DBMaster

Reference Guide for SQL Server 2008 Migration to DBMaster 5.1

Version: 02.00

Author: DBMaster Support Team and Research & Development Division, SYSCOM Computer Engineering CO.
Document No: 51/DBM51-T05272010-03-MRSQ

Publication Date: June 14, 2010

Content

1. Overview

This document introduces some related messages about migrating SQL Server 2008 to DBMaster 5.1. Migration processes include schema transitions, DML mappings, Data storage and so on. With this document, the users could easily understand the pros and cons between SQL Server and DBMaster. Users would be also aware of characteristics of DBMaster and SQL Server.

SQL Server is known as an integrated RDBMS product. It is used to individuals and enterprises. The SQL Server 2008 SQL Query analyzer adds Intelligent Functions for Cue. However, its many limitations, such as platform dependence on Windows, are sometimes regarded very inadequate if you want to migrate your database to other platforms. It is very important for an IT professional to choose appropriate methods.

In addition, users who are not familiar with DBMaster may also find the helpful information in this manual. It will be very easy to catch the similar idea from DBMaster that they had already known in SQL Server. It can shorten learning curve for users.

We will introduce DBMaster in the following aspects:

- Migrate SQL Server database to DBMaster 5.1.
- Create ANSI-compliant names.
- Customize Users, Tables, Indexes, and Tablespaces.
- Remove and rename database objects if they are reserved words in DBMaster.
- Customize the default data type mapping rules.
- Migrate groups, users, tables, primary keys, foreign keys, unique constraints, indexes, rules, check constraints, views, triggers, stored procedures, AP, user-defined types and privileges to DBMaster.

2. Analyze the current system

Firstly, we should analyze the system before migrating a database from SQL Server to DBMaster in some aspects, through which we can evaluate the workloads and costs of the migration.

For instance, we should analyze current operating system and get to know what system we use, Windows or any other. Different operating systems have different characteristics. We also need to know what should be considered as emphasis and difficulty in the migration process.

System analyses include both AP system analyses and DB system analyses. In the following chapters we will introduce them in two aspects.

2.1 Analyze AP system

Users should understand system architectures first, and know what technologies have been used. Such as Hiberanate, NHibernate, C, C++, Java, .Net, PHP, Ruby, etc.

In addition, the driver type is also important; users should know which one was used. For example: JDBC, ODBC, DCI, OLEDB, and so on.

Next, analyze the hierarchical structure of AP system, for example: Client/Server, Browser/Server, N-Tier.

Last, users need to analyze the special feature of SQL Server through which users get to know how to convert them to DBMaster. For the special feature, we should consider the following aspects.

We should consider the following aspects before migration.

- Special features of SQL Server
- The workaround of SQL Server special feature
- Special syntax of SQL Server
- How to convert special syntax into DBMaster

2.2 Analyze Database Objects

For a database, we should analyze all database objects. First of all, we have to know how many database objects should be migrated, for example: tables, views, trigger, etc. We need evaluate how much space is required for storing data.

Next, we should analyze all tables' structures and get to know what contents of these tables will be stored. It can help user divide tables into different table spaces to improve performance. Then users can begin preparing for creating a corresponding database with DBMaster.

There are many differences between SQL Server and DBMaster. So we should consider these differences in advance. We will introduce some aspects as followings:

- Data types belong to SQL Server but not apply to DBMaster.
- Data types belong to DBMaster but not apply to SQL Server.
- Built-functions belong to SQL Server but not apply to DBMaster.
- Built-functions belong to DBMaster but not apply to SQL Server.
- Indexes belong to SQL Server but not apply to DBMaster.
- Indexes belong to DBMaster but not apply to SQL Server.

In addition, users can evaluate workloads with above analyses. It will help user estimate the costs of migration.

3. Setup migration environment

We mainly introduce environment and which aspects users should pay high attention to in this section.

We must ensure the application can run normally with SQL Server before we do any works. Then we will install DBMaster and create a database. Certainly, before that users should reserve enough disk space for DBMaster database db files. For convenience, you can install DBMaster in same machine with SQL Server (on Windows platform).

Next, we need to adjust or configure web server if users' system has web server that is used for deployment and testing web applications.

We also need to pay attention to the following aspects before migrating a database from SQL Server to DBMaster.

- Adjust DBMaster configure parameters if necessary.
- Enroll settings in windows system.
- Special workaround for migration.
- DSN or environment variables should be installed.
  - UnixODBC in Linux system (if users move to Linux system)
  - ODBC Driver Manager in Windows
- Which DBMaster.(normal or bundle)
- Which DBMaster server is running? (dmserver or dmservice)
  - dmservice is only for windows system, customer can install it as a windows service with JServer Manager tool or dmsvcutl.exe. Then, the user can set the database service as Auto Start when OS being started.

4. Methods for migrating table schema and data

Database Migration involves all of the database objects. But we only introduce the migration methods for table schema and data in this section and other aspects will be described in chapter 6 and chapter 7.

4.1 Database transfer tools

4.1.1 SQL SERVER IMPORT AND EXPORT WIZARD

The SQL Server provides a tool-**SQL Server Import and Export Wizard**. It offers a simple method to transfer data from a source to a destination.

4.1.1.1 How to start the SQL Server Import and Export Wizard

1. On the Start menu, point to **All Programs >>Microsoft SQL Server 2008**, and then click **Import and Export Data**.
2. In Business Intelligence Development Studio, right-click the **SSIS Packages** folder, and then click **SSIS Import and Export Wizard**.
3. In *SQL Server Management Studio* (SSMS), connect to the Database Engine server type, expand Databases, right-click a database, point to Tasks, and then click **Import Data or Export data**.
4. In a command prompt window, run DTSWizard.exe, located in *C:\Program Files\Microsoft SQL Server\100\DTS\Binn*.

4.1.1.2 Operation steps

Users must ensure both SQL Server and DBMaster have been started before migration. In whole process of migration, there are two points we should pay more attention to when we use **SQL Server Import and Export Wizard**.

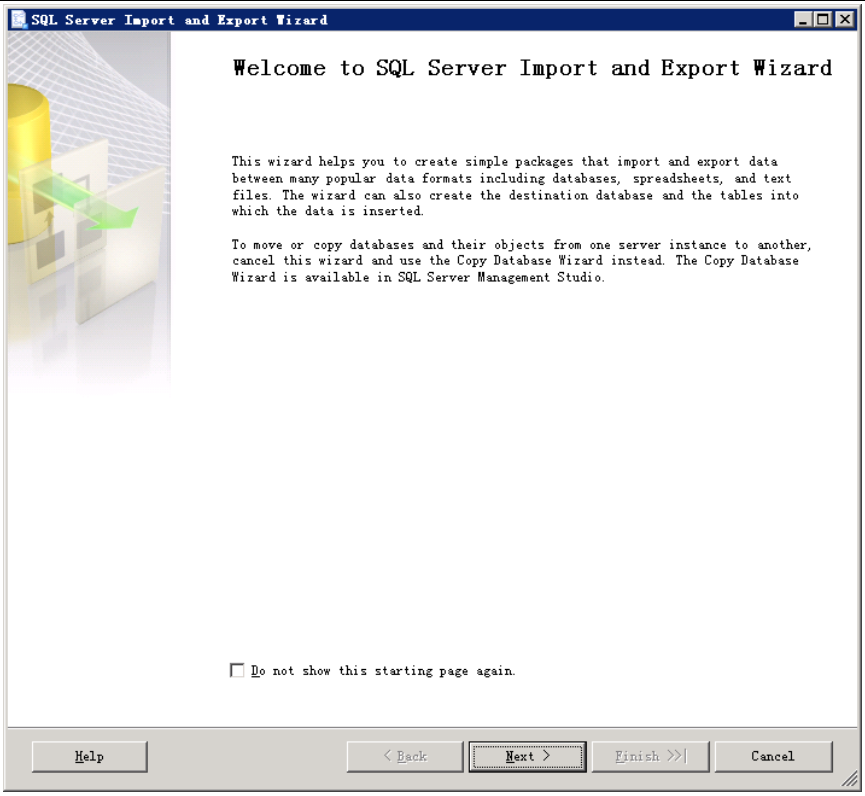
One point is about selecting provider. For example: ODBC Provider is available for migration from SQL Server to DBMaster by **SQL Server Import and Export Wizard**.

The other point is about adjusting mapping data type and editing SQL statement manually, we will introduce two methods – one is from SSIS import and export wizard, the other is from SSMS import and export wizard.

More detail information for each step will be introduced as followings.

4.1.1.2.1 Import and Export Wizard via SSMS

Step 1: Start the **SQL Server Import and Export Wizard** via SQL Server Management Studio (SSMS).



Step 2: Open the wizard **Choose a Data Source** for selecting a data source. The available data sources include .NET Framework data providers, OLE DB providers, SQL Server Native Client providers, ADO.NET providers, Microsoft Office Excel, Microsoft Office Access, and the Flat File source.

You can set different options such as the authentication mode, the server name, the database name, and the file format that depend on the data source you have chosen.

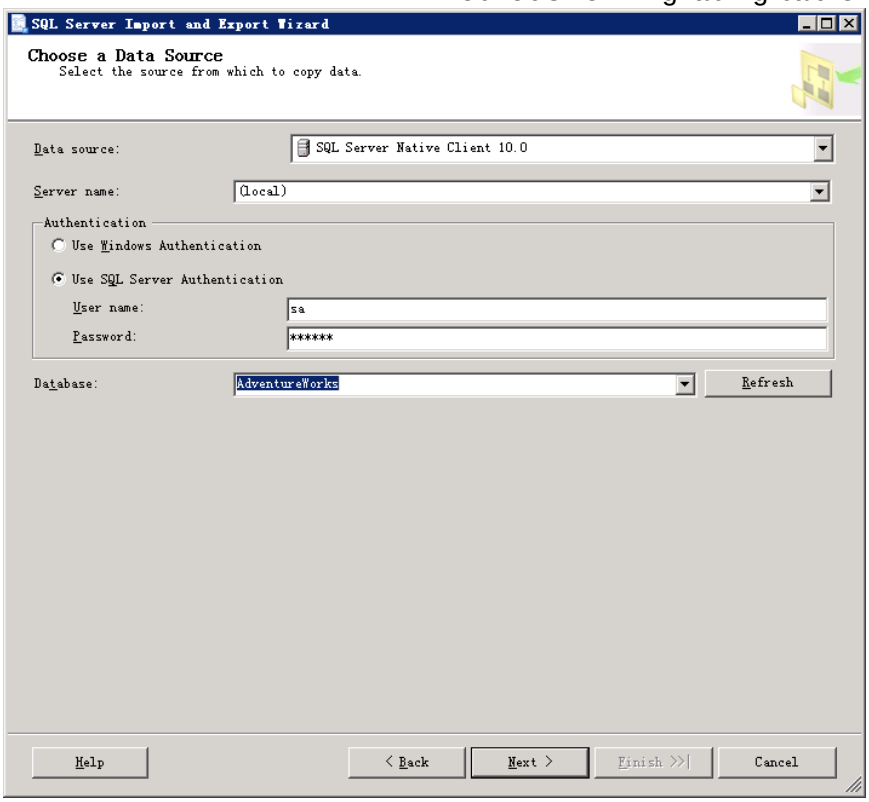
Here you can specify:

Data Source: SQL Server Native Client10.0 or other supported.

Serve name: (local) or available server name.

Authentication identify: you can use windows mode or SQL Server mode.

Database: specify the name of the source database which need migrated.



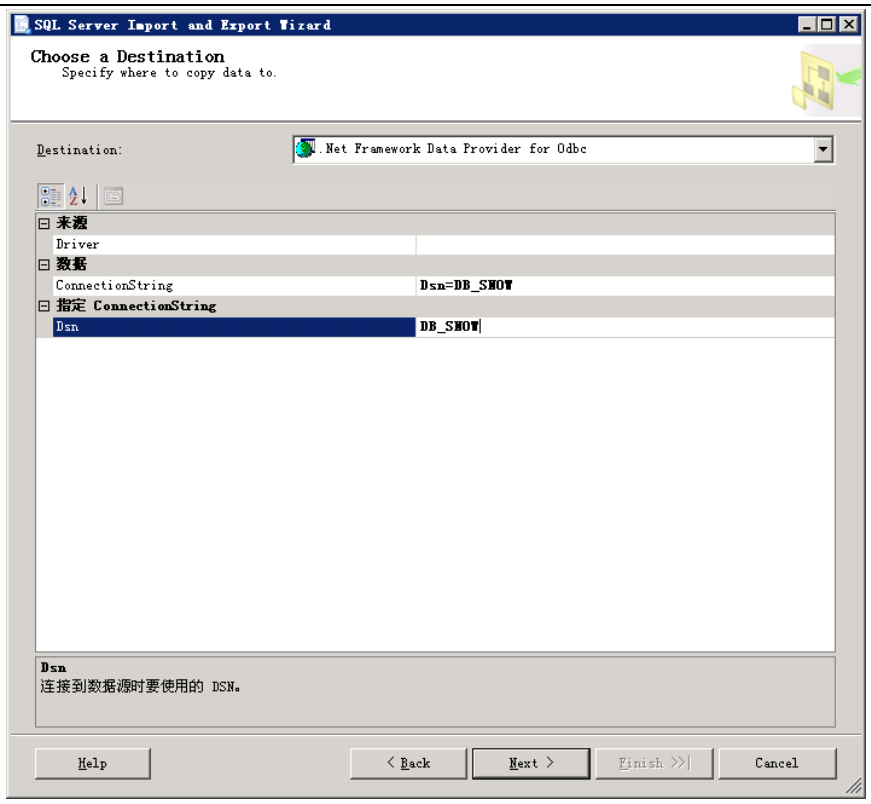
Step 3: Open the wizard **Choose a Destination**. The available data destination includes .NET Framework data providers, OLE DB providers, SQL Server Native Client, Excel, Access, and the Flat File destination.

Here you can specify:

Target: select ".Net Framework Date Provider for Odbc".

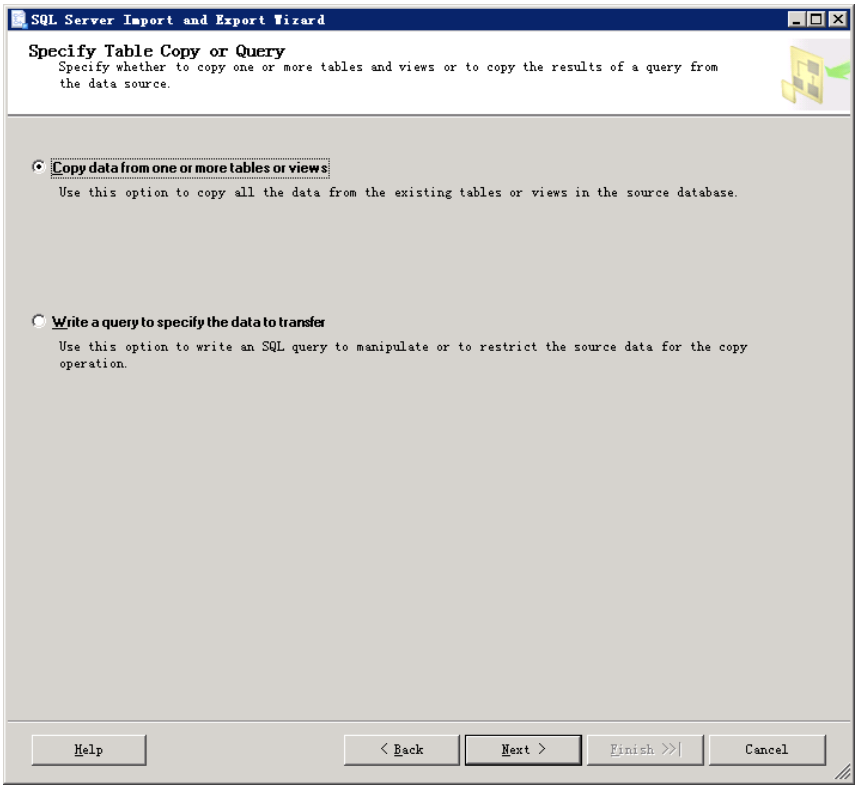
Driver: NULL. Don't need any parameters.

DSN: the DSN name created by ODBC data source. You should build DSN in advance and ensure connection available.

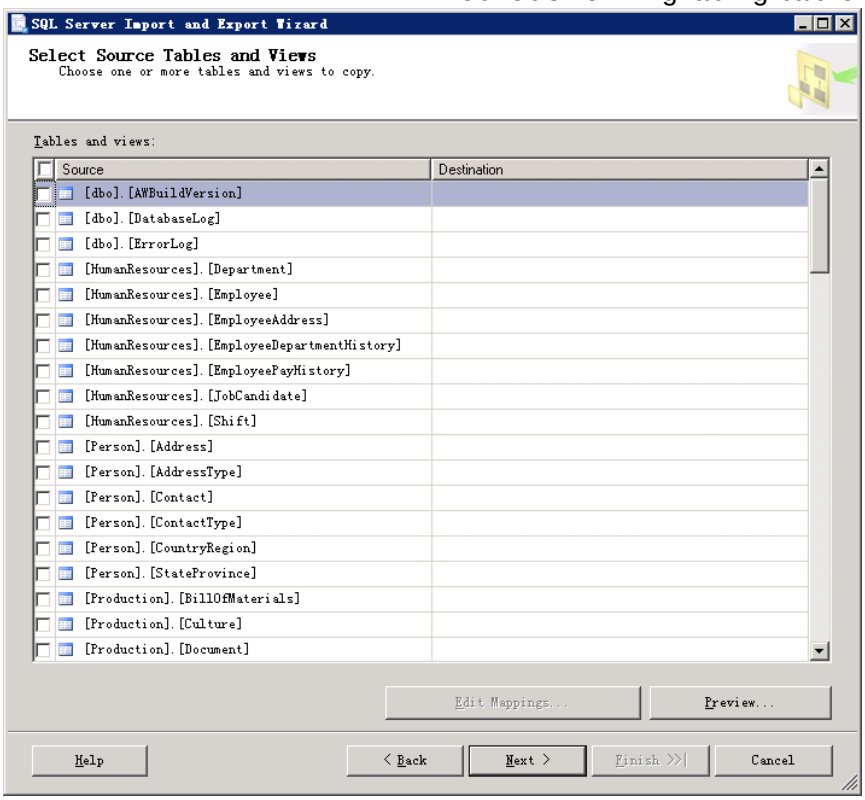


Step 4: Open the wizard **Specify Table Copy or Query.**

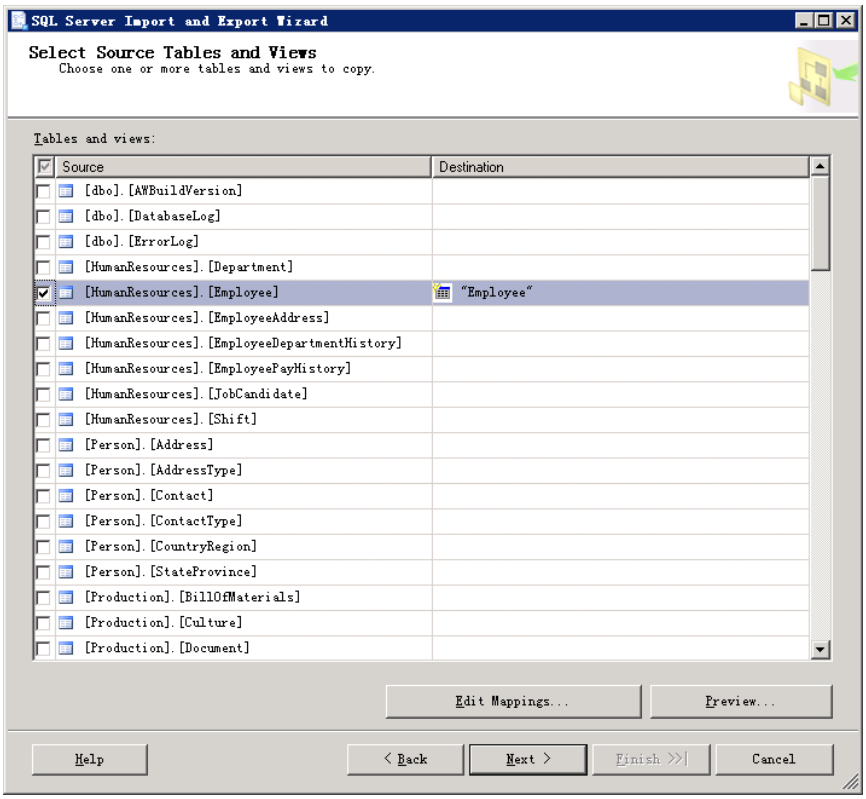
There are two options provided, including replicating one or more tables or view's data and editing query SQL. You can choose one of them to specify which data will be transmitted.



Step 5: Open the wizard **Select Source Tables and Views**



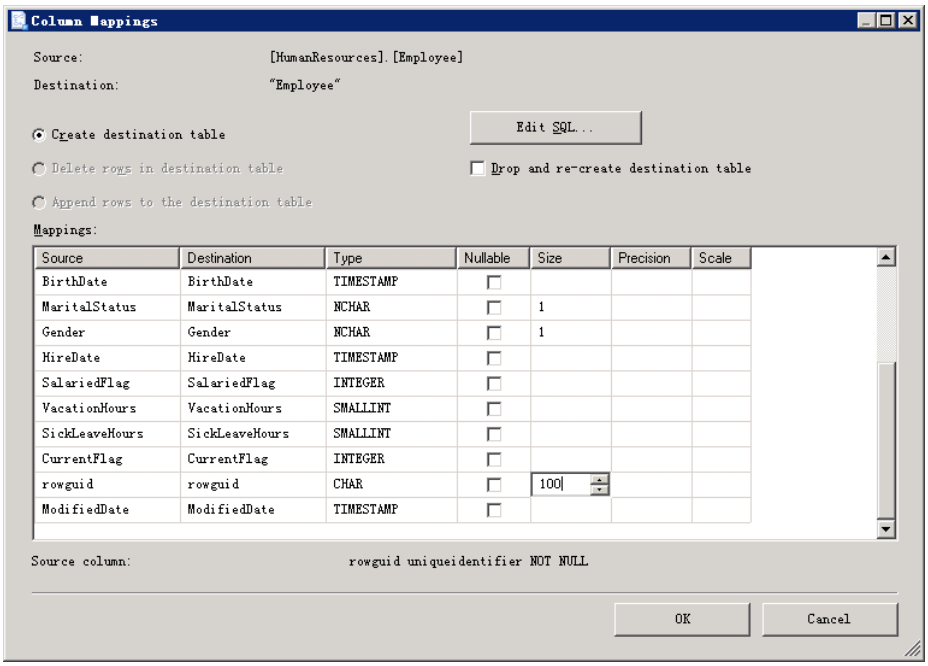
♦ First is selection. You can specify whether to copy one or more tables and views from the data source, and finally select the tables and/or views to copy.



