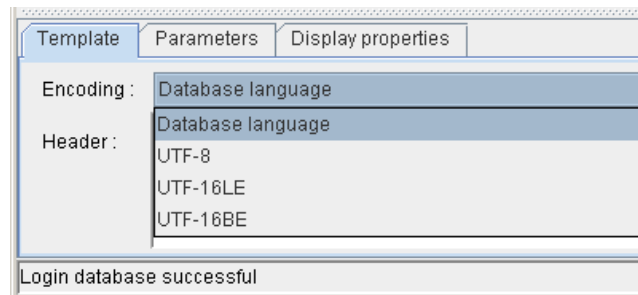
### The Database Schema Panel

The database schema panel shows the table/views of the connected database as a schema tree. By expanding the table/view node, it will show each column as its sub node. The nodes in the schema tree can be dragged into the XTT tree panel. It will then add a new node or link the schema information into XTT tree depending on he settings selected

### The Detailed Editing Panel

Detail editing panel shows the detailed properties of the selected node in the XTT tree. The general rule for text fields in the editing panel is that when an object outside the text field or area is selected the value is set. For example, the name field will be set after the next field is selected or the XTT tree selection is changed. The change can be seen on the XTT tree.

#### ***xtt:template***



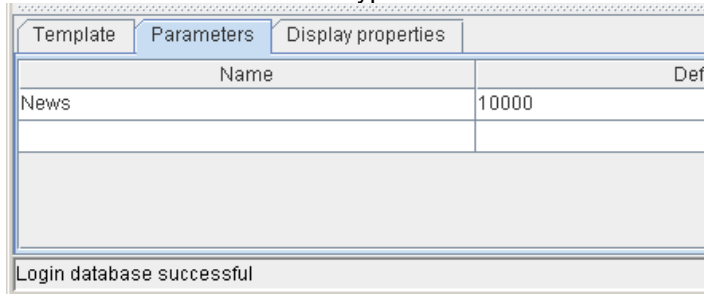
*Figure8-6: The language encoding menu of the detailed editing panel*

**Encoding** – The **Encoding** menu specifies the text encoding for any output XML file. The choices are database local (the text encoding specified in the database), UTF-8, UTF-16LE and UTF-16BE.

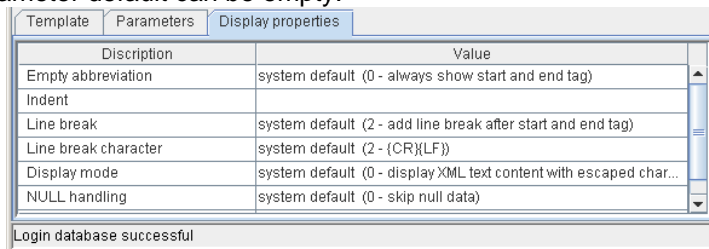


The default is database local. If database local is Big5 and the encoding setting is database local, then the output XML will be encoded in Big5.

**Header** – The header box it is where you add information like a schema file or applicable XSL in the output XML file. The XTT engine will print the content in this block to the output XML file after `<?xml version="1.0"...?>` . Be sure to type valid XML content in this header block.



**Parameters** – you may add as many parameters as you want. The parameter name must be unique. Press the ‘delete’ key to remove a selected row. The last empty row cannot be removed. The parameter default can be empty.



There are a few display settings available for the output XML file. The default value for each setting is ‘system default’. The display settings will apply to all XTT objects unless specified otherwise.

**Empty abbreviation** –

0 – always show start and end tags for an element, even if its content is empty.

1 – to use the abbreviated form of the end tag if no sub element is produced. For example, **<Department id="1001" />**.

2 – hide the start and end tags if the element content is empty; no text, no attribute, no child element. If the element only has attributes, it will use type 1 abbreviation.

**Indent** – the number of indent spaces in the source document for each sub level as displayed in a text editor. For example, if the number is 2, then the start tag of the root will be indented 2 spaces, and a sub node of the root will be indented 4 spaces.

**Line break** –

0 – do not add any line break.

1 – add a line break after an end tag.

2 – add a line break after every start and end tag.

**Line break char** – the character(s) to be added as line break.

0 – {CR}

1 – {LF}

2 – {CR} {LF}.

**Display mode** – the way to display text data. Generally, the text character ‘<’ will be replaced with ‘&lt;’ in the text content. But ‘<’ can be used if it is enclosed in the CDATA section, or the text

content itself is already an XML fragment and can be added into the output XML as a part of the XML content. There are 3 different ways to display text data.

0 – Display XML text content with the escape character.

1 – Use a CDATA section to enclose the text value.

2 – Make the content native XML text.

**Null handling** – specify how to handle null data.

0 – skip null data. If it's an attribute, then do nothing.

1 – Show empty content. For example, `<NAME></NAME>`.

2 – Display 'NULL'. For example, `<NAME>NULL</NAME>`.

**LO mode** – specify how to handle large objects.

0 – dump large object data directly into the XML file. Print a string if it is CLOB type data or print in hexadecimal format if it is BLOB type data.

1 – To store large object data in external file.

### User-defined element

The following represents the properties of a user-defined element in the XTT object tree.

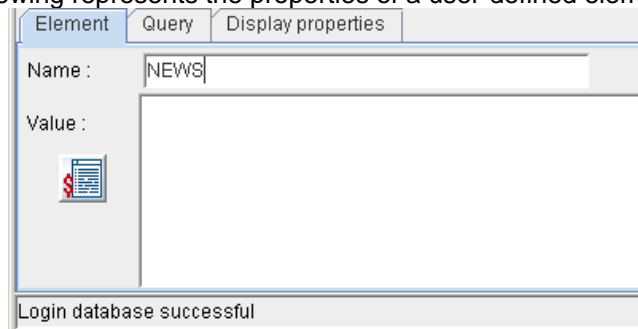


Figure 8-7: The detailed editing panel for a user-defined element

**Name** – the element name. It is case-sensitive and cannot be empty.

**Value** – the text value. This is a text expression, which can have both constant text and a variable reference. For example, you might want to add the country code to all phone number data, such as "886 - \$SQL1.PHONE", where "886 –" is the constant text and "\$SQL1.PHONE" is the variable reference. The XTT engine will concatenate both into the output XML file as the text value of the element.