♦ Second, click **Edit Mappings…** button and open the page **Colunm Mappings** in which you can adjust column data types to be suitable for DBMaster. Select one table and change the mappings between source columns and destination columns, or change the metadata of destination columns

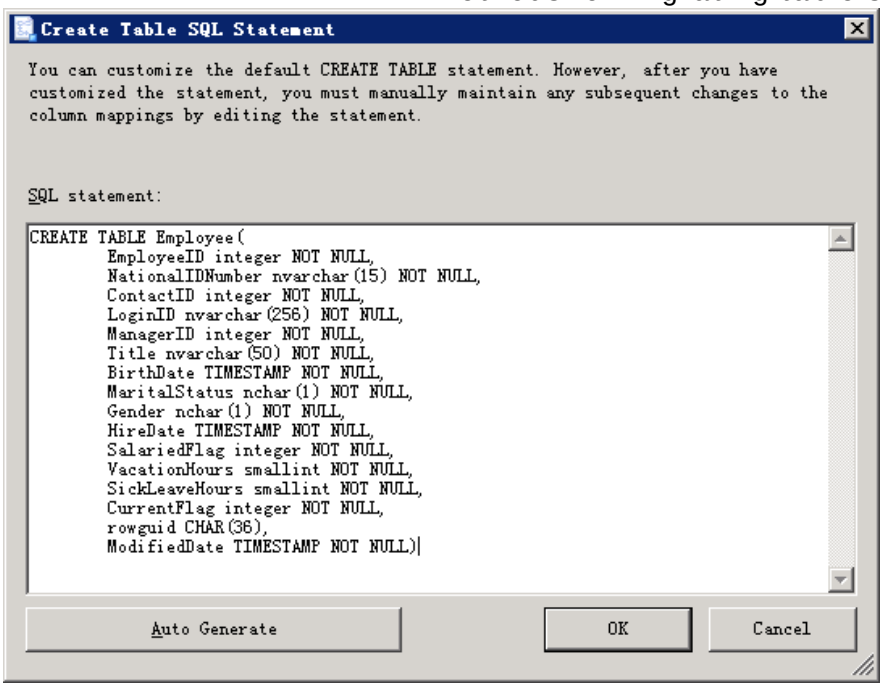
Include:

- Map source columns to different destination columns.
- Change the data type in the destination column.
- Set the length of columns with character data types.
- Set the precision and scale of columns with numeric data types.
- Specify whether the column can contain null values.



♦ You also can create a new destination table by select the **Create destination table** option. Select the **Delete and recreate destination table** option will drop the destination table and then re-created, enable identity inserts.

♦ The most important step is clicking the **Edit SQL** button. Click it and open the wizard **Create Table SQL Statement"** to check the SQL statement which generates automatically whether correct .If incorrect, please modify the problematic part and add the missing part to ensure the SQL statement correct and complete.

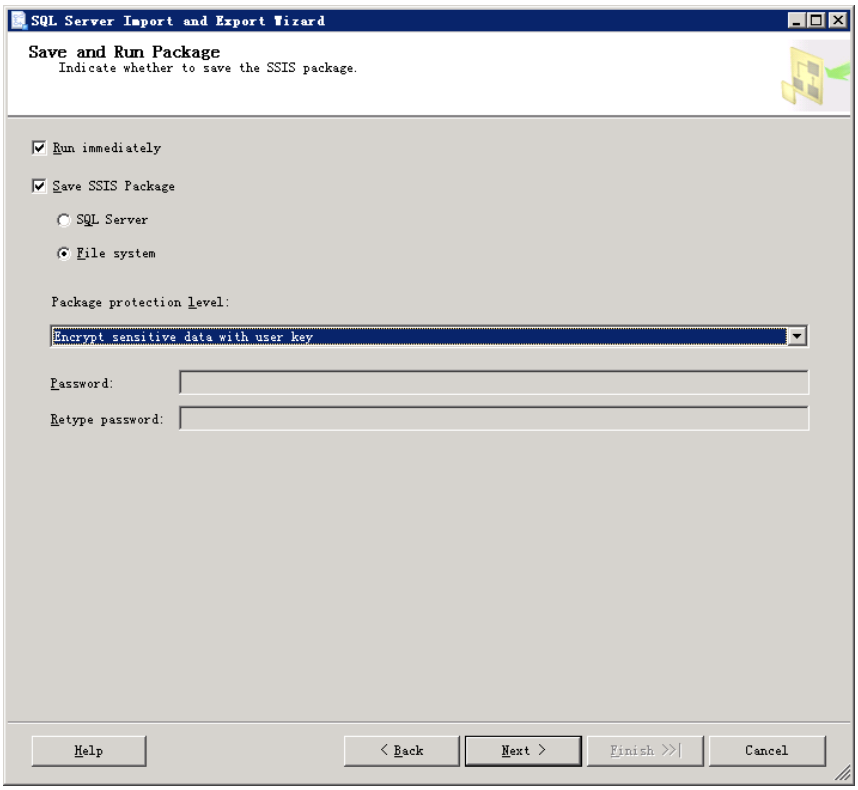


Step 6: Open the wizard **Save and Run Package**.

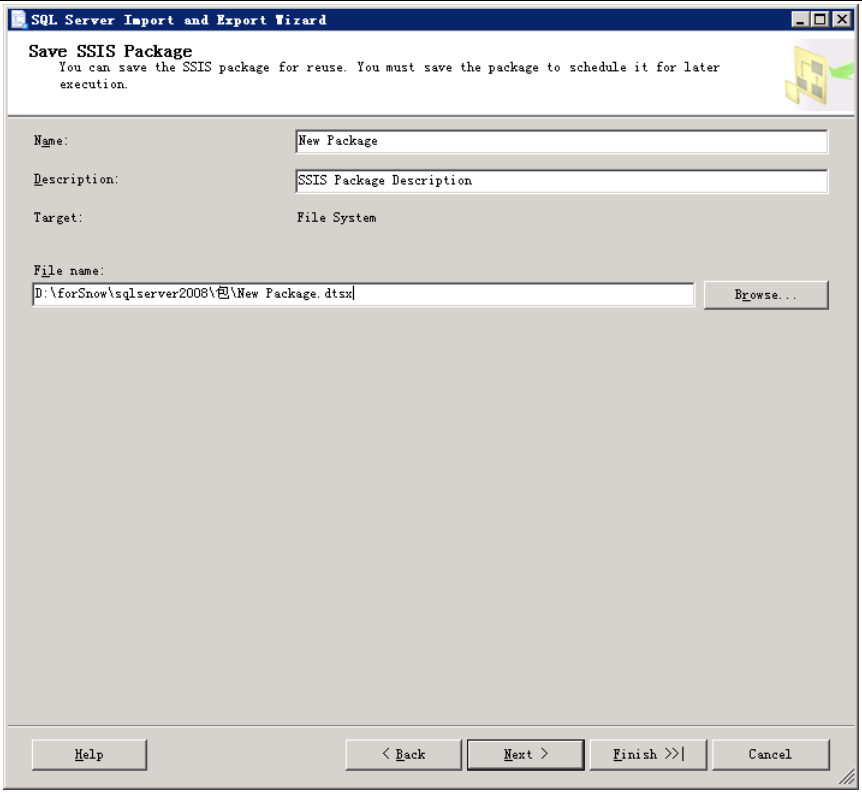
You can optionally save the package to the SQL Server msdb database or to the file system.

Execute immediately: By default, this check box is selected and the package runs immediately.

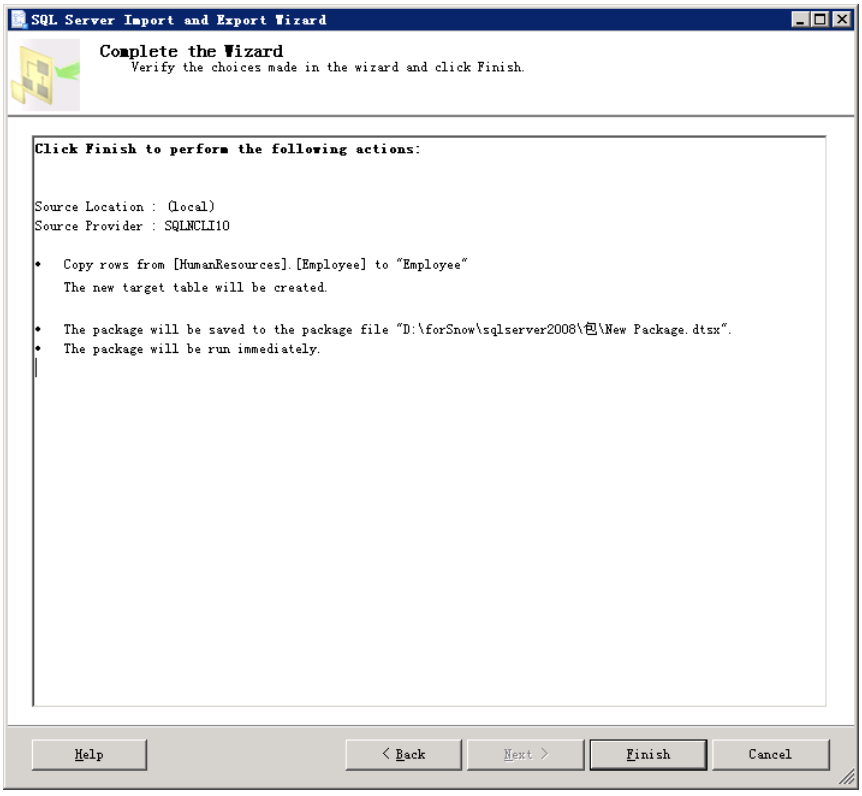
Save SSIS package: you can also decide whether to save the package to SQL Server or to the file system. If you select to save the package, you must also specify a package protection level. And if the protection level uses a password, please provide the password.



Step 7: Specify save path for SSIS package if you select **File system** in last step.



Step 8: Next



Step 9: Finish

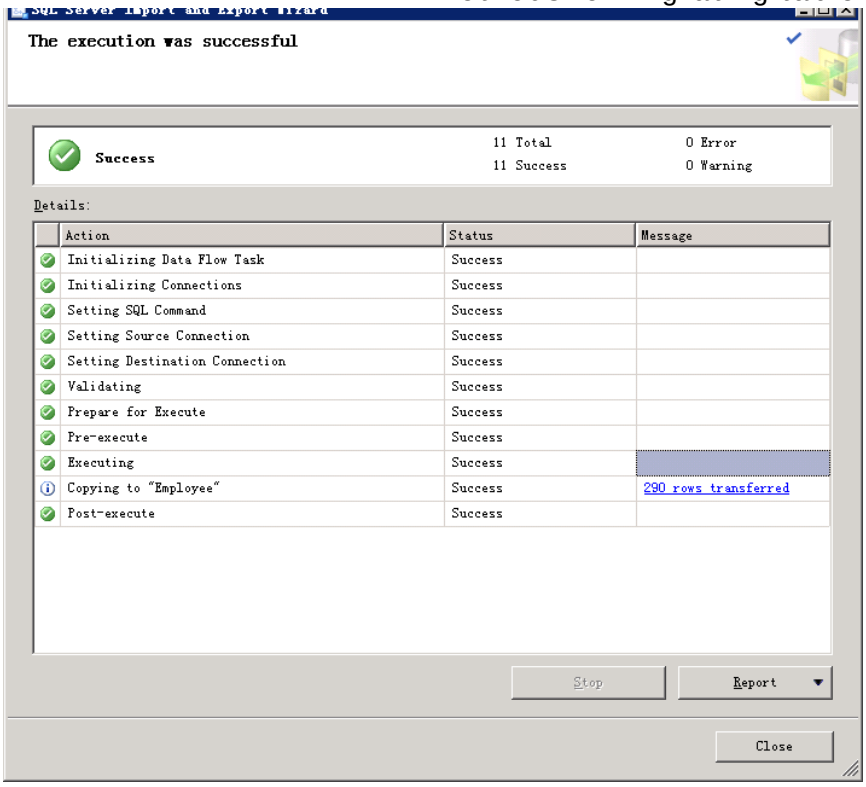


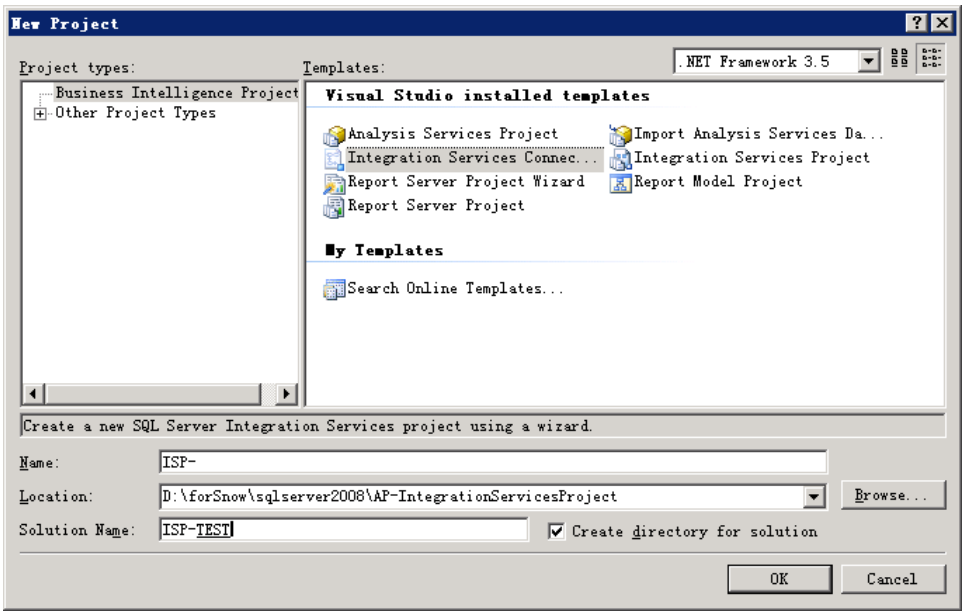
Table Employee migrates from SQL Server to DBMaster successfully.

4.1.1.2.2 Import and Export Wizard via SSIS

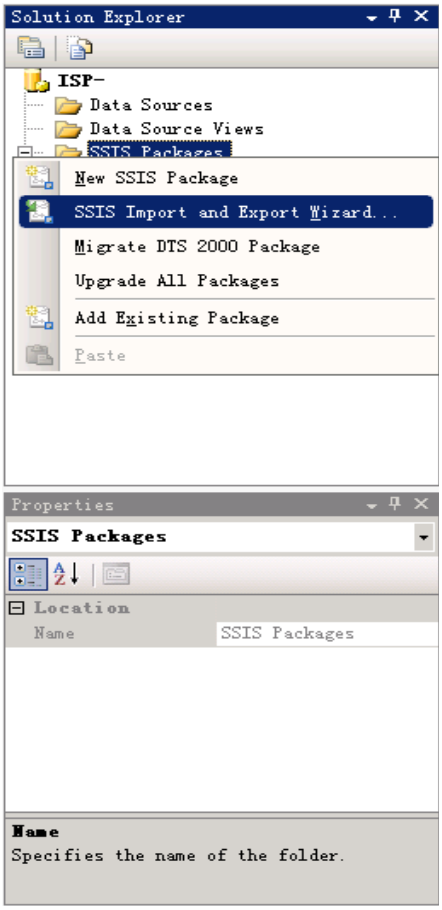
This step is to create a project before creating an Integration Services Package.

- Executing steps

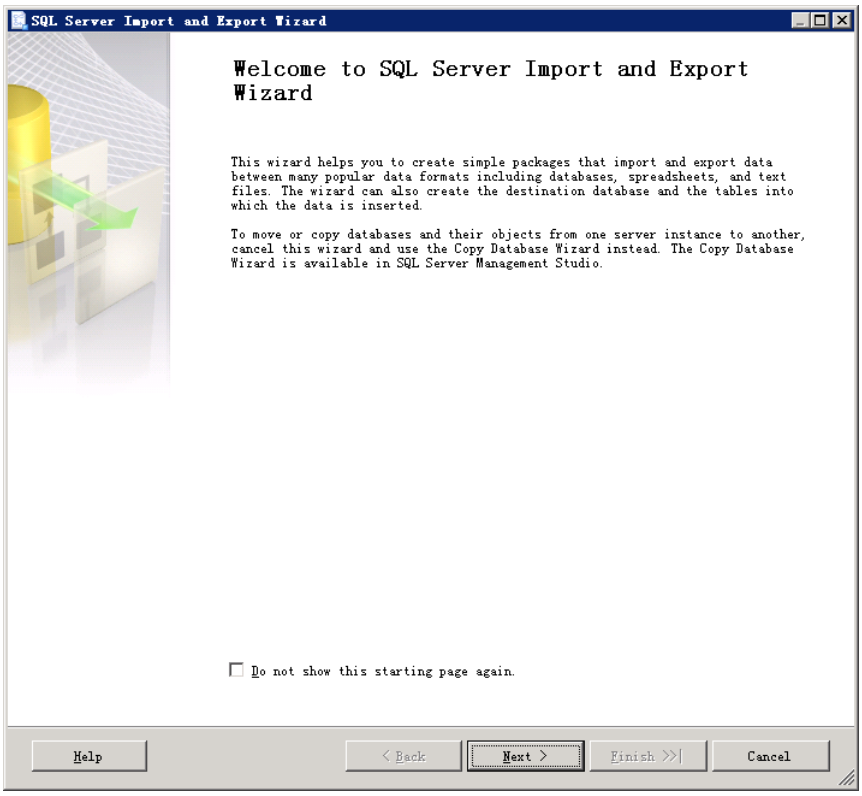
Step 1: program | SQL Server Business Intelligence Development Studio, create an Integration Services Project.



Step 2: Start the **SQL Server Import and Export Wizard** in two ways. One is right-clicking the **SSIS Packages** folder, and then clicking **SSIS Import and Export Wizard**. The other is In Business Intelligence Development Studio, on the Project menu, clicking **SSIS Import and Export Wizard**.



Come to the page **Welcome to SQL Server Import and Export Wizard**.



Step 3: this is different from SSMS “step 3 “

Here you must specify following parameters to find the DSN from ODBC data source:

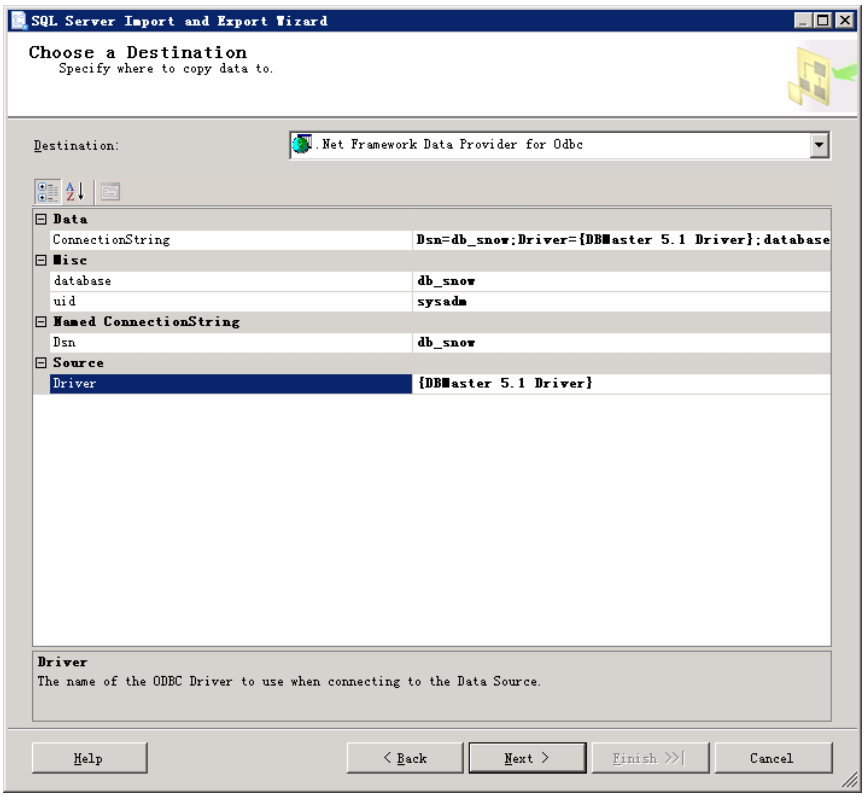
Target: select ".Net Framework Date Provider for "Odbc" provider.

ConnectionString: Dsn= DB_SNOW; Driver= {"Driver= {DBMaster 5.1 Driver}; Database= DB_SNOW; uid=sysadm; Pwd= ;"}

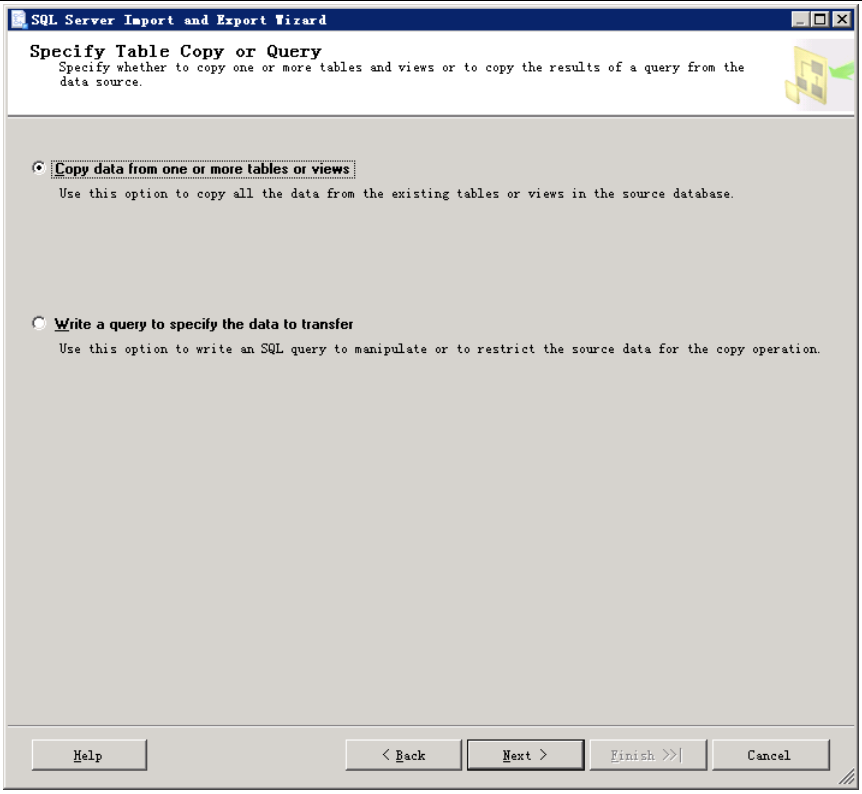
Driver: {DBMaster 5.1 Driver};

DSN: DB_SNOW

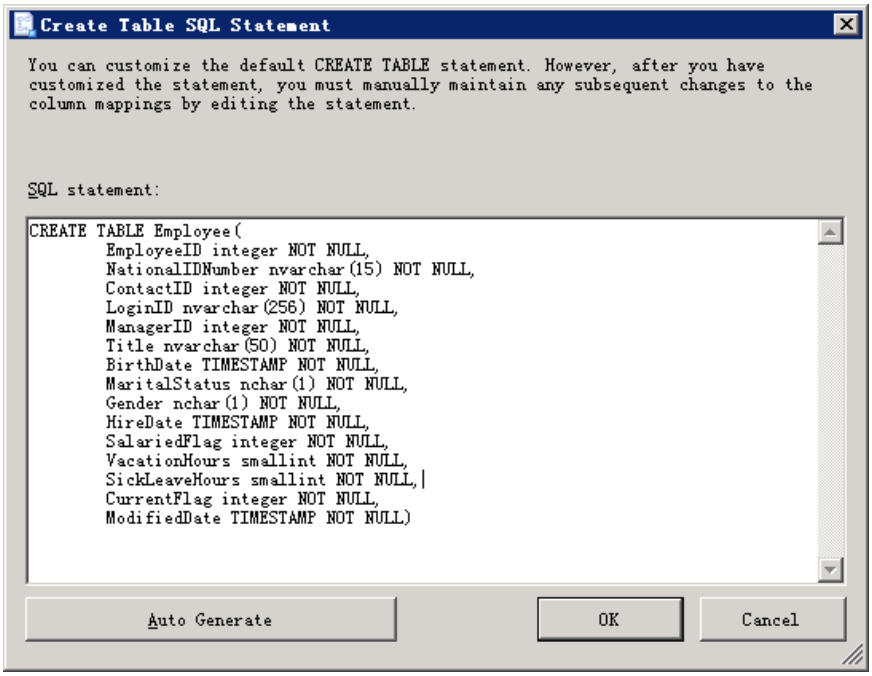
The DSN name is created by ODBC data source. You should ensure the connection is available after setting the System or the User DSN.



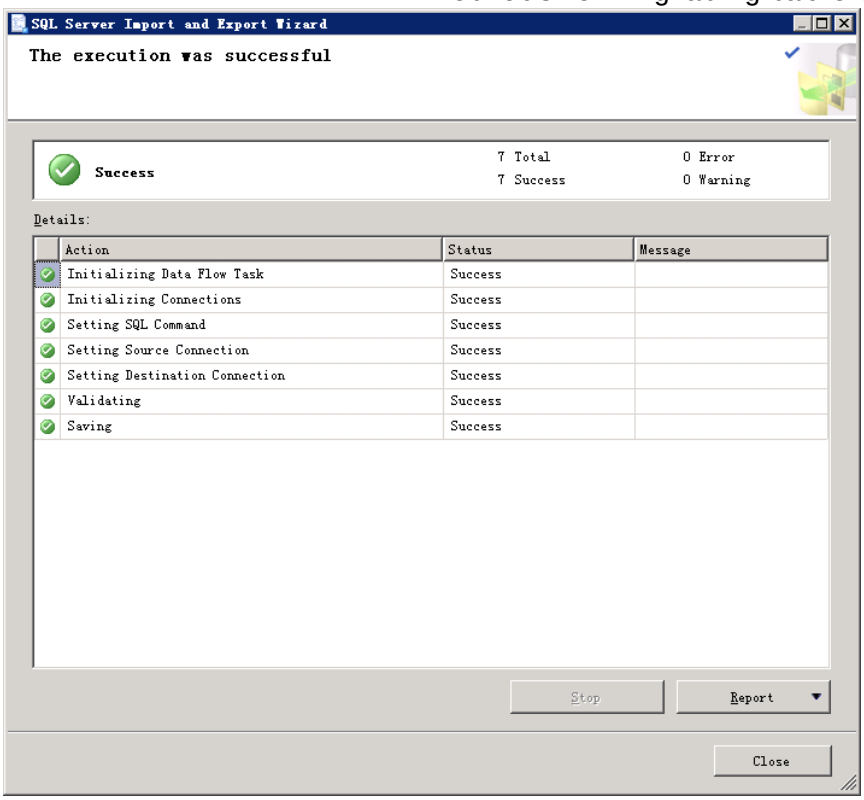
Step 4: Open the page **Specify Table Copy or Query**



Step 5: Select source tables and views

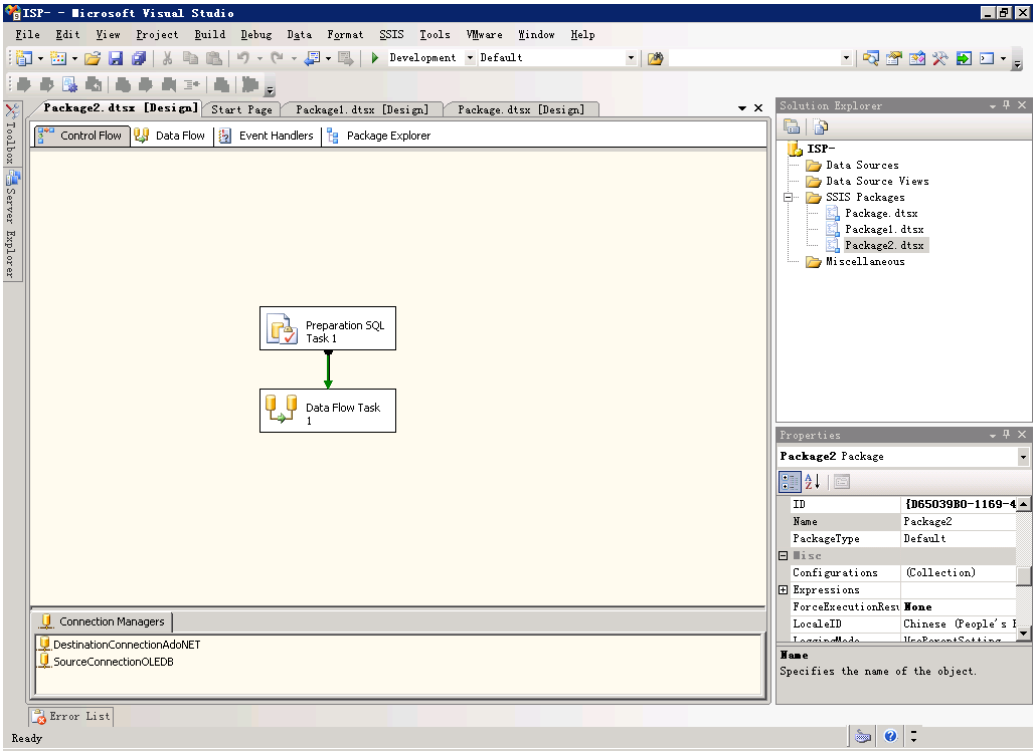


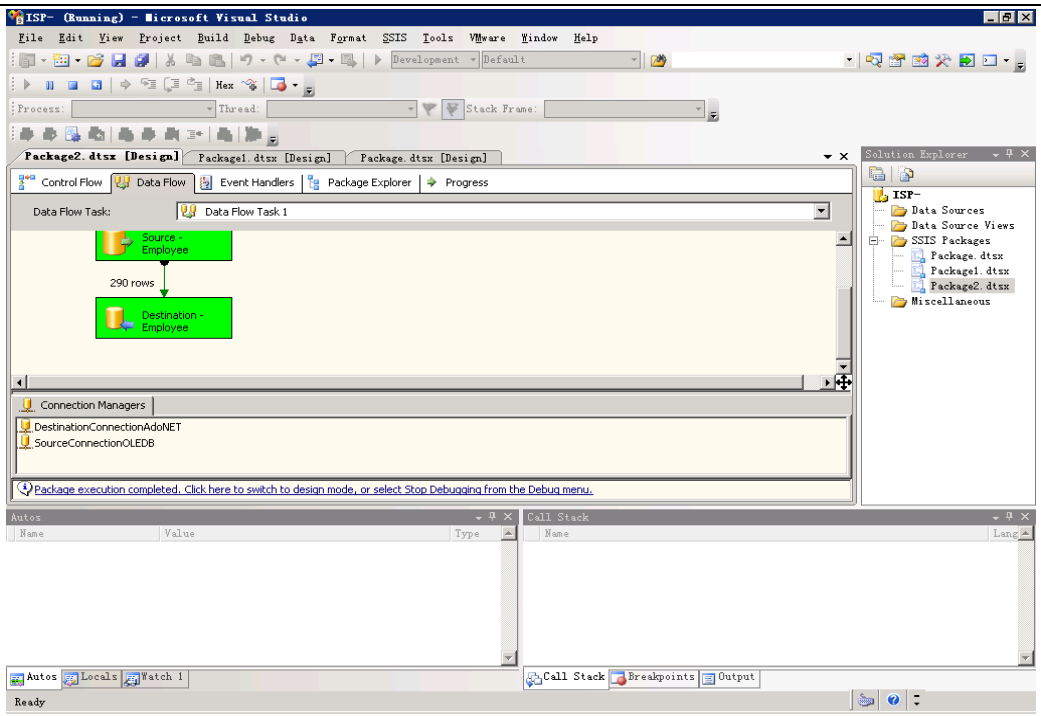
Finish



Steps 6: Run the package from the wizard

Worthy of noting is “Step 6”,"save and run a package”. It’s different from SSMS. If the wizard is started from an Integration Services Project in Business Intelligence Development Studio, you can’t run the package from the wizard. Instead, the package is added to the Integration Services Project from which you started the wizard. Then you can run the package in Business Intelligence Development Studio.





We can see 290 pieces of records migrated from SQL Server to DBMaster from the green arrowhead.

4.1.2 JDATATRANSFER TOOL IN DBMASTER

The Data Transfer Tool provides a user-friendly interface for transferring data in and out of the database. The tool performs the following functions:

- Import from text
- Import from XML file
- Import from ODBC
- Export to text
- Export to XML
- Batch transfer

For more information about performing each type of data transformation please reference *JDBA Toll* Chapter **Data Transfer.**

Here we mainly introduce **Import from ODBC**. DBMaster supports importing data from other data sources via ODBC. Other data sources may include other database engines such as Oracle, Microsoft SQL Server, etc.

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

Data may be imported by using three methods:

- Choose the tables directly
- With one or more SQL SELECT statements
- Via 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 which allow saving a 'map' of the data transformation to an XML batch file. The XML batch files are saved as 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.

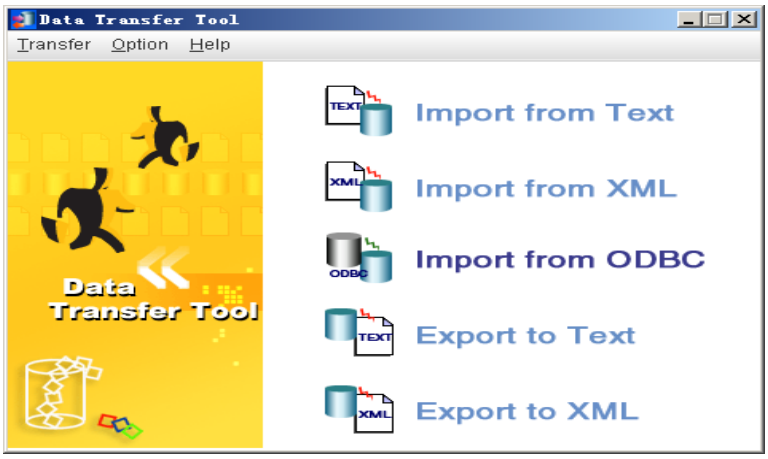
4.1.2.1 How to start the DBMaster JDataTransfer

The Data Transfer Tool is a separate application which can be started as GUI.

Start>programs>DBMaster 5.1>DataTransfer, or opened within *JDBA Tool*.

4.1.2.2 Execute steps Import from ODBC

Step 1: Open the **Data Transfer Tool** and select **Import from ODBC** option.

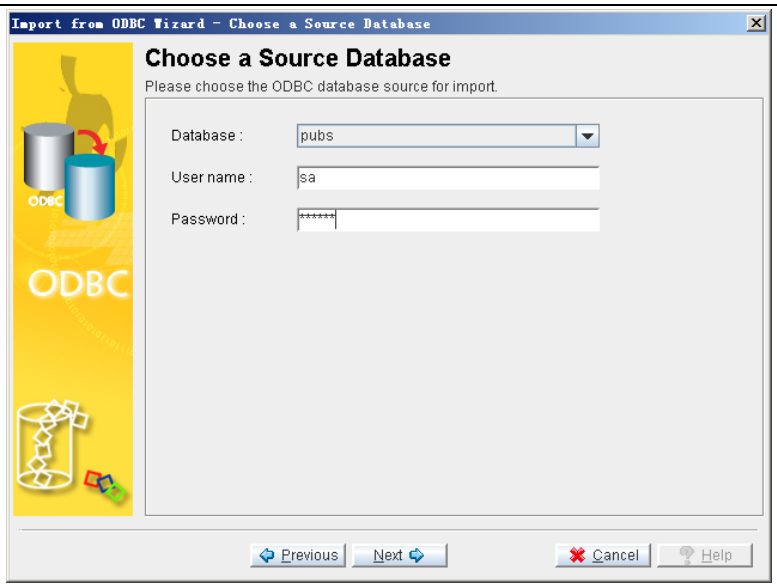


Step 2: Click **Next**. The **Choose a Source Database** window appears.

Database: Select the database to export data from in the Database menu.

Username: Enter a user name into the appropriate field.

Password: Enter corresponding password into the appropriate field.

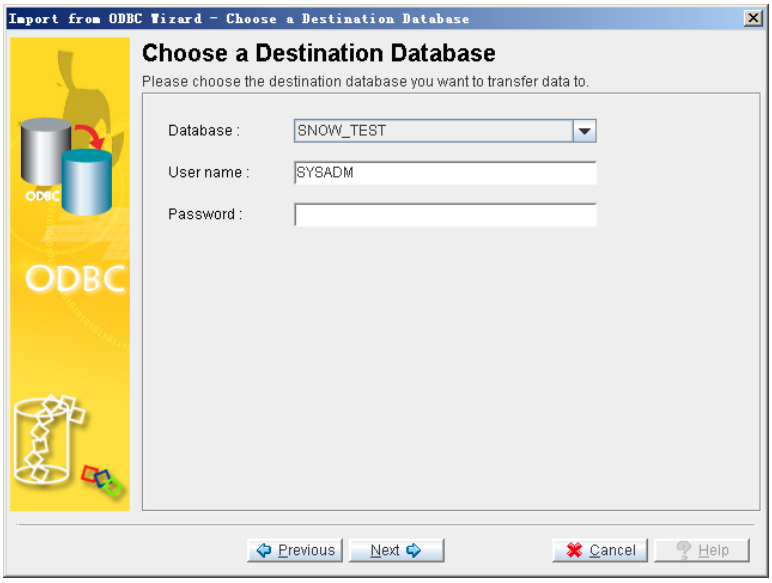


Step 3: Click **Next**. The **Choose a Destination Data Source** window appears.

Database: Select the database to import data to from the Database menu.

Username: Enter a user name into the appropriate field.

Password: Enter corresponding password into the appropriate field.



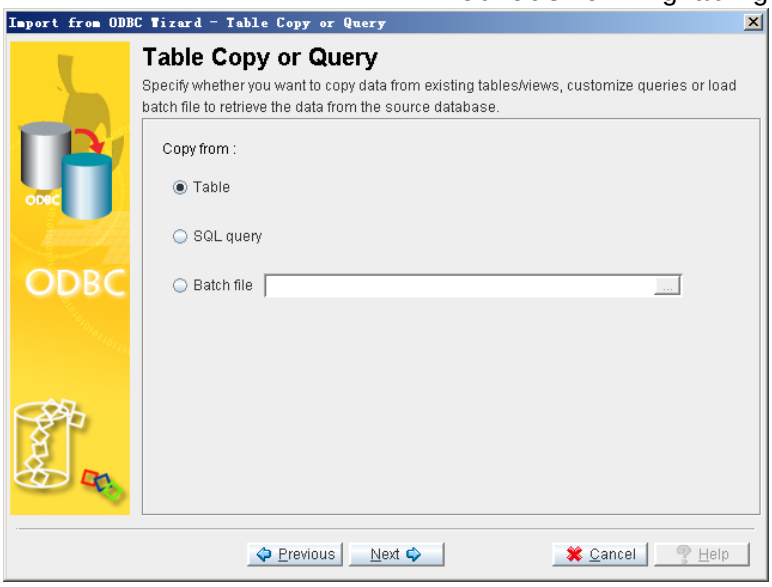
Step 4: Click the **Next** button. The **Table Copy or Query** window appears.

There are three options provided, select one of the three methods for data transfer:

Table: To import data from a list of tables,

SQL query: To import data using a series of SQL SELECT statements

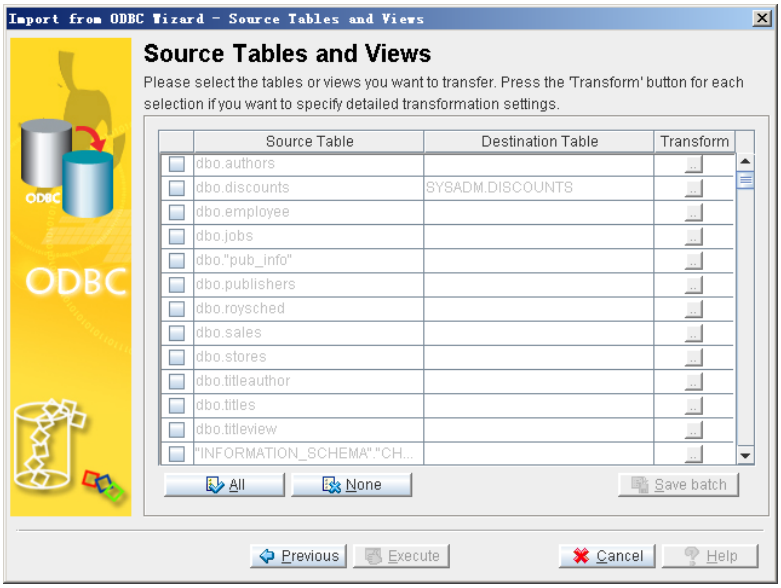
Batch file: To import data through an XML file.



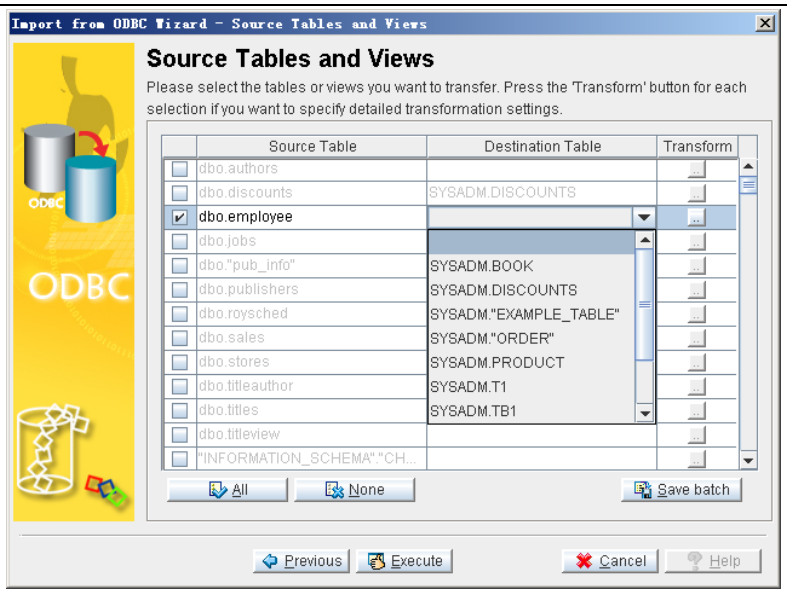
Three choices and corresponding operations:

1. Selected **Table** check box.

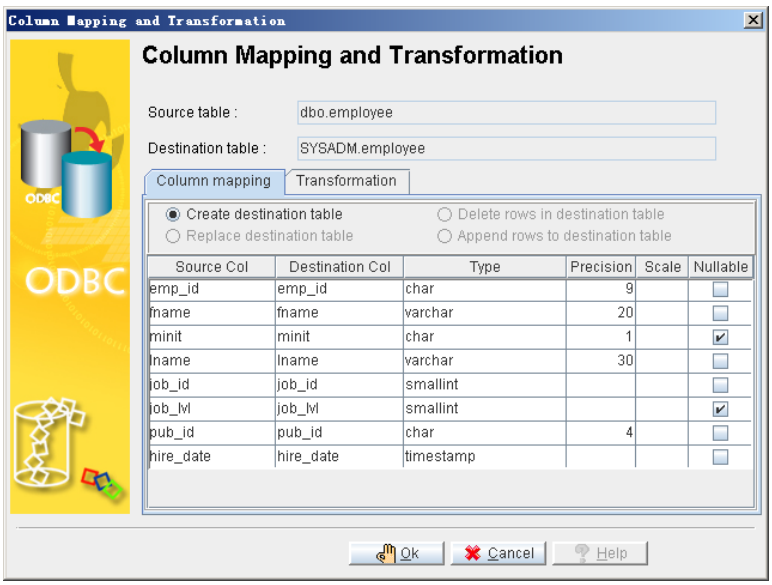
Sub_step 1: Click **Next**. The **Source Tables and Views** window appears. All tables from the source database will appear in the Source Table column. Check the box to the left of each table to import. You may also choose to save the map of the import ODBC schema to an XML file by clicking **save batch**. The **Save Batch File** will open. 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.



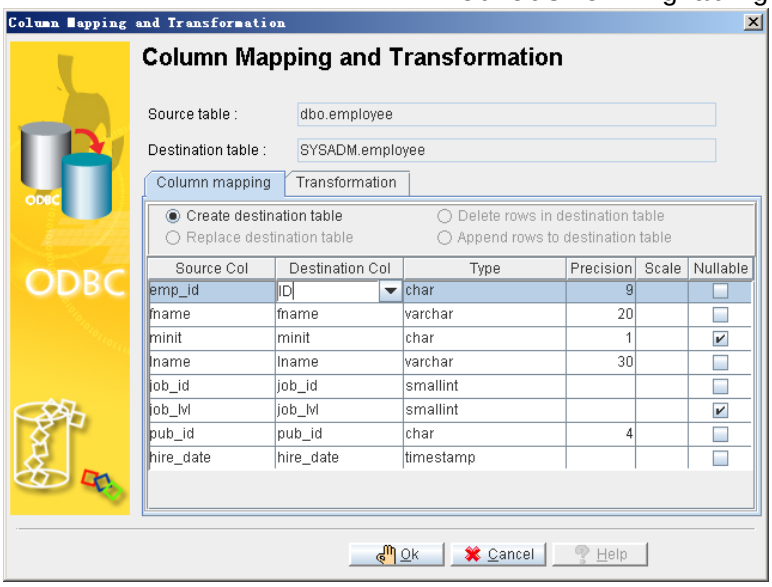
Sub_step 2: For each source table or view selected, click the **Destination Table** field. If desired, change the name of the destination table by selecting a new table from the menu or entering a new name.