**Browse button** – the browse button will list all the available parameters and variable references at the current level. You can choose the variable name(s) and insert them into the text expression field next to the browse button.

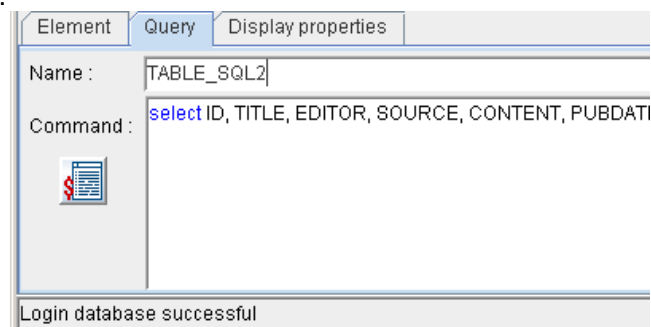


Figure 8-8: The query view for a user-defined element in the detailed editing panel

An element can have embedded query properties in it. A valid XTT element will have both fields (name and command) specified or both left as empty.

Name – the query name. It will be used as a variable reference in the sub nodes.

Command – SQL query statement. The statement must be able to generate a result set. For example, you cannot type a **delete table** statement here.

The display settings are similar to the ones in xtt:template, But the default is 'follow template setting'.

Null handling – there are only two choices instead of three. If 'follow template setting' is selected and the template has the null handling setting as '0 - skip null data', then it will be treated as '1 show empty content'.

1 – Show empty content. For example, <NAME></NAME>.

2 – Display 'NULL'. For example, <NAME>NULL</NAME>.

## Attribute Node

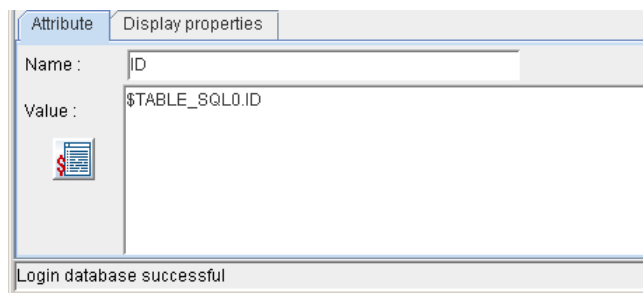


Figure 8-9: An attribute viewed in the detailed editing panel

Name – attribute name. The Name field cannot be empty. Multiple attributes must have names unique to the same parent element.

Value – the attribute value; a text expression field.

There are only two display settings for an attribute node; null handling and LO mode. The default for both is 'follow template setting'.

## The Customize Dialog

The customize dialog appears when performing drag-and-drop operations if you have selected **show customize dialog**. The appearance of the customize dialog depends on the settings selected in the tree operation options dialog. There are two main views for the customize dialog depending on whether you have selected **Link by source structure** or **Add a new node**.

## The User Preferences Dialog

The user preferences dialog is where you select the user interface language, and the method by which you wish to view the results. You may select English, Chinese, or Japanese as the user interface language. You may also choose to select your default XML browser or your default text editor to view the output when running the XTT.

## The Tree Operation Options Dialog

The tree operation options dialog is where you select the behavior of drag-and-drop operations. You may select to add database objects as elements or attributes; add objects as child nodes or sibling nodes; and to add objects as new trees, or to link data to existing elements or attributes.

## 9.1.2 CREATING A NEW XTT

---

An XTT is the map by which XML data files are produced. An XTT may be created in one of four ways: from an empty XTT file, from a DTD file, From an XSD file, and from an XML file.

### Creating an empty XTT file

Creating an empty XTT file is useful if you have data in a database that you want to display in XML form, but have no preconditions for how the XML data must be formatted. An empty XTT file consists only of the root node. After creating an empty XTT file, you may add elements and attributes. To learn about adding elements and attributes refer to *Editing an XTT*.



To create an empty XTT file:

1. Open the XTT tool and log in to the database that you want to use.
2. Click **File > New XTT > Empty XTT**. The root node will appear in the XTT object tree.

### Creating an XTT from a DTD file

You may want to define the structure of your XML documents based on an existing Document Type Definition (DTD) file. If you have an existing XML document based on an external DTD and want to produce XML files from database data that conforms to that DTD, then you can create an XTT from a DTD file.

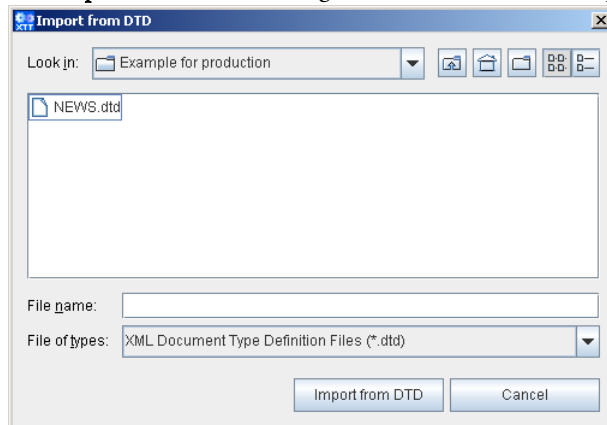
A root element must be specified for an XTT. During the import process a dialog allows you to select any valid element definition as the root for the XTT.

After creating the XTT, all the elements and attributes under the selected root node of the DTD will appear in the XTT object tree, however, none of the element or attribute definitions contain values – there is no method to pass SQL data to an XML file using the XTT. This may be accomplished by editing element nodes in the XTT object tree. For details on how to modify nodes in the XTT object tree, refer to Mapping Data to Elements.

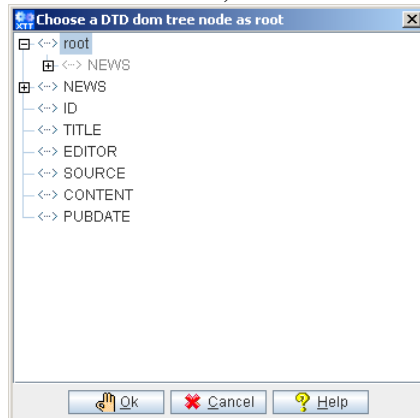


To create an XTT from a DTD:

1. Open the XTT tool and log in to the database that you want to use.
2. Click **File > New XTT > With imported DTD**.
3. In the **Import from DTD** dialog, select the DTD file to import and click **Import from DTD**.



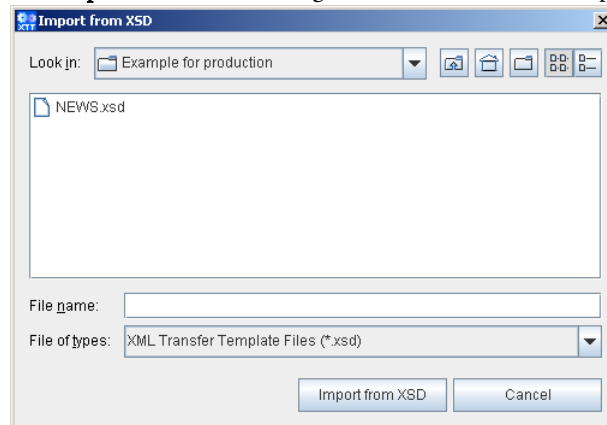
4. In the **Choose a DTD dom tree node as root** dialog, select an element definition to be the root element in the XTT object tree and click **Ok**.



## Creating an XTT from an XSD file

It is also possible to base the XTT object tree structure on an XML schema defined by an XML Schema Definition (XSD) document file. As with DTD files, a root element must be specified, you may select any valid element definition as the root for the XTT. Also, none of the newly created attribute or element definitions in the XTT object tree has a value; you must modify the element and attribute definitions in order to get SQL data into an XML file. For details on how to modify nodes in the XTT object tree, refer to *Editing an XTT*.

- To create an XTT from an XSD file:
  1. Open the XTT tool and log in to the database that you want to use.
  2. Click **File > New XTT > With imported XSD**.
  3. In the **Import from XSD** dialog, select the DTD file to import and click **Import from XSD**.



4. In the **Choose an XSD Dom tree node as root** dialog, select an element definition to be the root element in the XTT object tree and click **Ok**.

## Creating an XTT from an XML file

If you do not have a DTD or XSD file to base your XTT structure on, but need to maintain consistency with an existing XML structure, then you may create an XTT directly from an XML file. The XTT tool will parse the structure of your XML document to generate an XTT object tree. The primary difference between this method of creating an XTT versus using a DTD or XSD is that you are not given an option as to which element will constitute the root node of the XTT object tree.

- To create an XTT from an XML file:
  1. Open the XTT tool and log in to the database that you want to use.
  2. Click **File > New XTT > With imported XML**.
  3. In the **Import from XML** dialog, select the DTD file to import and click **Import from XML**.

### 9.1.3 EDITING AN XTT

After you have created the root node and structure (if creating an XTT from XML, DTD, or XSD), you will want to provide content for the generated XML file. For blank XTT, this is primarily a drag-and-drop operation from the tables in the database schema panel into the XTT object tree. The sections *Inserting a Table* and *Adding New Elements and Attributes*, describe the primary tasks you will need to accomplish if creating a new XTT.

For XTT based on an XML, DTD, or XSD file, you will need to add query statements and values to the attribute and element definitions in the XTT object tree. These tasks are described the section *Mapping Data to Elements and Attributes*.

These tasks are not mutually exclusive and the above guidelines are only provided to enable you to quickly understand how to create a valid XTT document. At times you may find it useful to modify an element definition in a new XTT – one example being if you only wish to select values from an SQL table that meet some conditions. Or you may not need to conform precisely to the XML schema that your XTT is based on – in which case it is possible to use drag-and-drop operations to build your XTT object tree.

## About the Design View

The design view of the XTT editing panel displays the XTT object tree. For a new, blank XTT, the XTT object tree contains only the XTT template node and an empty root node. An XTT object tree

that has been created from an XML, XSD, or DTD will have a different root. Refer to *The XTT Editing Panel* for detailed information on the objects and functions available in the design view.

## Inserting a Table

You can insert a table as a child or sibling. The first table you insert must be a child of the root node; attempting to add an element to the xtt:template node will return an error.

Before a table is added to the XTT object tree, you can choose which columns are to be added as elements, which columns to add as attributes, which columns to select but not add to the XTT object tree, and which columns not to select. This is accomplished in the customize dialog. Click the **show customize dialog** button if you want the customize dialog to appear when adding object to the XTT object tree.

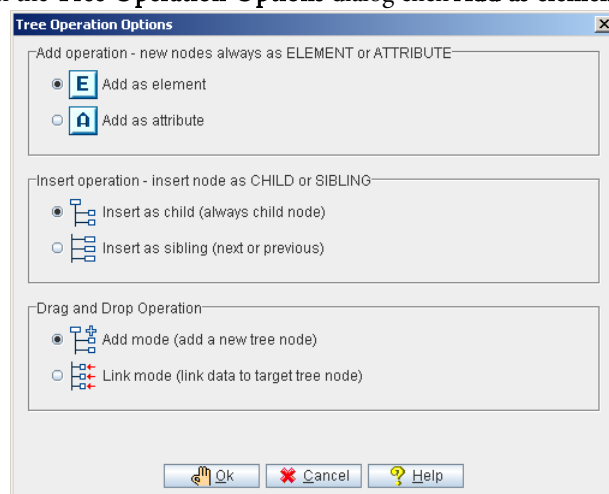
When inserting a table, the customize dialog will display a query object name, and the structure of the database object as it will appear after it is entered into the XTT object tree. By default both parent and child objects are inserted as elements. You may also choose to insert child objects as attributes, to select database objects without inserting them as attribute objects or element objects, or to not select the database objects.

After you have inserted the first table, subsequent tables may be added as children or siblings of the first table. Tables must always be represented as elements in the XTT object tree. If you have created in the XTT an existing file, any tables you insert can be children or siblings of any of the existing XTT elements.



To insert a table as a child node:

1. Click **Option > Tree Operation Options**.
2. In the **Tree Operation Options** dialog click **Add as element**, **Insert as a child**, and **Add mode**.



3. Select a table from the database schema panel
4. Drag the table from the database schema panel to the XTT object tree node that is to be the parent of the new node.
5. In the customize dialog, select any columns that you do not want to include and click **remove**. Select any columns that you wish to select in the SQL command, but do not want to export to the XML file and click **hidden**. Select any columns that you wish to add as attributes and click **attribute**. Select any columns that you wish to add as elements and click **element**.
6. Click **OK**. The table will appear as a new element in the XTT object tree. Columns will appear as elements and attributes, depending on how they were selected in the customize dialog.