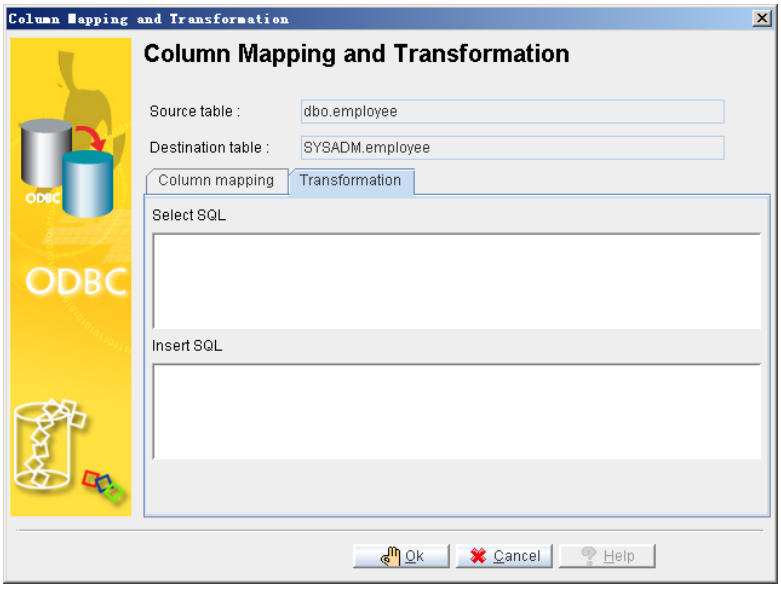
Sub_step 3: You may modify column mapping or the result set to import by clicking the **Transform** button of the corresponding source and destination table.



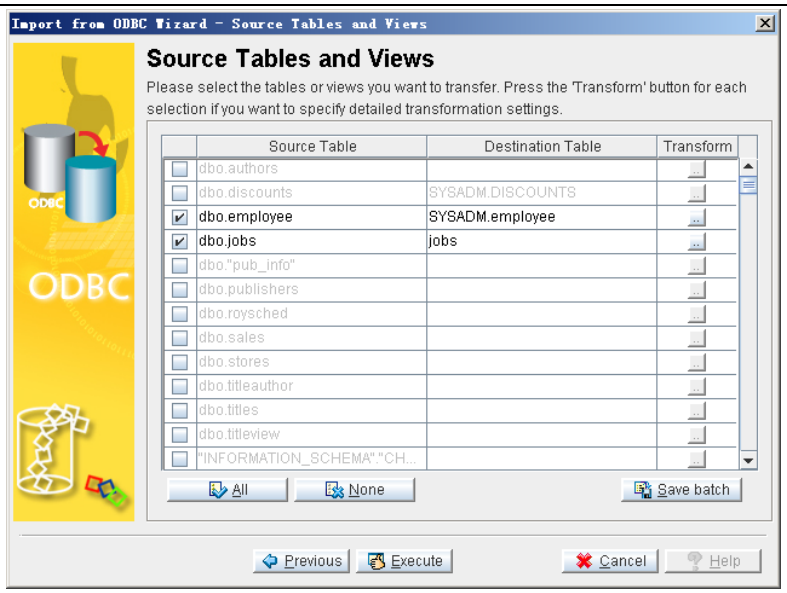
Sub_step 4: Change the name of the destination column by selecting a new column from the menu or entering a new name.



Sub_step 5: 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.

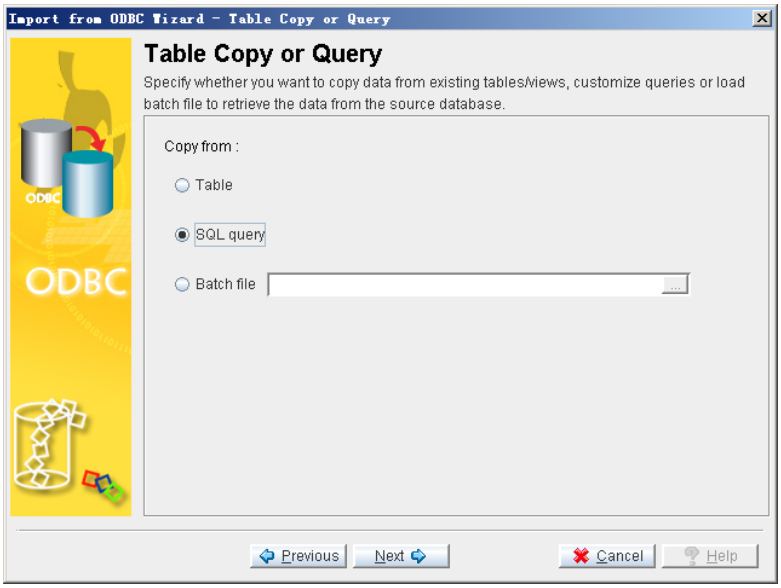


Sub_step 6: Click **OK** to return to the **Source Tables and Views** window.

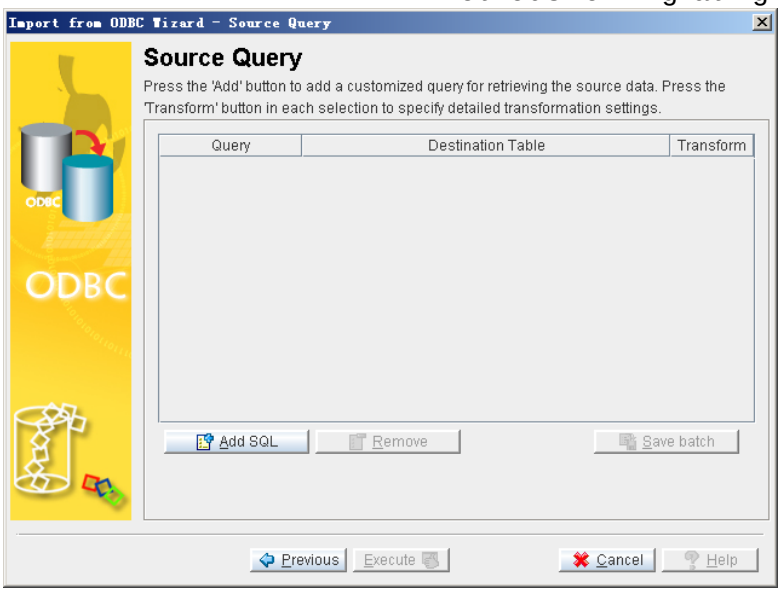


2. Select **SQL query** check box.

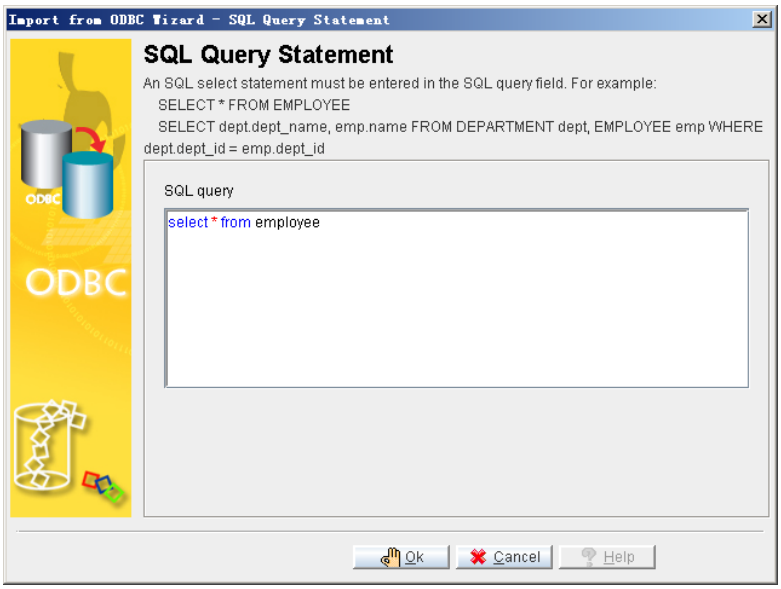
Sub_step 1: Click **Next**. The **Source Query** window appears. Click **Add SQL**. The **SQL Query Statement** window appears. And enter a valid SQL SELECT statement into the **SQL Query** field.



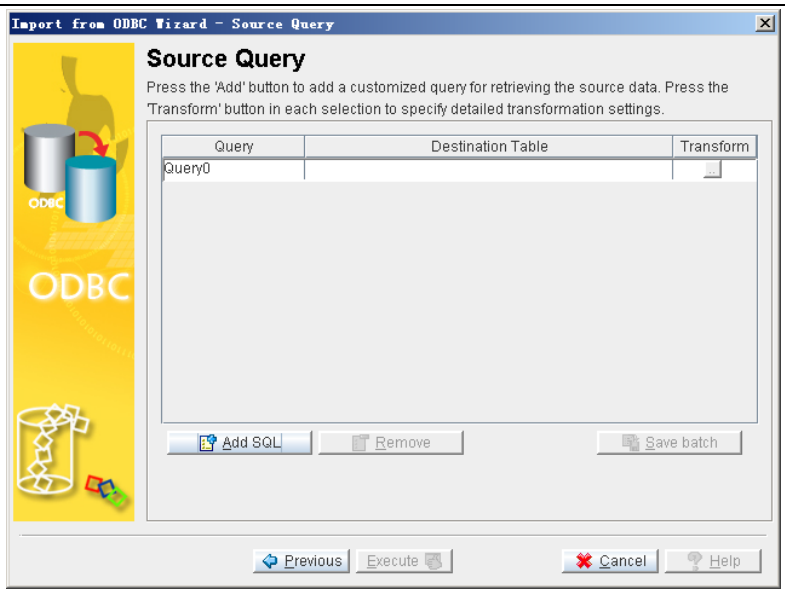
Sub_step 2: Click **OK**. The **Source Query** window reappears.



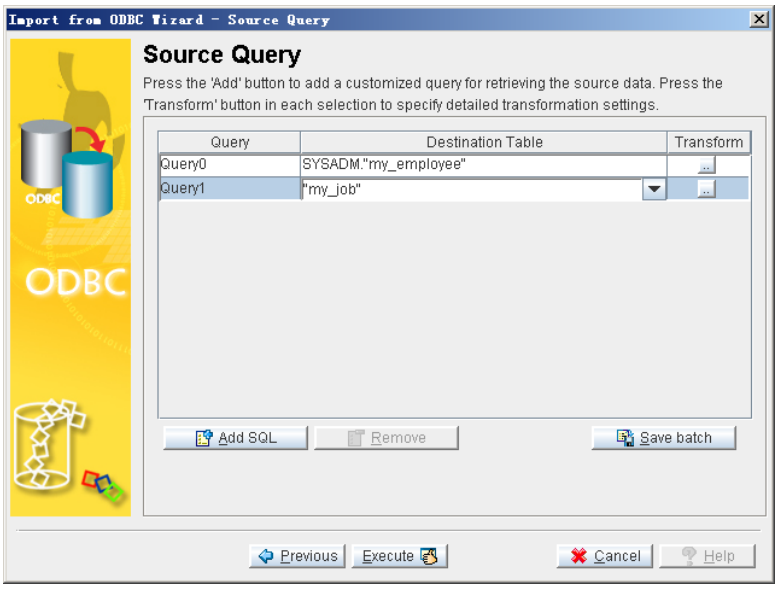
Sub_step 3: Click **Add SQL**. The **SQL Query Statement** window appears. Enter a valid SQL SELECT statement into the **SQL Query** field.



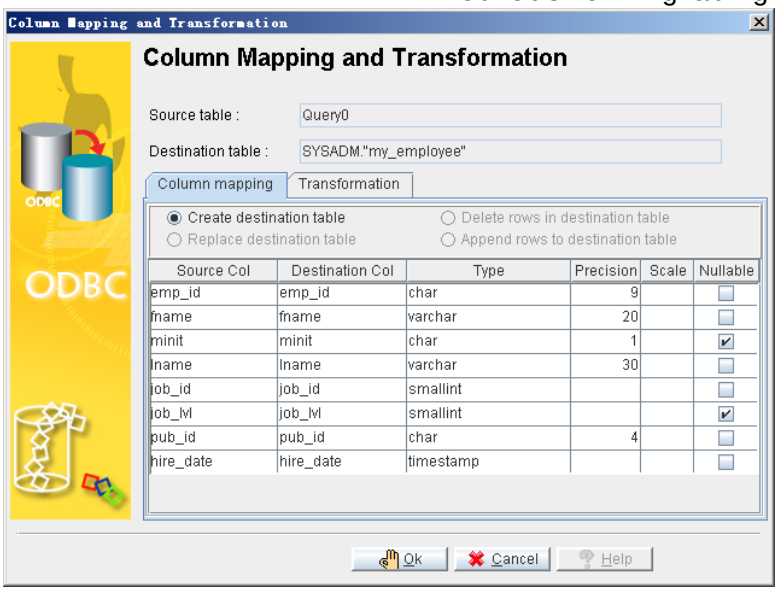
Sub_step 4: click **OK** to return to the **Source Query** page.



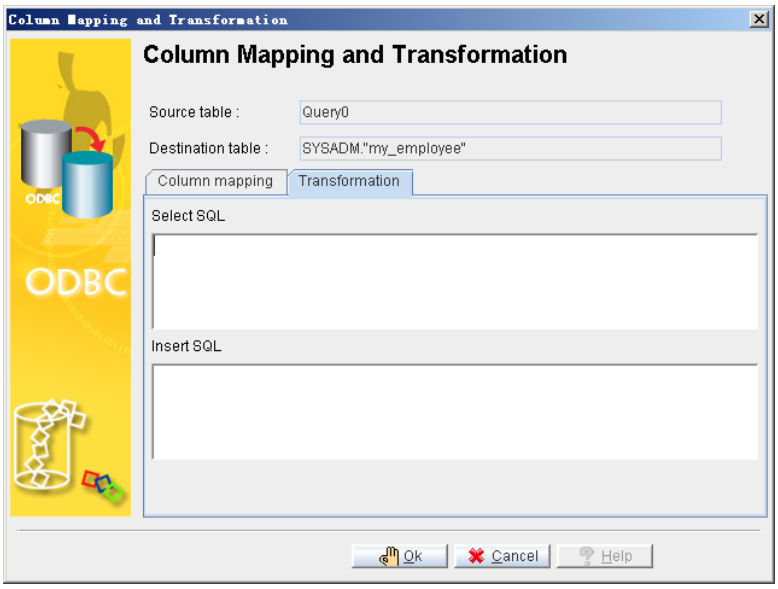
Sub_step 5: You may add more SQL query statements by clicking **Add SQL** and change the name of the destination column by selecting a new column from the menu or entering a new name.



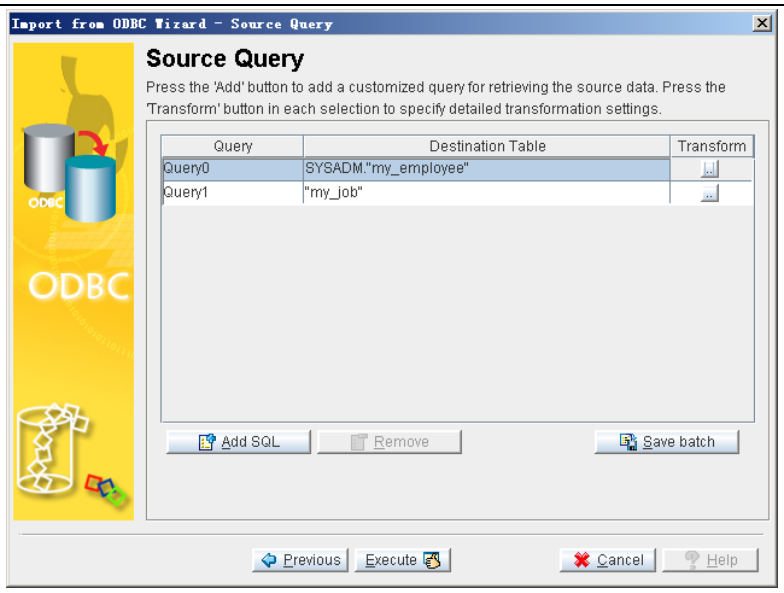
Sub_step 6: you also can modify the mapping of source and destination columns by clicking the **Transform** button.



Sub_step 7: 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.



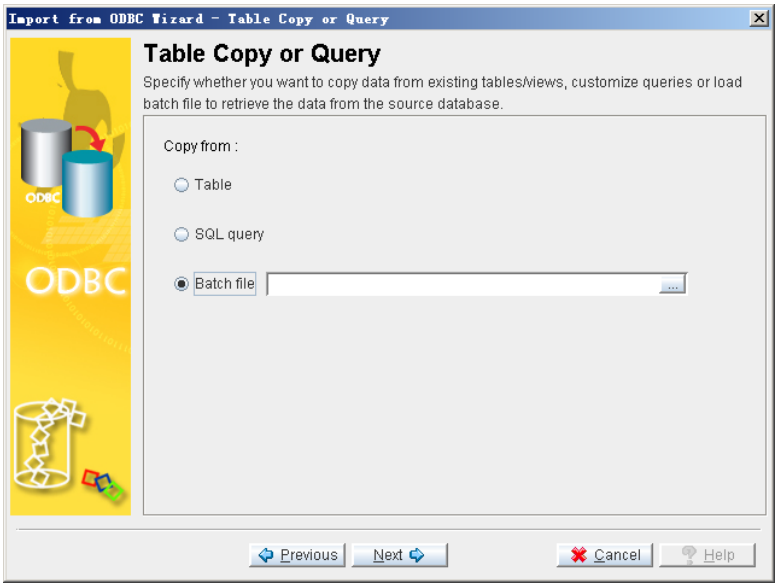
Sub_step 8: click **OK** to return to the **Source Query** page.



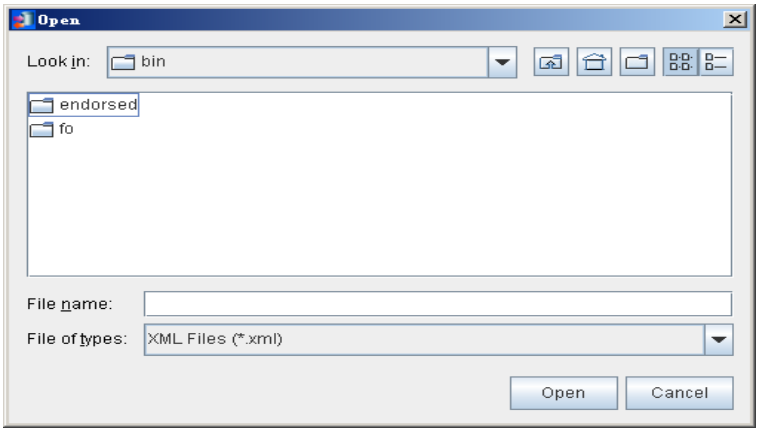
Note: The remaining operations are same as selecting **Table**.

3. Select **Batch file** Check box

Sub_step 1: Select an XML file from which to import the ODBC map schema. Click **Open**. The **Table Copy or Query** window reappears.

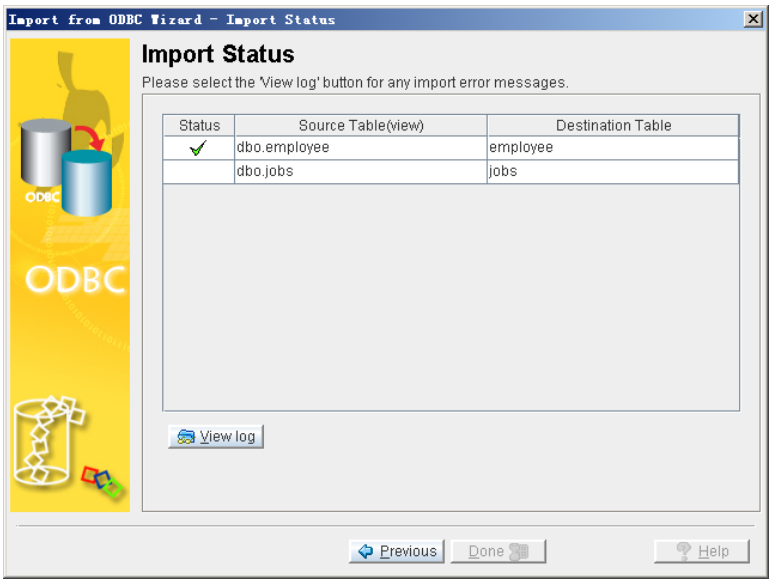


Sub_step 2: Click **Next**. The **Source Query** window will open, displaying a mapping Schema according to the XML file.

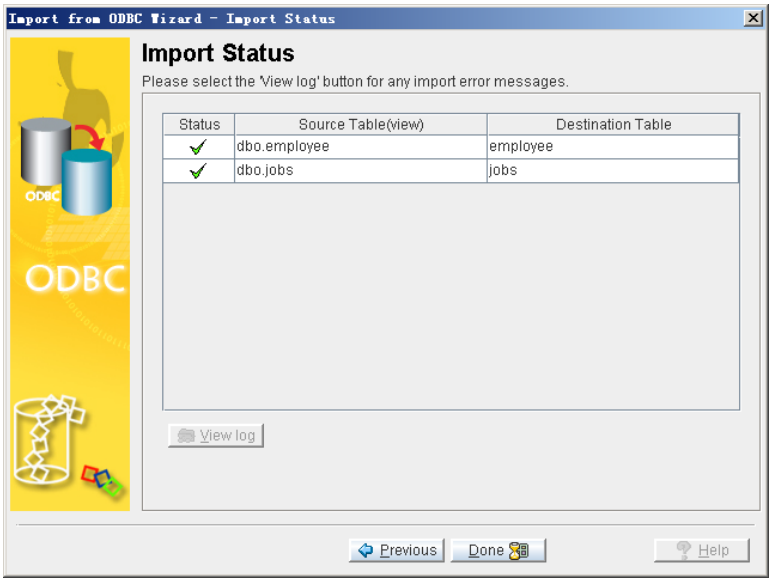


NOTE: The remaining operations are same as select **Table**.

Step 5: Click **Execute** to import the source data. The **Import Status** window appears.



Step 6: If errors appear, click **View log** and scroll to the bottom to see the error message. If no errors occurred, click **Done**.



4.2 Other 3rd party tools

Currently, there are many kinds of database migration tools can be used. Some of them are popular which can be used for most Databases. Certainly, some of them are only designed for special databases.

The user can choose a popular tool for their migration according to different requests.

In following chapters, we will introduce two popular database migration tools.

4.2.1 SQL Script Builder

SQL Script Builder is a powerful software by which users can create a database migration sql script (or dump file) or database files from any ODBC data source. The script will migrate the

database (multiple tables selection) or only one table and records. Scripts are available in five output formats; MySQL, MS SQL, Oracle, Pervasive and PostgreSQL, and files come in Access mdb, Excel csv, MS xml. SQL Script Builder is very simple to use, you just have to choose the database and tables from the list. SQL Script Builder scripts can be used on your DBMS (database management system) or uploaded on a server.

SQL Script Builder can be used. For example, if you migrate a database from SQL Server to DBMaster, you don't have to transfer whole database, you can import only one table at a time and have no limit, what you need is the ODBC driver for the database you wish to import from. ODBC is a universal interface almost every database provider supports it.

With SQL Script Builder, you can create an ODBC connection for origination database and generate the script, then, you need to ensure the script can work well on the destination database.

- Operation Steps:

Step 1: Use this tool to convert the data from origination database to a supported file format. For DBMaster, we recommend the XML format.

Step 2: Use JDataTransfer tool in DBMaster and select **Import from XML** option to import datafiles that have been exported from SQL Server to DBMaster.