## Adding New Elements and Attributes

You may need to add elements or attributes to the XTT object tree in order to create the desired data structure. Note that the following procedure only describes how to add empty elements and

attributes. Refer to defining elements and defining attributes for information on how to map data from the database to empty elements in the XTT object tree.

- ➔ To add new elements or attributes:
  1. Select the element to which you will add a new element or attribute.
  2. Determine the relationship between the selected element and your new object: click **insert as sibling** to make the new object a sibling of the selected element, click **insert as child** to make the new object a child of the selected element.
  3. Click **add attribute object** to add an attribute. Click **add element object** to add a new element.
  4. Type a name for the new object in the name box of the **detailed editing panel** and press Enter.

## Mapping Data to Elements and Attributes

Elements should be associated with data. To associate an element object with data, the parent element object must contain an SQL query. The SQL query must select the table and column that will map to the child element object. After associating the parent element with an SQL query, you must associate a child element with the column.

Use the link by source structure tree operation option to associate elements with SQL queries. When you associate an element with an SQL query, use the customize dialog to associate child elements or attributes with the selected columns of the SQL query. Dragging a table from the database schema panel into a parent element in the XTT object tree will cause the customize dialog to open.

When linking element objects by source structure, the customize dialog will show the query object name, and two columns: the XTT object column, and the mapping column. The XTT object column displays all child elements and attributes of the element that you have dragged the table into. The mapping column displays any existing content, and is where you select the column content that you want to map from. Click a row in the mapping column to select an SQL data source for the corresponding XTT object.

- ➔ To map data to an empty XTT element object:
  1. Click a table in the database schema panel.
  2. Drag the table to an element object in the XTT object tree. The element you dragged the table into will be the parent element.
  3. In the **Customize dialog**, select the mapping that you wish to perform. For each element and attribute object:
  4. Click the Mapping column that corresponds to the XTT object that you wish to map database data to.
  5. Select the value for the XTT object from the drop-down menu. The value is the equivalent of the column data in the database. It appears in the format of the name of the SQL query followed by a dot, and then the column name.
  6. Click **OK** when you have completed selecting mapping values for all the XTT object nodes that you wish to map to database columns.

## Saving an XTT

After you have completed editing the XTT object tree, you should save the XTT file. The saved XTT file can be later recalled for modification, or called in a stored procedure to automatically pass database data to XML files.

To save an XTT click the **Save** icon, or click **File > Save** from the menu bar.



## 9.1.4 GENERATING A DTD

If your development requirements determine that you will need to create a Document Type Definition (DTD) file to describe the structure of generated XML files, then you can use the **Generate DTD** function in the XML Transfer Template tool.

The DTD file structure will be consistent with the structure of the source XTT

➡ Example:

Given the following XTT file:

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
  <root>
    <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME, LASTNAME,
TITLE, BMP from SYSADM.CARD">
      <NUM xtt:textvalue="$CARD_SQL0.NUM" />
      <FIRSTNAME xtt:textvalue="$CARD_SQL0.FIRSTNAME" />
      <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
      <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
      <BMP xtt:textvalue="$CARD_SQL0.BMP" />
    </CARD>
  </root>
</xtt:template>
```

The resultant DTD file will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT root (CARD*) >
<!ELEMENT CARD (NUM, FIRSTNAME, LASTNAME, TITLE, BMP) >
<!ELEMENT NUM (#PCDATA) >
<!ELEMENT FIRSTNAME (#PCDATA) >
<!ELEMENT LASTNAME (#PCDATA) >
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT BMP (#PCDATA) >
```

➡ To generate a DTD file from an XTT object tree:

1. Ensure that the XTT that you want to convert to a DTD is open.
2. Click **File > Generate DTD**.
3. In the **Generate DTD** dialog, select an output path and specify a file name and encoding type. Possible encoding types include: **UTF-8**, **UTF-6LE**, **UTF-16BE**, and Local Code.
4. Click **Generate DTD**, a DTD file will be created in the selected folder.

## 9.1.5 GENERATING AN XSD

You may wish to generate an XML schema file from the logical structure represented in the XTT file. Some tools may require a schema file to be able to parse XML data. The schema file structure is consistent with the XTT object tree structure. In the example below, a small XTT file is used to generate a schema file.

➡ Example

Given an XTT file with the following object tree structure:

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
  <root>
```

```
<CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME, LASTNAME,
TITLE, BMP from SYSADM.CARD">
  <xtt:attribute name="NUM" value="$CARD_SQL0.NUM" />
  <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME" />
  <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME" />
  <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE" />
  <BMP xtt:textvalue="$CARD_SQL0.BMP" />
</CARD>
</root>
</xtt:template>
```

The resultant schema file structure will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="root">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="CARD" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="CARD">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="BMP" />
    </xsd:sequence>
    <xsd:attribute name="NUM" type="xsd:int" />
    <xsd:attribute name="FIRSTNAME" type="xsd:string" />
    <xsd:attribute name="LASTNAME" type="xsd:string" />
    <xsd:attribute name="TITLE" type="xsd:string" />
  </xsd:complexType>
</xsd:element>
<xsd:element name="BMP" type="xsd:string" />
</xsd:schema>
```

➔ To generate an XSD file from an XTT object tree:

1. Ensure that the XTT that you want to convert to an XSD is open.
2. Click **File > Generate XSD**.
3. In the **Generate XSD** dialog, select an output path and specify a file name and encoding type. Possible encoding types include: **UTF-8**, **UTF-6LE**, **UTF-16BE**, and Local Code.
4. Click **Generate XSD**, an XSD file will be created in the selected folder.

## 9.1.6 GENERATING XML DATA

After creating the required transfer template and optional DTD or XSD file, you are ready to generate an XML file using the data stored in the database.

The **Run transfer** function is most useful for testing the XTT. After you have created an XTT, you can use it to generate XML documents on demand and pass data to your XML application.

The following example shows a completed XTT template file and corresponding output file.

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaster-com:xml-template">
  <root>
```

```

<CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME, LASTNAME,
TITLE, BMP from SYSADM.CARD">
  <NUM xtt:textvalue="$CARD_SQL0.NUM" />
  <FIRSTNAME xtt:textvalue="$CARD_SQL0.FIRSTNAME" />
  <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
  <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
  <BMP xtt:textvalue="$CARD_SQL0.BMP" />
</CARD>
</root>
</xtt:template>

```

The corresponding output file for the preceding XTT file:

```

<?xml version="1.0" encoding="US-ASCII" ?>
- <root>
- <CARD>
  <NUM>1</NUM>
  <FIRSTNAME>Eddie</FIRSTNAME>
  <LASTNAME>Chang</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile0.tmp</BMP>
</CARD>
- <CARD>
  <NUM>2</NUM>
  <FIRSTNAME>Hook</FIRSTNAME>
  <LASTNAME>Hu</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile1.tmp</BMP>
</CARD>
</CARD>
- <CARD>
  <NUM>7</NUM>
  <FIRSTNAME>Oscar</FIRSTNAME>
  <LASTNAME>Tseng</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile6.tmp</BMP>
</CARD>
- <CARD>
  <NUM>8</NUM>
  <FIRSTNAME>Jerry</FIRSTNAME>
  <LASTNAME>Liu</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile7.tmp</BMP>
</CARD>
</root>

```

## 9.2 XML Transfer Mapping Tool

The XML transfer mapping (XTM) tool allows you to pass XML data to a database using XSL transformations. The XML Transfer mapping tool consists of three parts: an XML schema part, which displays the schema of the XML file(s) that you are using as source data; an SQL database part, which displays the database tables; and an XTM part, which displays the mapping from the XML schema to the database tables.

Using the tool can be summarized in five basic steps: opening a source XML or XSD file to create a source XML schema, connecting to a database, creating an XTM structure from the database table, mapping elements and attributes from the source XML schema, and finally storing the XTM structure as an XSL file.

Once the XSL file is created, it can be used to transform any XML file that conforms to the source schema to database data.

### 9.2.1 GETTING TO KNOW THE XTM TOOL

Unlike the XTT tool, the XTM tool does not require a connection to the database when it is opened. The tool creates a database connection when you create or open an XTM, and is used to display the database schema tree, described in *XML Schema Tree*.

This chapter describes all of the major screen elements in the XTM tool.

#### The Main Console

The main console of the XTM tool can be broken down into five major areas as illustrated in Figure .

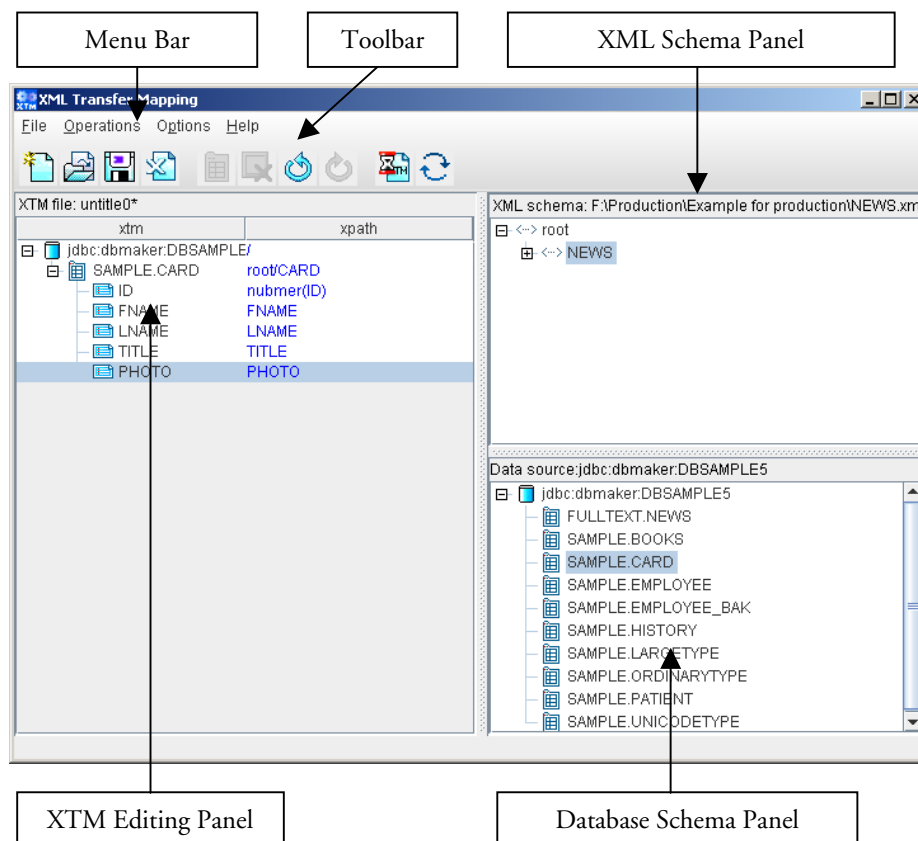


Figure 8-10: Elements of the XTM main console

## The Menu Bar

The menu bar consists of four menus: **File**, **Operations**, **Options**, and **Help**. Menu items are disabled if they cannot be used. Refer to the following sections for each menu item's function.

The **File** menu consists of the following items:

- **New:** opens the **New XTM** dialog, which prompts you to enter source schema and database information
- **Open:** opens the **Open** dialog with .XSL as the default file extension filter.
- **Save:** saves the XTM currently open in the XTM editing panel as an XSL file. If the XTM has not been saved before, the **Save as** dialog will open.
- **Save as:** Opens the **Save as** dialog with .XSL as the default file extension.
- **Close:** closes the XTM currently open in the XTM editing panel. If the XTM has been modified a confirmation dialog will ask to save changes
- **Recent files:** Lists the XSL files opened during the current session
- **Exit:** Exits the XTM tool. Opens the **Save as** dialog with .XSL as the default file extension if the current XTM has not been saved

## Operation

The **Operation** menu consists of the following items:







- **Undo:** returns the XTT object tree to the state it was in before the last modification
- **Redo:** Executes the last performed action again
- **Insert:** Inserts a new XTM control node. Opens the **New XTM Control Node** dialog
- **Remove:** Removes the selected node of the XTM object tree and all descendants
- **Run:** Opens the **Execute XTM** dialog. The XTM dialog offers the option to immediately execute the XTM and send data to the database or to generate an SQL script for later use
- **Refresh:** Queries the database to refresh database schema





## Options

The **Options** menu consists of the following items:

- **Preferences:** opens the **Preferences** dialog, which allows you to choose the language that the UI is displayed in, and to specify different syntax for the connect section of the XSL file
- **JDBC Drivers:** opens the **JDBC Drivers** dialog, which allows you to connect to other data sources

## 9.2.2 THE TOOLBAR

-  New File: Menu bar > File > New
-  Open File: Menu bar > File > Open
-  Save File: Menu bar > File > Save
-  Close File: Menu bar > File > Close
-  Add new Table/Statement Node: Menu bar > Operations > Insert
-  Delete Node: Menu bar > Operations > Remove

-  Undo: Menu bar > Operations > Undo {command}
-  Redo: Menu bar > Operations > Redo
-  Run: Menu bar > Operations > run
-  Refresh Database: Menu bar > Operations > Refresh

### 9.2.3 XTM OBJECT TREE

---

The XTM Object Tree is a logical representation of the structure of the XTM file. It contains two columns: the XTM column and the xpath column. The XTM column contains a graphical tree representation of objects that have been inserted from the Database Schema Tree. The xpath column contains a path to address the corresponding location in the XML schema tree.

### 9.2.4 XML SCHEMA TREE

---

The XML Schema Tree provides a graphical representation of the schema of the XML, DTD, or XSD file from which the XTM will create addresses. Any XML file that conforms to the schema represented in the XML Schema Tree can be used as a data source after the XTM has been created and saves as an XSL.

### 9.2.5 DATABASE SCHEMA TREE

---

The Database Schema Tree displays a logical representation of database tables within the selected database.

### 9.2.6 CREATING AN XTM

---

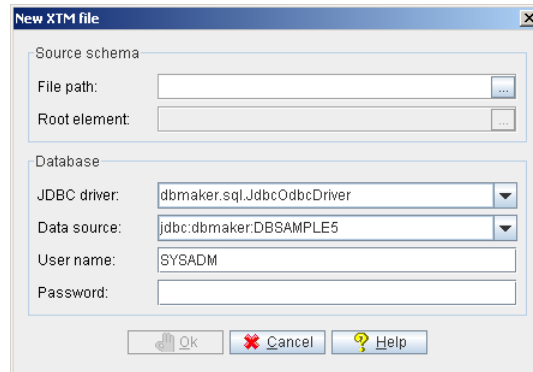
Creating an XTM requires selecting a data source and connecting to a database.

The data source must be an XML, XSD, or DTD file. You must choose one of the elements in the source file to be the root. Only child elements and attributes of the selected element will be visible in the XML schema panel after you create the XTM file. If you want to include other elements or attributes, you will need to create a new XTM file.

Connecting to a database requires a driver and a connection to the database. The standard driver for DBMaster databases is **dbmaster.sql.JdbcOdbcDriver**. For more information about drivers refer to *Adding a New JDBC Driver*. The connection to the database requires that the database is started, that a channel of communications over TCP/IP is open, and that a username and password for an account with privileges on the tables you wish to display is available.