4.2.2 SqlToTxt Tool

SqlToTxt is a tool by which users can export SQL Server data to flat files. It is useful and the operations are simple.

- Main features:

1. Support multiple formats files (txt, csv, xml, html, sql, Excel) for exporting data from SQL Server.

2. Support exporting from a single table, all tables or SQL query.

3. Support exporting SQL Server text (ntext) and image fields.

- Operation Steps:

Step 1: Use this tool to convert the data from origination database to a supported file format. For DBMaster, we recommend the XML format.

Step 2: Open JDataTransfer tool in DBMaster, select **Import from XML** or **Import from Text** option to import datafiles which have been exported from SQL Server to DBMaster.

4.3 Modify DDL manually

If you are familiar with DDL, you can create the entire schema from SQL Server firstly. And modify the schema and make syntax and data types fit to DBMaster. For example: Via **Management Studio** (Right-click the Database, then choose **Task/Generate Scripts**).

Then, run the schema script in dmSQL tool or JSQL tool. Please modify and try again if any errors happen.

At last, export the data from SQL Server (Right-click the Database, then choose **Task/Export Data**). You also can via **Import and Export Data** (Refer to *Chapter 4.1.1* and select the Flat File source for "**Choose a Data Source**" wizard). If exporting data is successful, please import the data into DBMaster via JdataTransfer tool.

Note: If you don't want to use JdataTransfer to import, after exporting the data to Flat format files, you may use the ***import*** command. Certainly, you may modify them for DBMaster and run in dmSQL or JSQL tool.

4.4 Write code

Users can use a programming language they are familiar with to develop a simple script or tool for migrating database. The work theory and process steps are similar as above Manual Methods.

5. Compare SQL Server and DBMaster

5.1 Schema Comparison

5.1.1 THE TERMINOLOGY COMPARISON

The following table enlists the terminologies in DBMaster and SQL Server. In many aspects, DBMaster have such common characteristics with SQL Server. However, there are some differences between them.

SQL Server	DBMaster
Database	Database
File Group	Tablespace
Block	Page/Frame
Login	User
Role	Group
Table	Table
View	View
Temporary Table (in the tempdb database)	Temporary Table (in .db file,)
Cluster	N/A
Check constraint	Check constraint
Sequences	Serial
Synonyms (after SQL Server2005 version)	Synonyms
DML Triggers	DML Triggers
DDL Triggers	N/A
Column default	Column default
Unique index	Unique index
Primary key	Primary key

Foreign key	Foreign key
Index	Non-unique index
Transact-SQL (T-SQL) stored procedure	Embedded-SQL (ESQL/C) stored procedures, Java stored procedures and SQL stored procedures (DBMaster supported after 5.2 version)
Rule	N/A
Default	N/A
UDF	UDF
User define data type	Domain
UNIQUE	N/A

5.1.2 STORAGE STRUCTURE COMPARISON

Basically, DBMaster has a great difference from SQL Server. Users should recognize these differences with discretion.

Item		SQL Server	DBMaster
Interface or tools to configure parameters		Property interface of DB	JConfiguration tool
Temporary Table		Stored in tempdb database	Temporarily in .db file
File type	Data Files	*.MDF or *.NDF	*.SDB or *. DB
	Journal File	*.LDF	*.JNL
	BLOB File	*.MDF or *.NDF	*.SBB or *.BB

Note that SQL Server allocates BLOB data and ordinary data within the same page. Despite it contains a pointer which indicates the real location of BLOB data. It isn't a good design for handling the BLOB data. On the other hand, DBMaster allocates the BLOB data separately from ordinary data. It would bring up some performance benefits.

5.1.3 PROCESS AND RELATED TERM DEFINITION

In SQL Server, every task would be taken care of by different processes. DBMaster, on the other hand, will use the same method but look like a general process instead of individual processes.

Item	SQL Server	DBMaster
Start-up mode	Single server supports multiple databases	Single server supports one database
Management functionality	System Monitor	DBMaster Server
	Backup Device	Backup Server

	Roll forward	Replay Journal
	Rollback	Rollback Journal
	Recovery	Recovery
	Checkpoint	Checkpoint
	Backup	Backup
	Restore	Restore
	Import	Import/Load
	Export	Export/Unload

5.1.4 RESERVED WORD CONFLICT IN DATABASE OBJECT

SQL Server and DBMaster reserved words are different. Many DBMaster reserved words are valid object names or column names in SQL Server. Likewise, many SQL Server reserved words are valid object names in DBMaster. Using reserved words as database object names makes it impossible to use the same names across the two databases.

Choose a unique database object name by case and by at least one other characteristic, and ensure that the object name is not a reserved word from either database.

Costumers can write object names in double quotation marks in DBMaster or brackets in SQL Server if you want to use reserved words as object names.

For example,

In DBMaster: create table test ("ADD" int);

In SQL Server: create table test ([ADD] char (5))

Different from SQL Server, in DBMaster, we also can set keyword **DB_ResWd** to be 0 in **dmconfig.ini** file before database creation, which allows objects containing reserved words to be imported.

SQL Server	DBMaster
ADD, EXCEPT, PERCENT, ALL, EXEC, PLAN, ALTER, EXECUTE, PRECISIONAND, EXISTS, PRIMARY, ANY, EXIT, PRINT, AS, FETCH, PROC, ASC, FILE, PROCEDURE, AUTHORIZATION, FILLFACTOR, PUBLICBACKUP, FOR, RAISERROR, BEGINFOREIGN, READ, BETWEEN, FREETEXT, READTEXT, BREAK, WRITETEXTFREETEXTTABLE, RECON-, FIGURE, BROWSE, REFERENCES, FROM, BULKFULL, REPLICATION, BY, FUNCTIONRESTORE, CASCADE, GOTO, RESTICTCASE, GRANT, CHACK, GROUP, REVOKECHACKPOINT, IF, HABING, RIGHT, CLOSEHOLDLOCK, ROLLBACK, CLUSTERED, IDENTITY, ROWCOUNT, COALESCE, IDENTITY_INSERT, ROWGUIDCOL, COLLATE, IDENTITYCOL, RULE, COLUMN, SAVE, COMMIT, IN, CHEMA, COMPUTEINDEX, SELECT, CONSTRAINT, INNER, SESSION_USER, CONTAINS, INSERT, SETINTO, CONTAINSTABLE, INTERSECTSETUSER, CONTICUE, SHUTDOWN, CONVERT, IS, SOME, CREATE, JOIN, STATISTICS, KEY, CROSS, SYSTEM_USERCURRENT, KILL, TABLE, CURRENT_DATE, LEFT, TEXTSIZE , CURRENT_TIME, LIKE, THEN , CURRENT_TIMESTAMP, LINENO, TO , CURRENT_USER, LOAD, TOP, CURSOR NATIONAL, TRAN, DATABASE, NOCHECK TRANSACTION, DBCC, NONCLUSTERED TRIGGER, ERRLVL DEALLOCATE, NOT, DROP, OUTER TRUNCATE , DECLARE, NULL, TSEQUAL DEFAULT, NULLIF, UNION , DELETE, OF UNIQUE, DENY, OFF, UPDATE, DESC OFFSETS, UPDATETEXT, DISK, ON, USE DISTINCT, OPEN, USER, DISTRIBUTED OPENDATASOURCE, VALUES, DOUBLE OPENQUERY, VARYING, WITH, WHILE ESCAPE, PENROWSET, VIEW, DUMMY, OPENXML WAITFOR, DUMP, OPTION, WHEN , OVER,ELSE, OR, WHERE, ORDER END	ABSOLUTE, ACTION, ADD, ADMIN, AFTER, AGGREGATE, ALIAS, ALLOCATE, ALTER, AND, ANY, ARE, ARRAY, AS, ASC, ASSERTION, AT, AUTHORIZATION, BEFORE, BEGIN, BINARY, BIT, BLOB, BOOLEAN, BOTH, BREADTH, BY, CALL, CASCADE, CASCADED, CASE, CAST, CATALOG, CHECK, CLASS, CLOB, CLOSE, COLLATE, COLLATION, COLUMN, COMMIT, COMPLETION, CONNECT, CONNECTION, CONSTRAINT, CONSTRAINTS, CONSTRUCTOR, CONTINUE, CORRESPONDING, CREATE, CROSS, CUBE, CURRENT, CURRENT_DATE, CURRENT_PATH, CURRENT_ROLE, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_USER, CURSOR, CYCLE, DATE, DAY, DEALLOCATE, DEC, DECIMAL, DECLARE, DEFAULT, DEFERRABLE, DEFERRED, DELETE, DEPTH, DEREF, DESC, DESCRIBE, DESCRIPTOR, DESTROY, DESTRUCTOR, DETERMINISTIC, DICTIONARY, DIAGNOSTICS, DISCONNECT, DISTINCT, DOMAIN, DOUBLE, DROP, DYNAMIC, EACH, ELSE, END, END-EXEC, EQUALS, ESCAPE, EVERY, EXCEPT, EXCEPTION, EXEC, EXECUTE, EXTERNAL, FALSE, FETCH, FIRST, FLOAT, FOR, FOREIGN, FOUND, FROM, FREE, FULL, FUNCTION, GENERAL, GET, GLOBAL, GO, GOTO, GRANT, GROUP, GROUPING, HAVING, HOST, IDENTITY, IGNORE, IMMEDIATE, IN, INDICATOR, INITIALIZE, INITIALLY, INNER, INOUT, INPUT, INT, INTEGER, INTERSECT, INTO, IS, ISOLATION, ITERATE, JOIN, KEY, LANGUAGE, LARGE, LAST, LATERAL, LEADING, LESS, LEVEL, LIKE, LIMIT, LOCAL, LOCALTIME, LOCALTIMESTAMP, LOCATOR, MAP, MATCH, MODIFIES, MODIFY, MODULE, NAMES, NATIONAL, NATURAL, NCHAR, NCLOB, NEXT, NO, NONE, NOT, NULL, NUMERIC, OBJECT, OF, OFF, ON, ONLY, OPEN, OPERATION, OPTION, OR, ORDINALITY, OUT, OUTER, OUTPUT, PAD, PARTIAL, PATH, POSTFIX, PREFIX, PREORDER, PREPARE, PRESERVE, PRIMARY, PRIOR, PRIVILEGES, PROCEDURE, READ, READS, REAL, RECURSIVE, REFERENCES, REFERENCING, RELATIVE, RESTRICT, RESULT, RETURN, RETURNS, REVOKE, ROLE, ROLLBACK, ROLLUP, ROUTINE, ROW, ROWS, SAVEPOINT, SCHEMA, SCROLL, SCOPE, SEARCH, SECTION, SELECT, SEQUENCE, SESSION, SESSION_USER, SET, SETS, SIZE, SMALLINT, SOME, SPECIFIC, SPECIFICTYPE, SQL, SQLEXCEPTION, SQLSTATE, SQLWARNING, START, STATIC, STRUCTURE, SYSTEM_USER, TABLE, TEMPORARY, TERMINATE, THAN, THEN, TIME, TIMESTAMP, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRANSLATION, TREAT, TRIGGER, TRUE, UNDER, UNION, UNKNOWN, UNNEST, UPDATE, USAGE, USING,VALUES, VARCHAR, VARIABLE, VARYING, VIEW, WHEN, WHENEVER, WHERE, WITH, WITHOUT, WORK, WRITE, ZONE

5.1.5 DATABASE OBJECT DESIGN CONCERNS

For Database Objects, or Schema Objects, users need to put many factors into account before migration. It will contain a variety of constraints checking, data types mapping and so on. We enlist the factors as followings.

- Data Types
- Entity Integrity Constraints
- Referential Integrity Constraints
- Unique Key Constraints
- Check Constraints
- Support Platforms

5.1.5.1 Entity Integrity Constraints

A primary key can be defined as part of a CREATE TABLE or an ALTER TABLE statement. SQL Server internally creates a unique index to enforce the integrity.

So does DBMaster, a primary key constraint will be applied to a unique index internally. The performance will be promoted for it's an index too. The constraint will be kept to retain integrity.

SQL server	DBMaster
CREATE TABLE table_name(Column1 datatype PRIMARY KEY, Column2 datatype, …); CREATE TABLE table_name(Column1 datatype, Column2 datatype, …, CONSTRAINT pk_name PRIMARY KEY (Column1, Column2,…));	CREATE TABLE table_name(Column1 datatype PRIMARY KEY, Column2 datatype, …); CREATE TABLE table_name(Column1 datatype, Column2 datatype, …, PRIMARY KEY (Column1, Column2,…));
ALTER TABLE table_name ADD PRIMARY KEY (column_name) ALTER TABLE Table_name ADD Constraint pk_name PRIMARY KEY [CLUSTERED] (Column1, Column2,…);	ALTER TABLE table_name PRIMARY KEY (Column1, column2,…); ALTER TABLE table_name ADD CONSTRAINT pk_name PRIMARY KEY(Column1,column2,…);

5.1.5.2 Referential Integrity Constraints

SQL Server provides declarative referential integrity. A CEATE TABLE or ALTER TABLE statement can add foreign keys to the table definition. You can also define a foreign key for a table in DBMaster. Foreign keys can be defined in a CREATE TABLE statement or an ALTER TABLE statement.

DBMaster and SQL Server have many similarities in the term of Integrity Constraints. It makes the migration process less labor.

SQL server	DBMaster
CREATE TABLE table_name1	CREATE TABLE table_name1(

(Column1 datatype NOT NULL PRIMARY KEY, Column2 datatype NOT NULL, Column3 datatype FOREIGN KEY REFERENCES table_name2(column_name)) CREATE TABLE table_name (Column1 datatype NOT NULL, Column2 datatype NOT NULL, Column1 datatype,… PRIMARY KEY (column_nameF), CONSTRAINT fk_name FOREIGN KEY (column_nameP) REFERENCES Persons(column_nameP))	Column1 datatype, Column2 datatype, …, FOREIGN KEY fk_name(column1,…) REFERENCES table_name2); CREATE TABLE table_name(Column1 datatype, Column2 datatype, …, Column datatype CONSTRAINT fk_name REFERENCES table_name2(column_name));
ALTER TABLE table_name ADD FOREIGN KEY (column_name) REFERENCES Persons(column_name) ALTER TABLE table_name1 ADD CONSTRAINT fk_name FOREIGN KEY (column_name) REFERENCES table_name2(column_name)	ALTER TABLE table_name ADD FOREIGN KEY (column_name) REFERENCES Persons(column_name) ALTER TABLE table_name1 ADD CONSTRAINT fk_name FOREIGN KEY (column_name) REFERENCES table_name2(column_name) ALTER TABLE tb_name1 FOREIGN KEY(column1,column2,…) REFERENCES table_name2;

5.1.5.3 Unique index Constraints

SQL Server defines unique index as part of CREATE TABLE or ALTER TABLE statements. SQL Server internally creates unique indexes to enforce these constraints.

You can also define a unique index for a table in DBMaster. Unique indexes can be defined in a CREATE TABLE statement or an ALTER TABLE statement. DBMaster owns the same function and can build unique index to confirm uniqueness of every record.

SQL server	DBMaster
CREATE TABLE table_name1 (Column1 datatype NOT NULL **UNIQUE**, Column2 datatype, ,…) CREATE TABLE table_name1 (Column1 datatype NOT NULL, Column2 datatype, ,…, CONSTRAINT uc_name UNIQUE (colunm1, column2,…))	CREATE TABLE table_name1 (Column1 datatype NOT NULL **UNIQUE**, Column2 datatype, ,…) CREATE TABLE table_name1 (Column1 datatype NOT NULL, Column2 datatype, ,…, CONSTRAINT uc_name UNIQUE (colunm1, column2,…)) CREATE TABLE table_name1 (Column1 datatype CONSTRAINT u **UNIQUE**, Column2 datatype, ,…)
ALTER TABLE table_name ADD **UNIQUE** (column_name) ALTER TABLE table_name ADD CONSTRAINT uc_name **UNIQUE** (column1,column2,…)	ALTER TABLE table_name ADD **UNIQUE** (column_name) ALTER TABLE table_name ADD CONSTRAINT uc_name **UNIQUE** (column1,column2,…)

5.1.5.4 Check Constraints

SQL Server defines check constraints as part of the CREATE TABLE statement or the ALTER TABLE statement. A check constraint is defined at the TABLE level and the COLUMN level. A table-level check constraint can refer to any column in the constrained table. A column can have only one check constraint. A column-level check constraint can refer to only the constrained column.

Check constraints can be defined in a CREATE TABLE statement or an ALTER TABLE statement in DBMaster as well. Multiple check constraints can be defined on a table.

Table-level check constraints from SQL Server databases map one-to-one with DBMaster check constraints. Since DBMaster has the column-level check, migration from SQL Server to DBMaster will have not lost the check constraints or sort of things.

SQL server	DBMaster
CREATE TABLE table_name(Column1 datatype CHECK boolean_expression, Column2 datatype CHECK check_expression, …); CREATE TABLE table_name(Column1 datatype , Column2 datatype,… CONSTRAINT ck_name CHECK (check_expression1 AND check_expression2 AND …));	CREATE TABLE table_name(Column1 datatype CHECK boolean_expression, Column2 datatype CHECK boolean_expression, …); CREATE TABLE table_name(Column1 datatype, Column2 datatype, [CONSTRAINT ck_name] CHECK(boolean_expression1 AND boolean_expression2 AND …) …);
ALTER TABLE table_name ADD CHECK (check_expression) ALTER TABLE table_name ADD CONSTRAINT ck_name CHECK (check_expression1 AND check_expression2,…)	ALTER TABLE table_name MODIFY (column1 to column1 datatype CHECK column1 boolean_expression,…);

5.2 Data Types Mapping

This section provides detailed descriptions of the differences in data types used by SQL server and DBMaster databases. Specifically, this section contains the following information:

5.2.1 COMMON DATA TYPE MAPPING

A table showing the base and available SQL Server data type and how they are mapped to DBMaster data types.

Recommendations based on the information are listed in the table:

SQL server	Description	DBMaster	Comments
Integer types		Integer types	
BIGINT	The BIGINT data type bytes storage with the range of -2^63 to 2^63-1.	INTEGER (4 byte)	Some huge number will lose its precision(DBMaster supported Bigint type after 5.2 version)
SMALLINT	Two-byte integer, 15 bits, and a sign. (-2^15 – 2^15-1)	SMALLINT (2 byte)	Two-byte integer, 15 bits, and a sign. (-2^15 – 2^15-1)
INT	The INT data type uses 4 bytes of storage with the range of -2,147,483,648 to 2,147,483,647.	INTEGER (4 byte)	The INTEGER data type uses 4 bytes of storage with the range of -2,147,483,648 to 2,147,483,647.

TINYINT	The TINYINT data type uses 1 bytes of storage with the range of 0-255	SMALLINT (2 byte)	The SMALLINT data type uses 2 bytes of storage with the range of -32,768 to 32,767
Floating point data types		Floating point data types	
REAL	The REAL data use 4 bytes and has a valid input range of -3.40E+38 to -1.18E -38、0 and 1.18 E -38 to 3.40E+38	FLOAT(REAL)（DB_FltDb=0）(4 byte)	The FLOAT data type uses 4 bytes of storage and has a valid input range of 3.402823466E38 to –3.402823466E38. The smallest valid input values are1.175494351E-38 and –1.175494351E-38.
FLOAT	The FLOAT data has a valid input range of -1.79E+308 to 1.79E+308.	DOUBLE(DOUBLE) (DB_FltDb = 1) (8 byte)	The DOUBLE data type uses 8 bytes of storage and has a valid input range of 1.0E308 to –1.0E308.
DECIMAL(p,s)	The DECIMAL data type bytes storage with the range of **-** 10^38 +1 to 10^38 –1 with fixed precision.	DECIMAL(p,s)	The default value for precision is 17 with a maximum value of = 38. Scale refers to the number of digits to the right of the decimal point. The default value for scale is 6.
Binary digit data types		Binary digit data types	
BINARY	The BINARY data type is a fixed-length data type that can contain any binary value. The length of RAW columns is between 1 byte and 8000 bytes.	BINARY(1-3992) (n byte)	The minimum length of BINARY columns is 1 byte and the maximum length is 3992 bytes.
VARBINARY(1-8000)	The VARBINARY data type is a variable-length data type that can contain any binary value. The length of RAW columns is between 1 byte and 8000 bytes	LONG VARBINARY	DBMaster cannot store any string whose length exceeds 4K into BINARARY or VAR BINARARY column. Therefore, VARBINARY in SQL Server is preferable to map into LONG VARBINARAY of DBMaster.
VARBINARY(max)	The store max value =2^31-1 Storage size for the actual length of the input data + 2 bytes.the length can be zero bytes.	LONG VARBINARY	DBMaster cannot store any string whose length exceeds 4K into BINARARY or VAR BINARARY column. Therefore, VARBINARY in SQL Server is preferable to map into LONG VARBINARAY of DBMaster. The maximum length of LONG VARBINARY columns is 8 TB.
Logical digit data types		Logical digit data types	
BIT Character types	The BIT data type bytes storage with the 1 or 0	N/A	User should use SMALLINT instead
Character data types		Character data types	
CHAR (1-8000)	The CHAR data type is a fixed-length data type that can contain any character from the	CHAR (n byte) 3968 (4KB page size)	In DBMaster, CHAR columns length can be Depending on DB_PGSIZ (4k, 8k, 16k, and 32k) (NO Unicode).

	keyboard. The CHAR length in SQL server is between 1 and 8000 bytes (NO Unicode).	size) 8064 (8KB page size) 16256 (16KB page size) 32640 (32KB page size)	
NCHAR	The NCHAR data type is a fixed-length data type that can contain any character from the keyboard. The CHAR length in SQL server is between 1 and 4000 bytes (Unicode).	NCHAR (n byte) 1984 (4KB page size) 4032 (8KB page size) 8128 (16KB page size) 16320 (32KB page size)	The NCHAR data type is a fixed-length data type that can contain any Unicode character. NCHAR columns length can be Depending on DB_PGSIZ (4k, 8k, 16k, and 32k) ((Unicode).
VARCHAR (1-8000)	The VARCHAR data type is a variable-length data type that can contain any character that can be entered from the keyboard. In SQL server, the length is between 1 and 8000 bytes (NO Unicode).	VARCHAR 3968 (4KB page size) 8064 (8KB page size) 16256 (16KB page size) 32640 (32KB page size)	VARCHAR columns length can be Depending on DB_PGSIZ (4k, 8k, 16k, and 32k) (NO Unicode).
VARCHAR(MAX)	The VARCHAR data type is a variable-length data type that can contain any character that can be entered from the keyboard. In SQL server, the length is between 1 and 2^31-1 bytes (NO Unicode).	LONG VARCHAR	SQL Server is preferable to map into LONG VARCHAR of DBMaster. The maximum length of LONG VARCHAR columns is 8 TB.
NVARCHAR NVARCHAR(MAX)	The NVARCHAR data type is a variable-length data type that can contain any character that can be entered from the keyboard. In SQL server, the length is between 1 and 4000 bytes (Unicode).MAX indicate storage maximum is 2^31-1 bytes	NVARCHAR 1984 (4KB page size) 4032 (8KB page size) 8128 (16KB page size) 16320 (32KB page size)	The NVARCHAR data type is a variable-length data type that can contain any Unicode character. NVARCHAR columns length can be Depending on DB_PGSIZ (4k, 8k, 16k, and 32k) (Unicode).

	bytes.		
Text and image data types		BLOB data types	
TEXT	The TEXT data type is a variable-length data type that can contain any character that can be entered from the keyboard. In SQL server, the length is between 1 and 2^31-1 bytes(NO Unicode).TEXT type will be instead of varchar(max) and TEXT type will be delete in future SQL Server version.	LONG VARCHAR	The maximum length of CLOB columns is 8T.
NTEXT	The NTEXT data type is a variable-length data type that can contain any character that can be entered from the keyboard. In SQL server, the length is between 1 and 2^30-1 bytes (Unicode). NTEXT type will be instead of nvarchar (max) and NTEXT type will be delete in future SQL Server version.	NCLOB	The NCLOB data type is a variable length data type that can contain any Unicode character. Each Unicode character occupies 2 bytes of storage. The maximum length for an NCLOB column is 8 TB.
IMAGE	The IMAGE data type is a variable-length data type that can contain any binary value. The length of RAW columns is between 0 byte and 2^31-1 bytes IMAGE type will be instead of VARBINARY(max) and IMAGE type will be delete in future SQL Server version.	LONG VARBINARY	Similar the BLOB
		File	DBMaster provides the SYSTEM FO and User FO.
Date and Time types		Date and Time types	
DATETIME	Stored date and time from 1753.1.1 to 9999.12.31	N/A	In DBMaster user should use TIMESTAMP instead but the precision of TIMESTAMP is one second. Migration from SQL Server to DBMaster would lose the precision of data in some cases.