➔ To create a new XTM:

1. Select **File > New** from the menu bar. The **New XTM file** dialog opens.



2. In the **File path** box, type a file path or select the browse button to locate a source schema file (XML, XSD or DTD).
3. To select a root element different from the one indicated in the Root element box, click the browse button to the right. The **Choose root element** dialog appears.
4. Select the element you want to be the root from the tree. Double-click an element in the tree to expand the node and see its child elements.
5. Click **Ok**.
6. In the **Database** box, select a JDBC driver and a data source from the menus.
7. Enter the User name and password of an account on the database.
8. Click **Ok**. The selected XML schema and database tables will appear in the XML Schema Panel and Database Schema panel, respectively.

➔ To open an existing XTM:

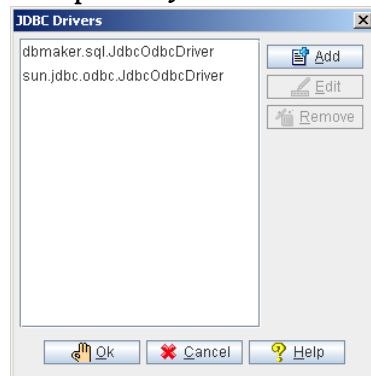
1. Select **File > Open** from the menu bar.
2. Select the XSL file that you want to open using the file chooser.
3. In the **File path** box, type a file path or select the browse button to locate a source schema file (XML, XSD or DTD).
4. To select a root element different from the one indicated in the Root element box, click the browse button to the right. The **Choose root element** dialog appears.
5. Select the element you want to be the root from the tree. Double-click an element in the tree to expand the node and see its child elements.
6. Click **Ok**.
7. In the **Database** box, select a JDBC driver and a data source from the menus.
8. Enter the User name and password of an account on the database.
9. Click **Ok**. The selected XML schema and database tables will appear in the XML Schema Panel and Database Schema panel, respectively.

## Adding a New JDBC Driver

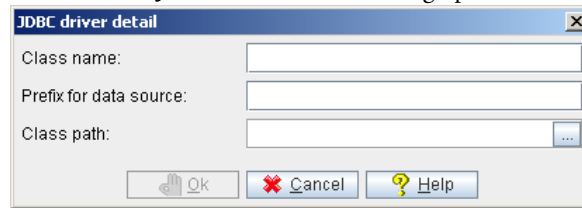
It may be necessary to use a different JDBC driver than the ones provided. It is possible to connect to databases provided by different vendors as long as the JDBC driver is present. The XTM tool provides this function through the Options menu. It is not necessary to have an XTM open to add a new driver.

You may also edit or remove drivers that you have added. The default drivers cannot be removed or edited.

- ➡ To add a new JDBC driver
1. Select **Options > JDBC Drivers** from the menu bar. The **JDBC Drivers** window opens.



2. Click **Add**. The **JDBC driver detail** dialog opens.



3. Type a class name and prefix for data source into the appropriate text boxes.
4. Type a class path or click the browse button to navigate to the location of the driver.
5. Click **Ok**.
6. The new driver will appear in the list of JDBC drivers. Click **Ok**.

## 9.2.7 MAPPING XPATH STATEMENTS TO XTM OBJECT NODES

After creating a new XTM, the next step is to create a map between the nodes in the XML schema and the database tables. First you will need to add database tables to the XTM object tree, and then you will need to add XML schema paths to properly map from the address in the XML schema where the data is located.

To create the structure of the XTM object tree, drag-and-drop tables from the database into the desired location in the XTM object tree. The first node must be dropped onto the root node; subsequent nodes can be dropped into the root node or any other node that represents a table. Tables cannot be dropped into a node representing a column. When you drop a table, it will appear in the XTM column with all of the columns visible.

After the XTM object tree structure is complete, drag-and-drop nodes from the XML schema panel onto the desired node in the XTM object tree. A statement will appear in the xpath column that corresponds to the xpath representation of the element or attribute address in the XML schema. Furthermore, xpath statements will be preceded by a data property if the are numerical or binary data types. A data property statement will not precede character data types. The xpath data property and corresponding data types are summarized in Table

<b>XPATH DATA PROPERTY</b>	<b>SQL DATA TYPE</b>
Character data	Char
	Varchar
	Longvarchar
	Longvarbinary



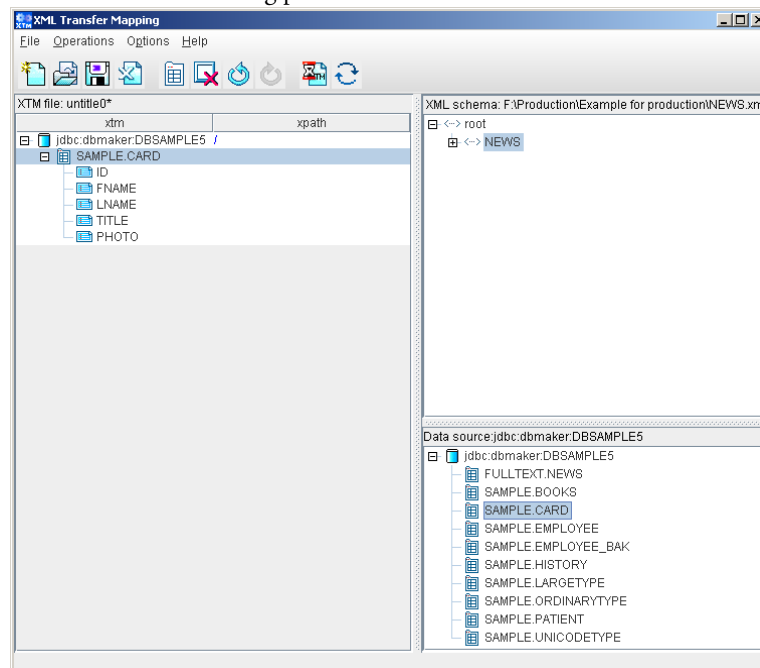
XPATH DATA PROPERTY	SQL DATA TYPE
	File
	Nchar
	Nvarchar
	Nclob
Numerical data	Serial
	Smallint
	Int
	Float
	Double
	Decimal
Normalize-space (binary) data	Binary
	Date
	Time
	Timestamp

Table 8-11: xpath data properties and corresponding SQL data types

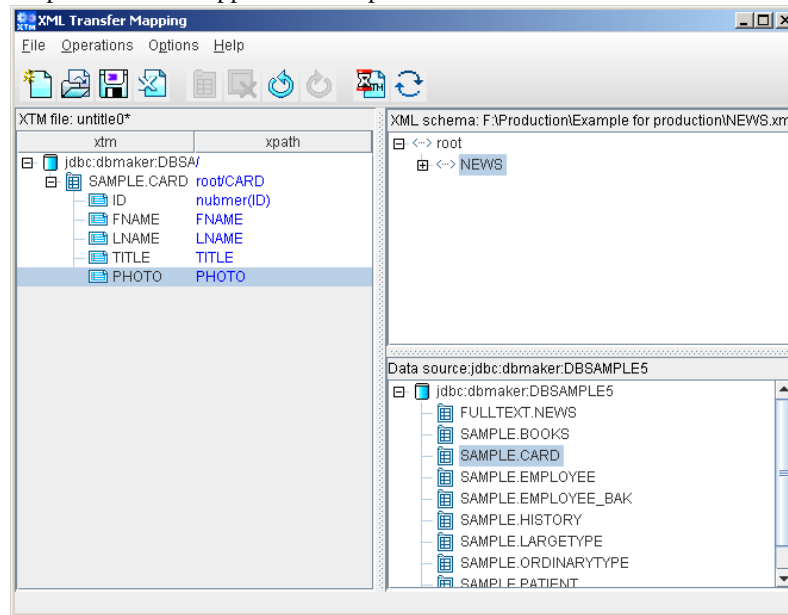


To build the structure of an XTM:

1. Create a new XTM or open an existing XTM:
2. Drag the table that you want to import data from the Database schema panel and drop to the xtm column of the XTM editing panel.



3. Drag-and drop the elements or attributes to the XTM nodes that you want to map the data to. The xpath statement appears in the xpath column.



4. When you are finished, click **File > Save**.
5. Browse to the folder where you want to store the XTM and type a file name. The file will automatically be saved with the extension **.XSL**.

## 9.2.8 EXECUTING AN XTM

After creating the XTM and saving it as an XSL file, you can pass data from the source XML file to the database by executing the XTM. When executing the XTM, you can choose to save the data as an XSL file and run the XTM, or save the transformation as an SQL script. Saving as an SQL script will not enter any data into the database. You must execute the script to enter the data into the database.

If you are trying to automate data transfer using a data transfer API or stored procedure, then you should save the transformation as an XSL file.

### Saving an XTM as an SQL Script

Saving an XTM as an SQL script allows you to create an SQL script that will perform the same operation on the database as executing the XTM file, only it stores the equivalent SQL commands in a script file. This method will not actually store any data in the database. Be sure to save the XTM as an XSL file before performing this operation, as it will not save either XML data or the transformation other than in the form of the SQL script.



To execute an XTM and save output as an SQL script:

1. Click **Operations > Run**. The **Execute XTM** window will appear.
2. Click **Save as SQL Script**.
3. In the **Save as SQL Script** box, enter the full path and file name for the XSL file, or select a file and path by clicking the browse button.
4. In the **Source XML** box, enter a full path and file name for the XML file to import data from, or select a file and path by clicking the browse button.
5. Click **OK**. The XTM Tool will create an SQL script. You can run the SQL script from the dmSQL prompt or using the JDBC tool to enter data into the database.

➤ Example:

SQL script output:

```
INSERT INTO DELPHI.CHINESE (ID,TEXT) VALUES (?,?);
1,'lobdir1\clobfile0.txt';
2,'lobdir1\clobfile1.txt';
3,'lobdir1\clobfile2.txt';
4,'lobdir1\clobfile3.txt';
5,'lobdir1\clobfile4.txt';
...
```

## Saving an XTM as an XSL File and Executing

Saving the XTM as an XSL file and executing performs the same operation as the XTM API or stored procedure. Executing the XTM file allows you to find errors in a transformation before you have automated it, and allows you to test the output for a given transformation to ensure that it produces the desired result.



To execute an XTM and save output as an XSL file:

1. Click **Operations > Run**. The **Execute XTM** window will appear.
2. Click **Save as XTM and run**.
3. In the **Save as XTM and Run** box, enter the full path and file name for the XSL file, or select a file and path by clicking the browse button.
4. In the **Source XML** box, enter a full path and file name for the XML file to import data from, or select a file and path by clicking the browse button.
5. Click **OK**. The XTM Tool will create an XSL file and add new data to the selected tables in the database.

➡ Example:

XSL output:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0"
xmlns:xtm="dbmaster.xml.xtm.XMLTransferMap"
extension-element-prefixes="xtm">
<xsl:template match="/">
<xtm:connect driver="dbmaster.sql.JdbcOdbcDriver"
datasource="jdbc:dbmaster:DBSAMPLE4">
<xtm:table owner="DELPHI" name="CHINESE" select="/root/CHINESE">
<xtm:column name="ID" select="number(@ID)"/>
<xtm:column name="TEXT" select="@TEXT"/>
</xtm:table>
</xtm:connect>
</xsl:template>
</xsl:stylesheet>
sel.xml ', 'c:\temp\case1.xml');
```