SMALLDATETIME	Smalldatetime data type can only accurate to 1 minute and valid years range from 1900 to 2079.	N/A	In DBMaster User should use TIMESTAMP instead but the precision of TIMESTAMP is one second. Migration from SQL Server to DBMaster would lose the precision of data in some cases
DATE	DATE data type allow only store a date and valid years range from 0001 to 9999 only occupy 3 bytes	DATE	The DATE data type uses 4 bytes of storage that contains the calendar date (year, month, and day). Valid year range from 0001 to 9999.
TIME(x)	TIME date type use 24 hours format and time values can accurate to 100 ns. Only store time value without time, TIME date type support from 0 to 7 different precision, the same as DATETIME2 format. Its disk spending is 3 to 5 bytes depends on accuracy.	TIME	Stored the time (accurate to second).map SQL Server time(0) I.e. x =0.
DATETIME2(x)	DATETIME2 date type expanded the DATETIME data type from scope and date acceptable in the date/time value adding additional precision parts of time. Expand valid years range from 0001 to 9999. time to store a part only hours, minutes and seconds of value or it can support most of 7 decimal in different storage microseconds.DATETIME2(X) to specify precision and accuracy length is representative of x(from 0-7)	TIMESTAMP(11 bytes)	In DBMaster, the precision of TIMESTAMP is one second. Migration from SQL Server to DBMaster would lose the precision of data in some cases .But if x=0 i.e. .DATETIME2 (0) in SQL Server equal to TimeStamp in DBMaster.
DATETIMEOFFSET	DATETIMEOFFSET data type storage date and time (24 hours) which consistent with time zone. Time section can accurate to 100 ns. Time zone represents a [- | +] hh: mm.An effective timezone specified range from -14:00 to +14:00, this values add or subtract UTC can get local time.	N/A	

Money data types		Money data types	
MONEY	Monetary data values from -2^63 through 2^63 - 1	N/A	User should use DECIMAL or INT to be the replacement or create domain replacement
SMALLMONEY	Monetary data values from -214,748.3648 through +214,748.3647		User should use DECIMAL or INT to be the replacement or create domain replacement
New data types		New data types	
SQL_VARIANT	SQL Server SQL_VARIANT can hold up the data type that SQL_SERVER support (EXPECT for text, ntext, timestamp,varchar(max),nvarchar(max) sql_variant,hierarchyid user define type),	N/A	
HierarchyId		N/A	
UniqueIdentifier		N/A	
XML		N/A	
		BLOB(CLOB)	DBMaster BLOB only holds up to 2GB. CLOB field exceeds over 2GB.it corresponding to long varchar and long varbinary data type
		Media Types	Large object columns may also be specified as media types to aid in media process functions such as full text search for MicrosoftO WordO documents. The following media types are available: MsWordType, HtmlType, XmlType, MsWordFileType, HtmlFileType, and XmlFileType.
TABLE	A special data type used to store a result set for later processing	N/A	
Identify	Auto increase	SERIAL	Auto increase
Special types		Special types	
UNIQUEIDENTIFIER		OID (8 byte)	Fn,pn,sn 8 byte.

5.2.2 DATA TYPES MAPPING CONCERN

This section outlines conversion considerations for the Datetime and Image data type as examples to illustrate the factors you should consider:

- DATETIME Data Types
- IMAGE and TEXT Data Types (Binary/Character Large Objects)

5.2.2.1 DATETIME Data Types

There are some kinds of data types in SQL server, such as DATETIME, SMALLDATETIME. Note that there's a TIMESTAMP in SQL Server. But this TIMESTAMP is totally irrelevant with the Date/Time data. In SQL server, TIMESTAMP must be acquired from the DB system when INSERT

or UPDATE this specific row, and the column value is not the normal Date/Time. It is used to realize the version control. And it has nothing to do with the Date/Time data at all. Do not misuse TIMESTAMP in DBMaster. It is equivalent to DATETIME in SQL Server.

The date/time definition and its precision in SQL Server differ from the ones in DBMaster. For example, SQL Server has two DATETIME Data types (before SQL Server2008 version) --- Datetime and Smalldatetime data type. Datetime data storing date and time values can be accurate to 3/100 of a second and valid years range from 1753 to 9999, Smalldatetime data type can only be accurate to 1 minute and valid years range from 1900 to 2079.SQL Server2008 adds DATETIME2, DATE, TIME, DATETIMEOFFSET several data types, giving us more data types selection which can be used to store date and time data.

DATETIME2 data type expands the DATETIME data type from scope and date acceptable in the date/time value adding additional precision parts of time. Expanded valid years range from 0001 to 9999..The accuracy of the part time DATETIME2 depends on how you define DATETIME2 column, time to store a part only hours, minutes and seconds of value or it can support most of 7 decimal in different storage microseconds. DATETIME2(X) to specify precision and accuracy length is representative of x(from 0-7)

You can use the DATE data type only to store a DATE or TIME data type only to store a time value. In these new types of data section support accuracy if time can be reached 100ns.If there is a need to store the date and time of the SQL Server to maintain consistent, can use DATETIMEOFFSET data types, with these new data/time data types, you should be able to find good solutions to help you store your date using the correct format and support different date and time range when migration. Microseconds can be accurate to 100 ns. In addition, DATETIME2(3) format equals the use of SQL Server DATETIME old version format, but DATETIME2(3) can be used to support accuracy 1 millisecond and old version supports 3.33 milliseconds. If you want to store a date is accurate to seconds, you can use the DATETIME (0).

DATE data type allow only storing a date and valid years ranging from 0001 to 9999.If you only need to store the DATE and no use of time DATE data type, its disk spending is only 3 bytes, saving 1 byte than SMALLDATETIME old data type.

TIME date type use 24 hours format and time values can be accurate to 100 ns. It only store time value without date data. TIME date type supports from 0 to 7 different precisions, the same as DATETIME2 format. Its disk spending is 3 to 5 bytes depending on accuracy.

DATETIMEOFFSET data type storage date and time (24 hours) which are consistent with time zone. Time section can be accurate to 100 ns. The time zone consistency means time zone identifier is stored in DATETIMEOFFSET list. Timezone represents a [- | +] hh: mm. An effective timezone specified ranges from -14:00 to +14:00, then this values add or subtract UTC to get local time.

But DATE data type only stores the date values in DBMaster, and DATE data type can contain valid years ranging from 0001 to 9999. TIMESTAMP has a precision of 1 second, and TIMESTAMP stores the date and time values and the valid years ranging from 0001 to 9999 in DBMaster. According to the description here, Migration from SQL Server to DBMaster would lose the precision of data in some cases. But new data types added in SQL Server2008 provide more type selections for us to choose and more valid years range for date migration with different data structures. For instance, DATETIME2 and DATE TYPE in SQL Server has same range with DATA and TIMESTAMP data type in DBMaster.

Example:

SQL Server:

```
CREATE TABLE example_table
```

```
(date_column DATE not null,
Datetime_column, DATETIME2 (0) not null
text_column bigint not null,
varchar_column varchar(10) not null)
DBMaster:
CREATE TABLE example_table
(date_column DATE not null,
Datetime_colunm TIMESTAMP not null,
text_column long varchar not null,(supported bigint after DBMaster5.2 version)
varchar_column varchar(10) not null)
```

5.2.2.2 BLOB/CLOB Data Types (IMAGE and TEXT Data Types)

The physical and logical storage methods for IMAGE and TEXT data in DBMaster differ from SQL Server. Given the LONG VARCHAR and LONG VARBINARY data type, DBMaster will automatically allocate the physical storage. While the BLOB size is less than 3952 bytes (in 4k page size) , 8048bytes (in 8k page size), 16240 bytes(in 16k page size), 32624 bytes(in 32 k page size), the BLOB data could be stored together with normal data. If the data size is greater than 4K (4k page size for example), a pointer is used to indicate the LONG VARCHAR, LONG VARBINARY data. But the real BLOB data will be put into so-called ".BB" files. The other alternative is to use FILE data type. DBMaster uses FULL PATH link to indicate FILE data type. The physical data is stored externally as a file appearance.

This dynamical arrangement allows multiple columns of BLOB data per table and better performance. Similarly, in SQL Server, text data, ntext data and image data may be stored in a BLOB type field. SQL Server has no CLOB type and adds another data type—sql_variant type, it supports all kinds of data types (text, ntext, image). SQL Server also allows multiple BLOB columns per table. However, despite that SQL Server have different methods to handle ordinary data and BLOB data, it comprises BLOB and ordinary data into the same file MDF.

After the version 4.0 of DBMaster, the keywords BLOB and CLOB are applied to LONG VARBINARY and LONG VARCHAR. In most cases, you only map TEXT to CLOB, IMAGE to BLOB when migrating from SQL Server. But if the CHAR or VARCHAR size is greater than 4K, you should use LONG VARCHAR instead of the original data type.

5.3 Index Mapping

A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of slower writes and increased storage space. Indexes can be created by using one or more columns of a database table, providing the basis for both rapid random look up and efficient access of ordered records. The disk space required to store the index is typically less than that required by the table (since indexes usually contain only the key-fields according to which the table is to be arranged, and excludes all the other details in the table), yielding the possibility to store indexes in memory for a table whose data is too large to store in memory.

SQL Server	Description	DBMaster	Comments
Clustered index	Clustered indexes base on clustered key index sorting and are stored in sequence data in tables or views. Clustered indexes are realized by B_tree index structure, b_tree index structure supports quick retrieval for row basing on clustered index key. Only one clustered index can be created on a given database table. Clustered indexes can greatly increase overall speed of retrieval, but usually only where the data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items are selected. The primary feature of a clustered index is therefore the ordering of the physical data rows in accordance with the index blocks that point to them.	N/A	

Nonclustered index	SQL server creates a non-clustered index by default and can support as many as 249 non-clustered indexes per table. The data is present in random order, but the logical ordering is specified by the index. The data rows may be randomly spread throughout the table. The non-clustered index tree contains index keys in sorted order, with the leaf level of the index containing the pointer to the page and the row number in the data page. In non-clustered index: The physical order of the rows is not the same as the index order. Typically created on column used in JOIN, WHERE, and ORDER BY clauses. Good for tables whose values may be modified frequently.	**Index**	DBMaster creates an index by syntax: CREATE [UNIQUE] INDEX index-identifier ON base-table-name ({column-identifier | expression} [ASC|DESC]',...)[IN tablespace-name] [FILLFACTOR unsigned-integer] DBMaster supports as many as indexes per table, have no limit. But create an index on one or more columns, up to a maximum of 32 columns. DBMaster limits indexes to a maximum record size of 4000 bytes. Creating indexes for frequently used expressions will improve query performance.

Text indexes	A special type of functional index basing on the mark. Generated and maintained by text engine. To help users to in search of complex words on the string data.	**SIGNATURE TEXT INDEX**	DBMaster creates a signature text index by syntax: CREATE SIGNATURE TEXT INDEX text- index-identifier ON table_name(column_name,…) [TOTAL TEXT SIZE number] [MB SCALE number] [ORDER BY column_name ASC|DESC] Signature text indexes are built in the same tablespace as the column for which the index is being built. A text index provides fast access to rows that contain one or more words or phrases in columns containing text. Text indexes contain a representation of all the text found in the text columns they are based on. The data is encoded and structured to make retrieval much faster than directly from the table. Typically created on column used Order By clause. Rebuild the text index if you load data after creating text index. Text index names must be unique for each table. Text index names have a maximum length of thirty-two characters, and may contain numbers, letters, the underscore character, and the symbols $ and #. The first character can not be a number

		IVF TEXT INDEX	DBMaster creates a ivf text index by syntax: CREATE IVF TEXT INDEX text-index-identifier ON table_name(column_name,…) [STORAGE PATH path] [TOTAL TEXT SIZE number MB] [ORDER BY column_name ASC|DESC] IVF indexes are built in a separate file and exhibit better performance for larger indexes. An IVF text index can be used in place of a standard index to increase the performance of queries, particularly on columns that contain more than 200 MB of data. IVF indexes are sorted in the operating system's file system, and are administered through the database. The location where the IVF index should be stored is specified when the index is created. DBMaster manages the creation of sub-directories within the IVF index root directory. Besides these special features others are same as signature text index.

5.4 Support platform

DBMaster supporting platform includes Windows system and non-Windows (Unix, Linux,etc) But SQL Server supporting platform is limited to Windows Series only. When migrating with database, DBMaster give you the more flexibility to move to other platforms if needed.

Database	Platforms
Microsoft SQL Server	Windows Series
DBMaster	Windows, Linux, Others by request (FreeBSD, Sun Solaris, Sun SPARC, HP UX, AIX, DG/UX)

5.5 Data Manipulation Language (DML)

This section uses tables to compare the syntax and description of Data Manipulation Language (DML) elements in SQL Server and DBMaster databases. The following topics are present in this section:

- Connecting to the Database
- SELECT Statements
- SELECT with GROUP BY Statements
- INSERT Statements
- UPDATE Statements
- DELETE Statements
- Operators
- Comparison Operators
- Arithmetic Operators
- String Operators
- Set Operators
- Bit Operators
- Built-In Functions
- Character Functions
- Date Functions
- Mathematical Functions
- Locking Concepts and Data Concurrency Issues
- Locking
- Row-Level Versus Page-Level Locking
- Read Consistency
- Logical Transaction Handling

5.5.1 CONNECTING TO THE DATABASE

SQL Server	DBMaster	Description
Use DB_NAME	Connect to DB_NAME USER_NAME PASSWORD;	

Recommendations:

SQL Server installation can support multiple databases, and in DBMaster it could start one database by one server. In SQL Server, we should note that the so-called “role” role. There are two sorts of roles-” the server role “and “the database role”. It is not similar as the DBMaster; the server role is used to divide users into different groups, which possess the management privileges. The database role is used to grant privilege to single user or multi-users to connect to the database. In DBMaster, users are assigned to multiple groups or roles. It is only permitted to login in as one role. Only the highest privilege will be activated after logging into the DBMaster Database System.

5.5.2 SELECT STATEMENT

SQL server	DBMaster	Description
SELECT getdate();	SELECT curdate();	SELECT Statements without FROM Clauses
Select clause	Select clause	
SELECT [ALL | DISTINCT] [TOP n [PERCENT][WITH THES]] {select_list}	SELECT [ALL | DISTINCT] {select_list} {LIMIT offset, count}	The ALL keyword means every record regardless of its duplicate occurrence. However, using DISTINCT keyword will eliminate the duplicate rows. Users can extract the first n rows in SQL Server by using “Top n clause”. DBMaster provides more powerful functionality as “Limit offset, count” instead.
{select_list} { * | {table_name | view_name | table_alias},|{column_name | expression | IDENTITY [[AS] column_alias] | column_alias=expression} […n]	{select_list} { * | [remote-table-name@][owner.]{table | view | synonym} | {[remote-table-name@] [owner.]table.column | expression } [AS] alias}}	COLUMN ALIAS is defined by putting the alias directly after the selected COLUMN. This function is supported after DBMaster 4.0.3. You can also retrieve data from SYNONYMS. EXPRESSION could be a column name, a literal, a mathematical computation, a function, several functions combined, or one of several PSEUDO-COLUMNS.
From clause	From clause	

SELECT …FROM {<table_source>} [,...n] <table_source> ::= table_name [[AS] table_alias] [WITH (<table_hint> [,...n])] | view_name [[AS] table_alias] | rowset_function [[AS] table_alias] | OPENXML | derived_table [AS] table_alias [(column_alias [,...n])]	SELECT … FROM [[db-name[@server-name]:]user.]|[databaselink]table-name|view] [ALIAS]	The main difference is DBMaster must use DB_NAME@HOST:USERNAME.TABLE_NAME as its full valid table name. It would take effort to transfer the SQL server @dblink to its corresponding name in DBMaster. DBMaster supports database link currently.
Join table	Join table	
{Select clause} <joined_table> <joined_table> ::= <table_source> <join_type> <table_source> ON <search_condition> | <table_source> CROSS JOIN <table_source> | <joined_table> <join_type> ::= [INNER | { { LEFT | RIGHT | FULL } [OUTER] }] [<join_hint>] JOIN	table-reference [,] { LEFT OUTER JOIN | LEFT JOIN | OUTER } { table-reference | (table-reference-list) }	Note that SQL Server provides a "FULL OUTER JOIN", which is not implemented in DBMaster. User should use workaround to generate the wanted results.
Where,Group by ,having ,into clause	Where,Group by, having, into clause	
SELECT …FROM … WHERE <search_condition> | <old_outer_join> <old_outer_join> ::= column_name { *= | =* } column_name GROUP BY [ALL] group_by_expression [,...n] [WITH { CUBE | ROLLUP }] HAVING <search_condition>	SELECT …FROM …[WHERE] [GROUP BY (column-identifier [, column-identifier]…)] [HAVING search-condition]	Aggregate functions and syntax are identical in both databases. If a GROUP BY clause is used, all non-aggregate select columns must be in a GROUP BY clause.
INSERT INTO new_table SELECT FROM....	INSERT INTO <table> SELECT COLUMN_NAME_LIST FROM.... SELECT FROM… INTO <table>	You could retain the original syntax in SQL server or rewrite it to the 2nd syntax. It allows you to insert the results of the SELECT statement into a table.

5.5.3 INSERT STATEMENT

SQL server	DBMaster	Description
INSERT [INTO] { table_name WITH (<table_hint_limited> [...n]) | view_name | rowset_function_limited } { [(column_list)] { VALUES ({ DEFAULT | NULL | expression }[,...n]) | derived_table | execute_statement } } | DEFAULT VALUES	INSERT INTO remote-table-name [(column-identifier [, column-identifier]...)] { VALUES (insert-value[,insert-value]...) | DEFAULT VALUES | select-order-by-statement }	Insertings can only be done on single table views. Additionally, DBMaster has not implement the syntax "insert into table_name values (sub-query)". User might need to create temporary table or (sub-query) into table_name syntax as a workaround.

Recommendations:

The values supplied in the VALUES clause in either database may contain functions. The SQL Server-specific functions must be replaced with the equivalent DBMaster ones. In addition, DBMaster has not implement "INSERT INTO table_name VALUES (sub-query)" syntax. Users need to rewrite their SQL commands with the temporary table or (sub-query) into table_name syntax alternatively.

5.5.4 UPDATE STATEMENT

SQL server	DBMaster	Description
UPDATE { table_name WITH (<table_hint_limited> [...n]) | view_name | rowset_function_limited } SET {column_name = {expression | DEFAULT | NULL} | @variable = expression | @variable = column = expression } [,...n] {{[FROM {<table_source>} [,...n]] [WHERE <search_condition>] } | [WHERE CURRENT OF { { [GLOBAL] cursor_name } | cursor_variable_name}] } [OPTION (<query_hint> [,...n])]	UPDATE [remote-table-name@][owner.]{table | view } [table-option] SET column-identifier = {expression | subquery | NULL} [, column-identifier = {expression | subquery | NULL}]... [WHERE CURRENT OF cursor-name] | [WHERE search-condition]	A single subquery may be used to update a set of columns. This subquery must select the same number of columns (with compatible data types) as are used in the list of columns in the SET clause. The CURRENT OF cursor clause causes the UPDATE statement to affect only the single row currently in the cursor as a result of the last FETCH. The cursor SELECT statement must have included in the FOR UPDATE clause. Updates can only be done on single table views.

5.5.5 DELETE STATEMENT

SQL Server	DBMaster	Description
DELETE [FROM] { table_name WITH (<table_hint_limited> [...n]) | view_name | rowset_function_limited } [FROM {<table_source>} [,...n]] [WHERE { <search_condition> | { [CURRENT OF { [GLOBAL] cursor_name } | cursor_variable_name }] }] [OPTION (<query_hint> [,...n])] <table_source> ::= table_name [[AS] table_alias] [WITH (<table_hint> [,...n])] | view_name [[AS] table_alias] | rowset_function [[AS] table_alias] | derived_table [AS] table_alias [(column_alias [,...n])] | <joined_table> <table_hint_limited> ::= { FASTFIRSTROW | HOLDLOCK | PAGLOCK | READCOMMITTED | REPEATABLEREAD | ROWLOCK | SERIALIZABLE | TABLOCK | UPDLOCK } <table_hint> ::= { INDEX(index_val [,...n]) | FASTFIRSTROW | HOLDLOCK | NOLOCK | PAGLOCK | READCOMMITTED | READPAST | READUNCOMMITTED | REPEATABLEREAD | ROWLOCK | SERIALIZABLE | TABLOCK | TABLOCKX | UPDLOCK } <query_hint> ::= { { HASH | ORDER } GROUP | { CONCAT | HASH | MERGE } UNION | FAST number_rows | FORCE ORDER | MAXDOP | ROBUST PLAN | KEEP PLAN }	DELETE FROM remote-table-name [table-option] [WHERE search-condition]	FROM keyword is optional in SQL Server but elementary in DBMaster. In SQL Server, ALIAS can be specified for the table name as a correlation name, which can be used in the condition clause. But users can't use ALIAS in DBMaster. Deletes can only be performed through single table views in both SQL server and DBMaster. SQL Server provides many functions for performance concern. But DBMaster performs way better than SQL Server in DELETE operations. It is not very necessary to have this function in DBMaster.