## 9.2.9 XTM API FUNCTIONS

After you have created an XTM file you are ready to automate the transfer process. The XTM API allows you to automate the process. DBMaster provides four XTM Transfer APIs; the XTM API in C++, the XTM API in C, the XTM API in Java, and the XTM API stored procedure.

Please consult '*XTT/XTM User's Guide*' for the detail

## 10. ROLLOVER

ROLLOVER is a command line tool for backup-restore. You can easily use it with the following description.

## 10.1 ROLLOVER usage

User can also use the Rollover which is a command line tool to restore the database. Its principle is same as the Restore Database of JServer Manager.

The usage of rollover is like:

**rollover database\_name [-i inifile] [-r rtime] [-h hisfile] [-m foMapfile] [-f FOtype]**

There are five optional parameters in the square bracket:

- i specifies full path of dmconfig.ini. If user specifies the dmconfig.ini to restore, rollover will replace the database section in system dmconfig.ini with the corresponding database section in specified dmconfig.ini, otherwise, DBMaster will not restore dmconfig.ini.
- r denotes the time that database should be restored to. The option -r is the first method to specify rtime, the second method is to add DB\_RUNTIME keyword into system dmconfig.ini or backup dmconfig.ini which will be specified to restore database. If neither -r option nor DB\_RUNTIME keyword, the rtime will be the current time.
- h gives full path of dmBackup.his. The default is "DB\_BKDIR/dmBackup.his".
- m gives full path of dmFoMap.his. The default is "DB\_BKDIR/FO/dmFoMap.his".
- f specifies which type FO files to be restored. There are four values, the value of 0 means no FO files to be restored; the value of 1 will restore system FO; value of 2 will restore user FO and value of 3 will restore all FO. The default value is 3.

# 11. DMRestoreTB

By using the JTOOL of DBMaster, we can easily restore the database to the specific time. But sometimes we only want to restore some tables to the last full-backup time and don't influence the other tables. Now we provide a new tool in DBMaster5.1 "DMRestoreTB" to do the requirement.

## 11.1 How to use

The DMRestoreTB is a command-line tool and we must setup environment and give it some parameters.

### 11.1.1 ENVIRONMENT:

---

Now let's see the running environment:

1. We must run the tool in the database directory.
2. We must drop the table that we want to restore first in the current database.
3. We must check the all settings on the dmconfig.ini are correct, especially the setting "DB\_BKDIR". It must be assigned to the full-backup database directory.
4. We must also check the full-backup files in the DB\_BKDIR are correct.
5. We must also check the backup history files are correct.
6. The password of SYSADM in the current database and the full-backup database must be the same.

### 11.1.2 INPUT PARAMETERS

---

By running the DMRestoreTB directly, we can get the following message:

```
DMRestoreTB version (5.1)
Usage: DMRestoreTB <db_name> <table_name> <password> <online> [port_num]
      <db_name>      : Database name
      <table_name>   : Table name wants to restore
      <password>     : SYSADM's password. Please input "" if the password is empty.
      <online>       : 1 -> online restore table
                     0 -> offline restore table
      [port_num]    : Optional, one free port in your system
```

To run the DMRestoreTB, we must provide at least 4 parameters. They are <db\_name>, <table\_name>, <password> and <online>, respectively.

1. The <db\_name> means that the database name we want to restore.
2. The <table\_name> means that the table name we want to restore. Each time we can only restore one table.
3. The <password> means that the SYSADM's password. If the password is empty, we must input "". We must check the SYSADM's password in the current database and full-backup database are the same.
4. The <online> means that the database status is online or not when we do the restoring work. If we want to restore table from the full-backup database when the current database is online, we must set the parameter to 1. We must note that the above 4 parameters are necessary.
5. The last parameter [port\_num] is an optional parameter. Because we will use the port 59998 when restoring the table, so the program would fail when the port 59998 is unavailable. If the port 59998 is unavailable, we can specific another port to the program by using the parameter [port\_num].

### 11.1.3 EXAMPLES

---

Let's see some examples:

➤ Example 1:

If we want to restore a table "salary" in the database "employee" and the database status is online and the password of SYSADM is 1234, the command would be:

```
DMRestoreTB employee salary 1234 1
```

**NOTE** *we use the default port 59998*

➤ Example 2:

If we want to restore a table "storage locations" in the database "inventory" and the database status is offline and the password of SYSADM is empty, the command would be:

```
DMRestoreTB inventory storagelocations "" 0 4287
```

**NOTE** *we use the port 4287*