5.5.6 OPERATORS

5.5.6.1 Operator comparison

Operator	Same in both Databases	SQL server only	DBMaster only
Equal to	=		
Not equal to	!=, <>		
Less than	<		
Greater than	>		
Less than or equal to	<=	!>	
Greater than or equal to	>=	!<	
Greater than or equal to x and less than or equal to y	BETWEEN x AND y		
Full Text Search		Contain	Match, Contains
Pattern Matches	LIKE 'a\%' ESCAPE '\'		Contain, Match
"a" followed by 1 character	LIKE 'a_'		
Does not match pattern	NOT LIKE		
No value exists	IS NULL		
A value exists	IS NOT NULL		
At least one row returned by query	EXISTS (query)		
No rows returned by query	NOT EXISTS (query)		
Equal to a member of set	IN,=ANY/SOME		
Not equal to a member of set	NOT IN != ANY/SOME, <> ANY/SOME		
Less than a member of set	< ANY/SOME		
Greater than a member of set	> ANY/SOME		
Less than or equal to a member of set	<= ANY/SOME		
Greater than or equal to a member of set	>= ANY/SOME		
Equal to every member of set	=ALL		
Not equal to every member of set	!= ALL, <> ALL		
Less than every member of set	< ALL		
Greater than every member of set	> ALL		
Less than or equal to every member of set	<= ALL		
Greater than or equal to every member of set	>= ALL		
Add	+		
Subtract	-		
Multiply	*		
Divide	/		
Modulo	Mod(x,y)	%	
Concatenate		+	||, Concat(substr1,substr2)
Identify Literal	'this is a string'		
Distinct row from either query	UNION		
All rows from both queries	UNION ALL		

5.5.6.2 Search String methods

DBMaster supports “MATCH”, “CONTAIN”, “CONTAINS”, and “LIKE” for pattern search.

Basically, “Like” operators will scan the whole record and seek the pattern as a token. DBMaster provides another operator “CONTAIN” to seek the word fragment. In addition, users could use “MATCH” operator to seek the full word. Only the MATCH and CONTAINS operators can be applied to a text index search.

SQL Server supports “CONTAINS”,” FREETEXT “and ‘LIKE’ for pattern search, basically,” LIKE” operators used to seek the word fragment. In addition, users could use “FREETEXT” and “CONTAINS” operators to seek the full word. Only the “FREETEXT” and “CONTAINS” operators can be applied to a TEXT INDEX search.

You could see the difference in the following case: DBMaster supports “Contain” and “Match” for pattern search and SQL Server supports “FREETEXT” for pattern search.

These keywords have totally different meaning and the syntax is different too. Users must know how to translate to the corresponding functions in DBMaster.

Table: Dept

ID	Name
1	DBMaster Support
2	Support SQL Server
3	SQL Server DBMaster

SQL Server	Usages	DBMaster	Usages
Like	Like is a predicate used in a WHERE clause to search columns for fuzzy search, in most cases, working with wildcard. Syntax: Expression [NOT] LIKE Condition	Like	This takes the form: x LIKE ‘y’ ESCAPE ‘z’; the LIKE condition is satisfied when the string value or expression to the left of the LIKE keyword meets the criteria specified in the case-sensitive quoted string to the right of the keyword. Syntax: Expression [NOT] LIKE Condition
Contains	A predicate used in a WHERE clause to search columns containing character-based data types for precise or fuzzy (less precise) matches to single words and phrases, the proximity of words within a certain distance of one another, or weighted matches. CONTAINS can search for: A word or phrase.	contains	The contains operator’s condition is satisfied when the concatenated string from concatenate columns matches the string pattern. Used in Full-Text-Index, so Create TEXT-INDEX first before using CONTAINS predicate. Syntax: CONTAINS (column_name,' contains_search_condition ');

	The prefix of a word or phrase. A word near another word. A word inflectionally generated from another Syntax: CONTAINS ({ column_name | (column_list) | * } , '< contains_search_condition >' [, LANGUAGE language_term]) Note:SQL Server support { < simple_term > | < prefix_term > } { **NEAR** | ~ } { < simple_term > | < prefix_term > } syntax to return adjacent rows and **FORMSOF** ({ **INFLECTIONAL** | **THESAURUS** } , < simple_term > [,...n]) do deformation matching If using these syntaxes, DBMaster doesn't support currently.		
N/A	SQL Server can't supported Match keyword, but we can use CONTAINS or LIKE as a replacement. One way is using 'LIKE' keyword and adding the percent symbol as wildcards in the quoted strings. The other way is use 'CONTAINS' instead but you must ensure a Text-Index had been build on these columns. Alternative syntax: Column_name LIKE '%condition%'; CONTAINS (column_name, 'condition');	match	This takes the form: x NOT CASE MATCH 'y'; the MATCH condition is satisfied when the quoted string to the right of the MATCH keyword matches the entire string value or expression to the left of the keyword. The NOT keyword inverts the search results and CASE keyword makes the search case-sensitive, Syntax: Column_name MATCH Condition
FreeText	FreeText command used to matching FULL SEARCH basing	N/A	DBMaster can't support FREETEXT keyword.

	on the deformation, literally, Synonyms. Syntax: FREETEXT ({ column_name | (column_list) | * } , 'freetext_string' [, LANGUAGE language_term])		We can use CONTAINS as a replacement. But we must ensure the string pattern precision. for example, you must write 'support' can find the records in which ID equal to 1 and 2.if you write pattern string 'supported' or 'supporting' no record will be return. Alternative syntax: CONTAINS (column_name,' contains_search_condition ');
N/A	SQL Server can't support CONTAIN keyword, but we can use CONTAINS or LIKE as a replacement. Alternative syntax: column_name LIKE '%condition%'; CONTAINS (column_name,'"*condition*"'); If using CONTAINS and asterisks wildcard character, conditions must be included in double quotes,	contain	This takes the form x NOT CASE CONTAIN 'y'; the CONTAIN condition is satisfied when the quoted string to the right of the CONTAIN keyword matches any part of the string value or expression to the left of the keyword. The NOT keyword inverts the search results and the CASE keyword makes the search case-sensitive, both are optional Syntax: column_name CONTAIN 'condition';

Recommendations:

Match and Contain are applied to full text search in DBMaster. DBMaster provides pattern matching for BLOBs. The CONTAIN and MATCH function are similar to the LIKE function except that wildcard characters are not supported. The difference between CONTAIN and MATCH is that the former is a partial word match and the latter is a full word match. For example, 'This is a character.' CONTAIN 'char' and 'this is a character.' MATCH 'character', but 'This is a character.' doesn't MATCH 'char'.

FREETEXT and CONTAINS are applied to Full text search in SQL Server. The difference between FREETEXT and CONTAINS is that the former supports the deformation, literally and thesaurus match and the later supports five-part which includes a word or a phrase, the prefix of a word or a phrase, a word near another word, a word inflectionally generated from another (for example, the word drive is the inflectional stem of drives, drove, driving, and driven),a word that is a synonym of another word using a thesaurus (for example, the word metal can have synonyms such as aluminum and steel).

Create TEXT-INDEX first before using CONTAINS or FREETEXT predicates in SQL Server. Match and Contains have no limit in DBMaster.

For example:

Like:

SQL Server:

```
Select * from dept where name LIKE '%DBM%'
```

	ID	Name
1	1	DBMaster Support
2	3	SQL Server DBMaster

DBMaster:

```
Select * from Dept where name like '%DBM%';
ID  NAME
1   DBMaster Support
3   SQL Server DBMaster
```

Contains:

SQL Server:

1. S

	ID	Name
1	1	DBMaster Support
2	3	SQL Server DBMaster

S (name,'DBMaster')

Note:

Execute a search to find any record containing words “DBMaster”.

The search for a word, double quotation marks are not required, only need a pair of single quotation marks.

2.

	ID	Name
1	1	DBMaster Support
2	3	SQL Server DBMaster

ONTAINS (name,'"*DBM*"')

Note:

The search will return any record that included ‘DBM’.

Use asterisks as one or more characters’ wildcard character. It’s similar with ‘LIKE’. If using wildcards, conditions must be included in double quote. Otherwise SQL Server will put asterisks as text value to search, for example, searching '*DBM*' without double quotation equals to put asterisks literal value as the part of search condition to process.

DBMaster:

```
1. Select * from dept where name CONTAIN ‘DBMaster’;
   ID  NAME
   1   DBMaster Support
   3   SQL Server DBMaster
2. Select * from dept where name CONTAIN ‘DBM’;
   ID  NAME
   1   DBMaster Support
    3   SQL Server DBMaster
```

FreeText:

SQL Server:

```
Select * from dept where FREETEXT (name,’ support')
Select * from dept where FREETEXT (name,’ supporting')
```

	ID	Name
1	1	DBMaster Support
2	2	Support SQL Server

e FREETEXT (name,’ supported')

Note:

The search will return records that include each tense of word ‘support’. Despite both ‘supported’ and ‘supporting’ are not exist in the records, but returned 2 rows, because find the past tense and doing tense about ‘support’.

However, FREETEXT is a vague way to search text index compared with CONTAINS.

DBMaster:

```
Select * from dept where CONTAINS (name,’ support');
ID    NAME
1     DBMaster Support
2     Support SQL Server
```

Match:

SQL Server:

```
Select * from Dept where name like '%DBMaster%';
Select * from Dept where contains (name,'DBMaster');
```

DBMaster：

```
Select * from Dept where name match 'DBMaster';
ID  NAME
1   DBMaster Support
3   SQL Server DBMaster
```

Contain:

SQL Server:

```
Select * from Dept where name like '%DBM%';
Select * from Dept where contains (name,'"*DBM*"');
```

DBMaster:

```
Select * from Dept where name contain 'DBM';
ID  NAME
1   DBMaster Support
3   SQL Server DBMaster
```

5.5.7 BUILT-IN FUNCTIONS

The user who reads the following table and functions listed will get surprise that SQL server had so many common functions as DBMaster. We classify all the functions into four categories:

- Math/Number Functions
- Character Functions
- Conversion Functions
- Date Functions

Reference functions, such as REF, DEREF are rarely seen in any other RDBMS. In addition, some unique functions of SQL server have not been put here for its uniqueness.

In most cases, users would need very little effort to migrate SQL server functions to DBMaster functions, the SQL server unique functions or Object-Reference functions are not commonly seen after all.

5.5.7.1 Math/Number Functions:

SQL server	DBMaster	Description
ABS(numeric_ expression)	ABS(n)	Return the absolute value of x as a double-precision floating-point number.
ACOS(float_expression)	ACOS(n)	Return the arc cosine of n in the range 0 to pi as a double-precision floating-point number.
ASIN(float_expression)	ASIN(n)	Return the arc sin of n in the range -pi/2 to pi/2 as a double-precision floating-point number.
ATAN(float_expression)	ATAN(n)	Return the arc tangent of n in the range -pi/2 to pi/2 as a double-precision floating-point number.
ATN2 (float_expression , float_expression)	ATAN2(n)	Return the arc tangent of *x*/*y* in the range -pi to pi as a double precision floating-point number.

CEILING(numeric_expression)	CEILING(n)	Return the least integral value greater than or equal to n as a double-precision floating-point number.
COS(float_expression)	COS(n)	Return the cosine of n as a double-precision floating-point number. n is expressed in radians.
EXP(float_expression)	EXP(n)	Return the exponential function e**x as a double-precision floating-point number.
FLOOR(numeric_expression)	FLOOR(n)	Return the greatest integral value less than or equal to x as a double-precision floating-point number.
LOG(float_expression)	LOG(n)	Return the natural logarithm of x as a double-precision floating-point number.
LOG10(float_expressionr)	LOG10(n)	Return the logarithm to base 10 as a double-precision floating-point number.
N/A(%)	MOD(m,n)	Return the remainder (modulus) of m divided by n as a double-precision floating-point number.
POWER(m,n)	POW(m,n) POWER(m,n)	Return x**y as a double-precision floating-point number.
ROUND (numeric_expression , length [,function])	ROUND(n)	Return the closest integer number of the real number x. different from DBMaster in SQL Server need specify length via second parameter
SIGN(numeric_expression)	SIGN(n)	Return the sign of a number codes as +1 for positive, 0 for zero, and -1 for negative. Returns an integer value 1, 0 or -1.
SIN(float_expression)	SIN(n)	Return the sine of n as a double-precision floating-point number. n is expressed in radians.
SQRT(float_expression)	SQRT(n)	Return a double-precision floating-point number y where x = y*y.
SQUARE(float_expression)	N/A	Instead of POW(x,y) or POWER(x,y) and make sure x equal to y. Returns x**y as a double-precision floating-point number.
TAN(float_expression)	TAN(n)	Return the tangent of n as a double-precision floating-point number. n is expressed in radians.
DEGREES (numeric_expression)	DEGREES(n)	Return the number of degrees in radians as a double precision floating-point number
RADIANS(numeric_expression)	RADIANS(n)	Return the number of radians in *degrees* as a double precision floating-point number.
PI()	PI()	Return the constant value of p, 3.1415926535897936, as a decimal number with a precision of 38 and a scale of 16.
RAND ()|(n)	RAND ()	Return a random Integer value. In SQL Server rand () function can have parameters but DBMaster can't.

5.5.7.2 Character Functions:

SQL server	DBMaster	Description
ASCII(<char_expression>)	ASCII(string)	Return the ASCII code value of the leftmost character of string_exp as an integer. These 2 functions between DBMaster and SQL server are identical.
CHAR(<integer_expression>)	CHAR(INT code)	Convert the decimal code for an ASCII character to the corresponding character. These 2 functions between DBMaster and SQL server are identical.
UNICODE ('ncharacter_expression')	N/A	Return the unicode value of the first character of string as an integer according to the Unicode definition.
NCHAR (integer_expression)	N/A	Convert to the corresponding character for a Unicode character.
CHARINDEX (expression1 ,expression2 [, start_location]) | CHARINDEX (expression1 ,expression2)	LOCATE(STRING string_exp1, STRING string_exp2, INT start)	Return the starting position of the first occurrence of string_exp1 within string_exp2, The search for the first occurrence of string_exp1 begins with the first character position in string_exp2 unless the optional argument, start, is specified. (1) If either string_exp1 or string_exp2 is null, the result should be null (2) If start is null, return 0 (3) If string_exp1 is an empty string, the result should be 1 In SQL Server if not specify the start position default begin with the first character position in string2 to search. but in DBMaster must specify the third parameter.
LEN(string_expression)	LENGTH(string) CHAR_LENTTH(string) CHARACTER_LENTTH (string)	Compute the length allocated to an expression, giving the result in bytes. These two functions between DBMaster and SQL server are identical.
SUBSTRING(expression, start, length)	SUBSTRING(string_exp, int start, int length)	Return the part of the string. Note that start position in DBMaster must be negative. However, SQL server could use negative to indicate scan the string backward.
COALESCE(expression [,…,n])	COALESCE (variable, new_value)	If the value of the variable is NULL, the new_value is returned.
1.CASE input_pression WHEN when_expression THEN result_expressuion[….,n][ELSE else_result_expression] END 2 CASE WHEN Boolean_express THEN result_express[…,n][ELSE else_result_expression] END	1.CASE input_pression WHEN when_expression THEN result_expressuion[….,n][ELSE else_result_expression] END 2.CASE WHEN exp1 THEN result1 WHEN exp2 THEN result2 ELSE default_value END	CASE compares expr to each search value one by one. If expr is equal to a search, then returns the corresponding result. If no match is found, then SQL Server returns default. If default is omitted, then SQL server returns null. In DBMaster, you could use CASE WHEN…THEN… WHEN…THEN… END to implement the same function.

REPLICATE (string_expression ,integer_expression)	REPEAT(string_exp, int count)	Produces a string with char_ exp repeated n times.
UPPER(character_exprtession)	UPPER(String), UCASE(String)	Convert lowercase characters to uppercase characters. These three functions between DBMaster and SQL server are identical.
LOWER(character_exprtession)	LOWER(string), LCASE(string)	Convert all upper case characters in string_exp to lower case. These three functions between DBMaster and SQL server are identical.
LTRIM(character_exprtession)	LTRIM(char_exp)	Truncate trailing spaces from the left end of char_exp. These two functions between DBMaster and SQL server are identical.
RTRIM(character_exprtession)	RTRIM(char_exp)	Truncate the trailing spaces from the right end of char_exp. These two functions between DBMaster and SQL server are identical.
N/A	TRIM(char_exp)	Truncate the trailing spaces from both end of char_exp.
REPLACE('string_expression1,''string_expression2,''string_expression3'	REPLACE(string_exp1, string_exp2, string_exp3)	Replace all occurrences of string_exp2 in string_exp1 with string_exp3. These 2 functions between DBMaster and SQL server are identical.
SOUNDEX(character_exprtession)	N/A	Return the numeric difference of the SOUNDEX values of the string. DBMaster hasn't supported this yet.
RIGHT (character_expression , integer_expression)	RIGHT(string_exp1,n)	Return the rightmost count characters in string
LEFT (character_expression , integer_expression)	LEFT(string_exp1,n)	Return the leftmost count characters in string
N/A	CONCAT(string_exp1,string_exp2) ||	Return a character string that is the result of concatenating string_expr2 to string_ep1. The resulting string is DBMS dependent.
STR (float_expression [, length [, decimal]])	N/A	Return a character data that is the result of converted by digital data.
STUFF (character_expression , start , length ,character_expression)	N/A	Return a character string that is the result of the one string insert into another string .It will delete the specified length character of the first string from beginning, and then will insert the second string into the first string from start location.
REVERSE (character_expression)	N/A	Return a character string of reverse expression.
SOUNDEX (character_expression	N/A	Return a character code (SOUNDEX) consists of four characters used to assess the similarity of the two strings.
DIFFERENCE (character_expression , character_expression)	N/A	Return an integer value to specify the difference of SOUNDEX between the two character expressions.
PATINDEX ('%pattern%' , expression)	N/A	Return the beginning position of one mode first occurring in one expression.

5.5.7.3 Conversion Functions:

SQL server	DBMaster	Description
CAST(expression as Datatype) CONVERT(<datatype>[(<length>)],<expression>,[,<style>])	CAST(Column as Datatype)	DBMaster uses CAST function to cast one data type to another. SQL server is able to set up all data types in the sql including bigint and sql_variant

5.5.7.4 Date Functions:

SQL server	DBMaster	Description
DATEADD(datepart,number,date)	ADD_DAYS(DATE date_val,INT s)	Add the int_exp number of days to the date contained in datetime_var. In sql ,the datepart contains the Year,Quarter, Month ,dayofyear, day, Week, Hour, minute, second, millisecond. Specify the first parameter datepart 'D'.
DATEDIFF(datepart,startdate,enddate)	DAYS_BETWEEN(DATE date1,DATE date2)	Return the number of days between the given two dates. date1 can be earlier or later than date2.
DATEADD(datepart,number,date)	ADD_MONTHS(DATE date_val,INT s)	Return a date which is got from adding s months to date_val. s can be a negative number. Specify the first parameter datepart 'M'.
N/A	CURDATE()	Return current date.In SQL Server we can use select statement CONVERT(varchar,year(getdate()))+'-'+CONVERT(varchar,month(getdate()))+'-'+CONVERT(varchar,day(getdate())) to replace
GETDATE()	NOW()	Return current date and time as a timestamp value.
DATEPART(<date_part>,<date>)	YEAR(date),MONTH(date),WEEK(date), QUARTER(date), DAYNAME(date), DAYOFYEAR(date), DAYOFMONTH(date), DAYOFWEEK(date), DATEPART(date), TIMEPART(date), MDY(date), HMS(date), HOUR(date), MINUTE(date), SECOND(date)	Return the specified part of the date as an integer.
N/A	LAST_DAY(dateval)	Return the last date of the month which dateval belongs to.
DATEDIFF(<date part>,<date1>,<date2>)	DAYS_BETWEEN(date1,date2)/30	Return the difference between the dates specified by the datetime1 and datetime2 variables. This difference is calculated in the number of months.
	NEXT_DAY(dateval,weekday)	Return the date of the next first Weekday.
DATE()	NOW()	Return the system date.

Note： Above comparing descriptions in table mainly base on DBMaster.

- Math/Number functions in SQL Server2008

The math/number functions (e.g. CEILING, ABS, FLOOR, POWER, DEGREES, RADIANS AND SIGN) return values and the input values, and both of them are same data type. But triangle function and other functions (including EXP, LOG, LOG10, and SQRT) will convert input values to float and return float values.

- Character Functions in SQL Server2008

The character function e.g. QUOTENAME SQL Server proprietary uncommon function we don't list and introduce here.

- Date Functions in SQL Server2008:

Date function refers to time zone that we don't mention in this document.

5.5.8 LOCKING CONCEPTS AND DATA CONCURRENCY ISSUES

SQL server	DBMaster
SQL server supports table-lock, page-lock and row-lock	DBMaster supports the record-lock, page-lock and table-lock. DBMaster supports the "Dirty Read", "Read with shared lock", and "Read with exclusive lock"

Recommendations:

In SQL server, before positioning UPDATE or DELETE statement, the SELECT statement declaration must contain the FOR BROWSE option. SQL Server2000 FOR BROWSE option is similar to the FOR UPDATE option in other SQL databases. This operation helps to prevent conflicts with other users. DBMaster, use the "select * for update" to prevent other session from updating the locked data. Basically, the reader of the data is never blocked both in SQL server and DBMaster. But users should be aware of the different manners when a "select" command is submitted on these two databases and its consequences.

SQL server	DBMaster
♦ SQL server transactions are explicit. Statements are automatically committed to the database. But users could change this pattern, for example use" SET implict_transactions{ ON | OFF }"to achieve the different state ♦ COMMIT WORK commits the pending changes to the database. ♦ ROLLBACK undoes all the transactions after the last COMMIT WORK statement. ♦ Savepoints can be set in transactions with the following command: SAVE TRAN savepoint_name ♦ The following command rolls back to the specified SAVEPOINT; ROLLBACK savepoint_name two-phase commit is automatic and transparent in SQL server. Two-phase commit operations are needed only for transactions, which modify data on two or more databases.	♦ DBMaster transactions are explicit. Statements are automatically committed to the database by default. But users could change the DB_ATCMT=0 to change this pattern, or use "set autocommit off" to achieve the same effect. ♦ COMMIT WORK commits the pending changes to the database. ♦ ROLLBACK undoes all the transactions after the last COMMIT WORK statement. ♦ Savepoints can be set in transactions with the following command: SET SAVEPOINT savepoint_name ♦ The following command rolls back to the specified SAVEPOINT; ROLLBACK <savepoint_name> Two-phase commit is automatic and transparent in DBMaster. Two-phase commit operations are needed only for transactions, which modify data on two or more databases.

Recommendations:

Transactions are not implicit in DBMaster and SQL Server. Therefore, applications expect that every statement they issue is automatically committed after it is executed. In DBMaster, you could use "DB_ATCMT=1" in dmconfig.ini or set autocommit on/off to change this manner and In SQL Server you can use "SET implict_transactions {ON | OFF}" to change state.

When converting an SQL Server application to a DBMaster application, you don't need to do some additional work. In DBMaster, transactions may also be explicitly begun by a client application by issuing a BEGIN TRAN statement during the conversion process. AUTOCOMMIT is default manner in DBMaster and SQL Server. Thus these two Databases will automatically commit every statement and write changes to Journal Files. It seems to bring some convenience. As a matter of fact, too many I/O for writing journals delays the operations. Users could easily find greatly improvement if turning off the "AUTOCOMMIT" option.

5.5.9 UDF DIFFERENCE

A UDF is a method that can be called in the context of a statement, can take any number of parameters, and can return any type of data.

SQL server	DBMaster
CREATE FUNCTION [schema_name.] function_name ([{ @parameter_name [AS][type_schema_name.] parameter_data_type [= default] [READONLY] } [,...n]]) RETURNS return_data_type [WITH <function_option> [,...n]] [AS] BEGIN function_body RETURN scalar_expression END [;]	CREATE FUNCTION <udf_dll_name.function_name> (<function_datatype>) RETURN <function_output_datatype>;

Recommendations:

Both SQL Server and DBMaster allow programmers to build their own user-defined functions (UDF). Once a UDF has been written in SQL Server or DBMaster, it is treated as a new built-in function with the same usages.

There is great difference exist between SQL Server and DBMaster although both of them have UDF objects.

In SQL Server, we can create user defined function (UDF) by T-SQL script language so it with more powerful in Access to the database. Whereas DBMaster use C language as the carrier to create UDF, More embodies the function and feature of C language. But it doesn't represent very well in data access aspect, Because of this user need to spend time and do a careful technical evaluation if using UDF in your application programs.

5.5.10 TRIGGER DIFFERENCE

SQL server	DBMaster

CREATE TRIGGER trigger_name
ON { table | view }
[WITH ENCRYPTION]
{
{ { FOR | AFTER | INSTEAD OF }
{ [DELETE] [,] [INSERT] [,] [UPDATE] }
[WITH APPEND]
[NOT FOR REPLICATION]
AS
sql_statement [...n]
} |
{ FOR | AFTER | INSTEAD OF { [INSERT] [,]
[UPDATE] }
[WITH APPEND]
[NOT FOR REPLICATION]
AS
{ IF UPDATE column
[{ AND | OR } UPDATE column]
[...n]
| IF COLUMNS_UPDATED { bitwise_operator }
updated_bitmask
{ comparison_operator } column_bitmask [...n]
}
sql_statement [...n]
}
}
	Create Trigger trigger_name {Before|After}
	{Insert|Delete|Update[OF column_name]} On
	Table_name {FOR EACH ROW|FOR EACH
	STATEMENT}[When trigger_condition] trigger_body

ALTER TRIGGER trigger_name ON table | view [WITH ENCRYPTION] { { FOR | AFTER | INSTEAD OF { [DELETE] [,] [INSERT] [,] [UPDATE] } [NOT FOR REPLICATION] AS sql_statement [...n] } | { FOR | AFTER | INSTEAD OF { [INSERT] [,] [UPDATE] } [NOT FOR REPLICATION] AS { IF UPDATE column [{ AND | OR } UPDATE column] [...n] | IF COLUMNS_UPDATED { bitwise_operator }updated_bitmask { comparison_operator } column_bitmask [...n] } sql_statement [...n] } }	ALTER TRIGGER trigger_name REPLACE WITH
DROP TRIGGER Trigger_name	DROP TRIGGER Trigger_name FROM Table_name
Trigger events	

There are three trigger events in the SQL Server：FOR, AFTER,INSTEAD OF	**There are four trigger events in the DBMaster**： BEFORE…FOR EACH STATEMENT”,”BEFORE…FOR EACH ROW”,”AFTER…FOR EACH STATEMENT” ,”AFTER…FOR EACH ROW” Note: In SQL Server, “FOR” has the similar meaning as “AFTER” in DBMaster.
function support	
Reference objects： SQL Server support Trigger applies to both **table** and **view** object **action time**： AFTER **trigger type**： table triggers **data reference**： deleted and inserted tables hold the old values or new values of the rows that may be changed by the user action in SQL Server	**Reference objects**： DBMaster only supports **Table** object currently. **action time**： AFTER，BEFORE **trigger type**： Row triggers and statement triggers. **data reference**： DBMaster has new and old buffer corresponding with deleted and inserted in SQL Server.

Recommendations:

There are lots of differences between SQL Server and DBMaster in the trigger.

Firstly, in syntax level, they are different from each other.

For example, creating a trigger tr0 on the table “w” in SQL Server, users would use the command-” create trigger tr0 on w after insert as insert into q values (300,300)”.Yet in the DBMaster users need to rewrite as ”create trigger tr0 after insert on w for each row (insert into q values (300,300));” The syntax difference would need a bit of attention.

Secondly, there are three trigger events in SQL Server, including the “FOR”, “AFTER”,” INSTEAD OF”. But there are four trigger events in DBMaster - “ BEFORE…FOR EACH STATEMENT”,”BEFORE…FOR EACH ROW”,”AFTER…FOR EACH STATEMENT” ,”AFTER…FOR EACH ROW”. In SQL Server, “FOR” has similar meaning as “AFTER” in DBMaster.

The primary advantage of INSTEAD OF triggers in SQL Server is that they allow views that would not be updatable support update. A view comprising multiple base tables must use an INSTEAD OF trigger to support insert, update and deletion reference data in the tables. Another advantage of INSTEAD OF triggers is that they allow you to code logically that can reject parts of a batch while other parts of a batch are allowed to succeed. Please refer to the SQL Server On-Line Help if interested.

In DBMaster, the keyword “FOR EACH ROW” means that if one record was modified in the database then the trigger would be executed. The keyword” FOR EACH STATEMENT” means execute the action before or after the SQL statement. The keyword [WITH ENCRYPTION] of trigger in SQL Server is used to encrypt the table sys-comments that store the information about triggers. However, DBMaster doesn’t provide such syntax when creating the schema object. Users must remove the original “WITH ENCRYPTION” from SQL Server syntax.

One of the most powerful features of a trigger is the ability to use a stored procedure as a trigger action in DBMaster. Therefore, if some complex job can't be done in DBMaster, try to implement it in triggering Stored Procedures.

Thirdly, functions support

1) Reference objects, SQL Server supports Triggers applies to both table and view objects but DBMaster only supports Table objects currently.

2) About trigger action time. Trigger for INSERT, UPDATE, and DELETE operations can be specified on a table or a view. SQL Server only supports AFTER fire. But DBMaster supports both AFTER and BEFORE.

3) About trigger types, SQL Server only supports table triggers but DBMaster includes row triggers and statement triggers. Namely either row or table data once change triggers will be fire.

4) Last one is about trigger data reference. DML triggers use the deleted and inserted logical (conceptual) tables. They are structurally similar to the table on which the trigger is defined, that is, the table on which the user action is tried. The deleted and inserted tables hold the old values or new values of the rows that may be changed by the user action in SQL Server. Different with SQL Server, DBMaster have new and old buffer corresponding with deleted and inserted in SQL Server.

5.5.11 STORED PROCEDURE AND STORED FUNCTION

SQL server	DBMaster
CREATE PROC [EDURE] procedure_name [; number] [{ @parameter data_type } [VARYING] [= default] [OUTPUT]] [,...n] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION }] [FOR REPLICATION] AS sql_statement [...n]	CREATE PROCEDURE procedure-name [(procedure-parameter [, procedure-parameter ...])] { [RETURNS STATUS] | [RETURNS [STATUS,] procedure-result [,procedure-result ...]] } | CREATE PROCEDURE FROM source-file-path

Recommendations:

In SQL Server, there are two types of stored procedures. The first one is the system stored procedures which are stored in the "master" database with the "SP_ "prefix .The other is the user stored procedure, which is put in the user database.

SQL Server stored procedures use the Transact-SQL. On the other hand, DBMaster uses the ESQL/C for ESQL/C stored procedures or Java for Java stored procedures to do coding. But future release version-5.2 version can supports SQL SP (Script Stored Procedure). This is the biggest difference between these two provided stored procedures (ESQL/JAVA). T-SQL is the core components of SQL Server. T-SQL includes the commands that can create the logic store cells. T-SQL could be used to add and manage data or the other database objects. If T-SQL commands were stored in SQL Server, it would be referred to as "Stored Procedures".

To develop ESQL/C stored procedures, DBMaster has to hook up to the external C-Compiler. This compiler is usually VC in Windows Platform, GCC in Linux. The normal process to build a C-Compiler in DBMaster is: compile the stored procedure, put it into the corresponding folder, and create the procedure in dmsqlc with the syntax "create procedure from …" syntax. How to write procedural Language in DBMaster is beyond the scope of this document. For details, please refer to the "ESQL C Programmer's Guide" manual.

For Java stored procedures, if you know how to access a database using Java programs, the coding and creating processes are very easy and fast.

5.5.12 SQL SERVER 2008 AND DBMASTER IN AP

SQL server	DBMaster
Supported driver: JDBC/ODBC,Hibernate,Nhibernate,OLE DB	**Supported driver:** JDBC/ODBC, DCI, Ruby, Hibernate, Nhibernate,OLE DB

Connection String:
 	
ODBC:
"Driver={SQL Server};" +
"Server=Server Name;" +
"Database=DarabaseName;" +
"Uid=Username;" + "Pwd=Password;"
 	
OLE DB:
"Driver=SQLOLEDB;" +
"Data Source=ServerName;" +
"Initial Catalog=DataBaseName;" +
"User id=UserName;" +
"Password=Secret;";
 	
JDBC:
 	
jdbc:SQL Server://IP_Address:TCP_Port;
databaseName=DatabaseName;
user=UserName;
password=Password;
	Connection String:
	
	ODBC:
	"Driver={DBMaster 5.1 Driver};Database=Database name;uid=Username;Pwd=Password;"
	
	OLE DB:
	"Provider=DMOLE51;Data Source= Databasename;User Id= Username; Password =;"
	
	JDBC:
	
	Class.forName("dbmaster.sql.JdbcOdbcDriver ");
	Connection conn= DriverManager.getConnection (jdbc: dbmaster:// IP_Address:TCP_Port /DatabaseName, user, password);

5.6 System Tables

Each database has its system table. User maybe need query these tables to get some informations.

We list three of them as following:

SQL Server	DBMaster
Check one table/view exist	
select 1 from sys.tables where name = 'XXXXX'	select 1 from systable where table_name='XXXXX'
select 1 from sys.views where name = 'XXXXX'	select 1 from sysviewdata where view_name='XXXXX'
Check DB Version from SQL	
select serverproperty('productversion')	select value from sysinfo where info='VERSION'
Check Procedure exist	
select count(*) from sysobjects where name='XXXXXXX' and Type='P'	select count(*) from sysprocinf where modulename = 'XXXXXXX'

6. DB Object Migration procedures

6.1 Schema and Date Migration

Please refer to chapter 4 for more information about how to migrate a database from SQL Server to DBMaster.

You should rebuild indexes, constrains and so on after migration.

6.2 Convert UDF

There are great difference exists between SQL Server UDF and DBMaster UDF. Please read the detailed introduction about it in chapter 5 sections 5.5.9.

First, we should analyze the UDF function in SQL Server.

Second, we can rewrite UDF according to DBMaster syntax.

Note: Please spend some time for a careful technical evaluation before using UDF.

6.3 Convert Trigger

SQL Server stored produces have three kinds, including DML, DDL, or logon triggers. DBMaster only includes DML stored produces. This article will focus on (DML) triggers.

Difference in the Trigger between SQL Server and DBMaster had been introduced in chapter 5 section 5.5.10 which include syntax levels, trigger events, ”for each row/statement” syntax, ”after/before” syntax and so on.

First, we should analyze the SQL Server Trigger.

Second, we can rewrite Triggers according to DBMaster syntax.

Note: some syntax we can’t support which should be replaced with other methods. For example, we think to write a stored procedure for some processes.

6.4 Convert Stored Procedure

Detailed Recommendations for stored procedures between SQL Server and DBMaster have been introduced in chapter 5.5.11. Stored Procedure and Stored Function”. Here we mainly discuss how to convert stored produces from SQL Server to DBMaster successfully. SQL Server stored procedures use the T-SQL but DBMaster uses the ESQL/C for ESQL/C stored produces or java for java stored procedures to do coding. T-SQL includes the commands that can create the logical store cells. DBMaster can create logical store cells with SQL SP in release 5.2. And in current DBMaster version, we can develop ESQL/C stored procedures with external C-Complier or Java stored produces.

Because the difference is so big as above description, we can’t convert them directly. So we should do following things step by step.

1. First, we have to analyze the purpose of the stored procedures created by T-SQL in SQL Server.
2. Next, we need choosing one language from ESQL/C and Java for creating stored procedure.
3. Rewriting the stored produces with suitable syntax for DBMaster and make it having same actions as the old one in SQL Server.
4. Creating and testing the stored procedure in DBMaster.

Note: For moe details description about creating stored procedures by ESQL/C or JAVA, please refer to the “ESQL C Programmer’s Guide” manual or “Creating Stored procedures using Java” section in DBA manual.

7. AP migration procedures

It's very important for us to check application program interfaces first. For example, we should check whether the interfaces are supported by DBMaster if we want to migrate them from another database.

Next, we must consider how to rewrite the connect strings according to the driver.

Finally, mark the special syntax in SQL Server and find the solution for DBMaster.

7.1 AP interface and Connect string

We must make clear what kinds of interfaces are used in application programs, and whether these interfaces are supported by DBMaster.

What types of data provider or divers are used to access data source. JDBC, ODBC or any others, for example: If data provider changes, we might consider changing driver.

We can discuss each tier from following aspects.

Finally, you'd better do a quick testing for the application program that has been modified. In order to make sure it can connect to DBMaster successfully.

7.1.1 AP IN CLIENT

A part of application program codes that related to database connection or manipulation may need to do some modifying. Such as DSN, CONNECT SRTING and so on in client.

In addition, if the application need get some information from SYSTEM Table (or CATALOG). Please refer to the chapter 5.5 to modify the usage.

7.1.2 MIDDLE-TIER

If use COM+ or implement DB-tier encapsulation implemented with similar technology, users need to consider modifying connect string and any other parameters of COM components in DB-tier.

7.1.3 AP OR (WEB) SERVER

Regarding AP server, users may need to modify some parameters that related to DB Server such as Server IP address, Port Number, Driver etc.

7.1.4 AP IN SERVER

Here, users need to check whether there are some schedules or tasks deployments exist in server separately and whether these programs need modifying.

7.2 SQL Server special syntax and feature

On one hand, we must solve connect situation, on the other hand, we must pay special attention to special syntax in SQL Server. Consider what method is substitute for these special grammars.

There are too many special syntax exists in SQL Server. In order to find replaced solutions for an alternative. We give some simple samples as followings. You must understand this aspect of knowledge about SQL Server and DBMaster before migration. You also can Comparison with chapter 5 that describes the difference between SQL Server and DBMaster. For more information you can reference SQL Server and DBMaster User Guide.

7.2.1 FOR INSERT STATEMENT

In SQL Server, you can write a command “insert [into] table_name values (data)”,”into” keywords can be omitted, but in DBMaster, you can’t omit “into”. So if you insert statements lacking of “into” keyword you must add it and make statement integrated when you transfer it to DBMaster.

7.2.2 FOR “TOP” KEYWORD

In SQL Server, you can write “select top count columns_name from table_name”, but in DBMaster, “top” keyword isn’t supported, we must use “select columns_name from table_name limit count” to realize the same function.

7.2.3 FOR NESTED QUERY

Suppose we have a table named tb_nest that records all staff information. If we want to know who is the latest one for each department.

In SQL Server, we can write following statements

```
select ID,emp_from,name,content,come_date
       from sysadm.tb_nest tb1 where
       ID=(select top 1 ID from sysadm.tb_nest tb2
       group by emp_from ,come_date,ID having tb2.emp_from = tb1.emp_from
       order by come_date desc)
```

In DBMaster, the grammar isn’t supported. In order to achieve the same function in DBMaster, we adopt the method of temporary table by rewriting statements.

```
select emp_from, max (come_date) as come_date from tb_nest group by emp_from
into temp;
select * from tb_nest tb1 join temp tb2 on tb1.emp_from =
tb2.emp_from and tb1.come_date=tb2.come_date
```

8. Testing application with new DB

Testing applications are required at any moment, at the beginning, in the process or at the end of migration. It can help us confirm our modifications or adjustments to be befitting.

8.1 How to pre-run for skip any object

In order to find problems timely and get to know where the problems exist, we must test the program every time to find out which part has something wrong.

It's better for us to begin migrating next section after having tested and ensured the part of you just finished has no problems. This is very helpful for you to migrate all application programs from SQL Server to DBMaster successfully.

8.2 Test application with DBMaster after migration

A validate testing is required after whole application programs have been migrated completely from SQL Server to DBMaster. You can ensure the application run normally on new platform with the validate testing.

9. Performance tuning

When you develop an application system with any database, the system performance is an important thing and you must be concerned about it, we must tune database after migration and make sure the application program run efficiently. Of cause, performance tuning is about the whole processes of using database not only tuning after migration .The amount data is growing in database. You should pay attention to database performance tuning often. If any database performance down, we should detect database and adjust timely in use.

Performance tuning need adjusting not until migration finished from SQL Server to DBMaster. It's from the beginning design and planning the whole db to the end use.

Generally speaking, there are many factors affecting the performance of DBMaster. We can see them from the following figure.

Application System	Query Optimization
	Concurrent Process
	Application System Architecture
	Database Model Design (Tablespace, Table, Index, Stored command, Stored procedure, Trigger)
Database System	Daemon (Auto-commit, Checkpoint, Update statistic, Backup server, Replication)
	Memory Allocation
	Disk I/O (Database Data Partition)
OS	(File system, Raid)
Hardware	Network
	I/O
	Memory
	CPU

9.1 Application

It comprises writing queries that limit the use of stored commands or searches for procedures. Designing a good schema or developing an application with better utilities can both significantly increase applications perform.

Using indexes can improve the application performance for accessing to database if you built the index reasonable. For example, if you build some indexes only on the required columns in a table. DBMaster will find the data effectively.

Another attention for Applications is Concurrent Processes. Obviously, minimizing lock contention and avoiding deadlocks can increase throughput of applications. In addition, shortening transactions can promote concurrency, but it is possible to degrade database performance oppositely.

9.2 Database System

It includes **Disk I/O**, **Memory Allocation** and **Daemon**. Make sure there are enough physical memory for DCCA and few I/O access times.

9.2.1 Tuning Memory Allocation

DBMaster stores information temporarily in memory buffers and permanently on disk. Since it takes much less time to retrieve data from memory than disk, performance will increase if data can be obtained from the memory buffers. The size of database memory allocation will affect performance of a database. However, performance will become an issue only if there is not enough memory. So we must tune the memory usage for a database and it includes how to calculate the required DCCA size, and how to monitor and allocate enough memory for the page buffers, journal buffers and system control area.

To achieve the best performance, follow the steps in the order shown:

1. Tune the operating system.
2. Tune the DCCA memory size.
3. Tune the page buffers.
4. Tune the journal buffers.
5. Tune the SCA.

Memory requirement for DBMaster varies according to the applications in use, tune memory allocation after tuning application programs and SQL statements.

9.2.1.1 Tuning an Operating System

The operating system should be tuned to reduce memory swapping and ensure that the system runs smoothly and efficiently.

Memory swapping between physical memory and the virtual memory file on disks takes a significant amount of time. It is important to have enough physical memory for running processes. Measure the status of an operating system with the operating system utilities. An extremely high page-swapping rate indicates that the amount of physical memory in a system is not large enough. In this case, you should remove any unnecessary processes or add more physical memory to the system.

9.2.1.2 Tuning DCCA Memory

The Database Communication and Control Area (DCCA) is a group of shared memory allocated by DBMaster servers. Every time DBMaster is started, it allocates and initializes the DCCA.

The DCCA is the resource most frequently accessed by DBMaster processes. It is important to ensure there is enough physical memory to prevent the operating system from swapping the DCCA to disks too often or it will seriously degrade performance of a database.

Usually a larger number of buffers are better for system performance. However, if the DCCA is too large to fit in physical memory, the system performance will degrade. Therefore, it is important to allocate enough memory for the DCCA but still fit the DCCA in physical memory.

You can set the appropriate parameters: **DB_NBufs**, **DB_NJnlB** and **DB_ScaSz** in **dmconfig.ini** before starting the database to configure the size of each of the DCCA components.

The total memory allocation for the DCCA is the sum of the size of **DB_NBufs**, **DB_NJnlB** and **DB_ScaSz**.

9.2.1.3 Tuning Page Buffer Cache

DBMaster uses the shared memory pool for the data page buffer cache. The buffer cache allows DBMaster to speed up data access and concurrency control. Adjusting the size of the page buffers will have the greatest effect on performance.

We can improve buffer cache performance by following ways

1. Update statistics on schema objects.
2. Set NOCACHE on large tables.
3. Reorganize data in poorly clustered indexes.
4. Enlarge cache buffers.
5. Reduce the effect of checkpoints.

For concrete realization of above methods please reference DBA manual Chapter “Performance Tuning”.

9.2.1.4 Tuning Journal Buffers

The journal buffers store the most recently used journal blocks. With enough journal buffers, the time required to write journal blocks to disks and roll back transactions when updating data and reading journal blocks from disks is reduced.

You should determine whether there are sufficient journal buffers for the system. The optimum number of journal buffers is the sum of journal blocks needed by the longest running transactions at the same time.

There are two ways used to estimate the number of journal buffers, one is the number of used journal blocks and the other measurement is the journal buffer flush rate.

More details please reference DBA manual Chapter “Performance Tuning”.

9.2.1.5 Tuning the SCA

Cache buffers and some control blocks, such as session and transaction information, have a fixed size, and are pre-allocated from the DCCA when a database is started. However, some concurrency control blocks are allocated dynamically from the DCCA while the database is running, their size is specified by DB_ScaSz.

If a database application gets the error message “database request shared memory exceeds database startup setting”, it means that DBMaster cannot dynamically allocate memory from the SCA area. Usually, this error is due to a long transaction using too many locks. If this situation happens often, solve it with the methods illustrated below.

1. Avoid Long Transactions

2. Avoid Excessive Locks on Large Tables

3. Increase the SCA size

For details please reference DBA manual Chapter “Performance Tuning”

9.2.2 QUERY OPTIMIZATION

The query optimizer will make a query of SQL commands much faster and efficient by means of choosing the best execution method internally.

If performance degrades, we should check the query plan by the command “Set dump plan on” and the SQL to improve the performance by forcing index scan, rewriting query, etc. For details please reference DBA manual Chapter “Performance Tuning”.

9.3 OS

A suitable OS is important for improving the performance of whole system, so please chose one OS with special designed for supporting the application disposal and the database as possible as you can.

In addition, about hard disks which support the technical Raid, please chose different Raid Level for different data types.For example, in DBMaster, you can put data file into Raid 1,3,5, and put journal file into Raid 0,which can guarantee safeness and a high efficiency.

9.4 Hardware

It is the basic factor not only affects the performance of DBMaster, but also affects the whole PC’s.

- **CPU:** A faster CPU or multi CPUs can help improving performance.
- **Memory:** Enough memory can hold more cached data, so I/O access time will be reduced.
- **I/O:** Faster hard disks can improve the I/O throughput and more hard disks can promote the I/O concurrency.
- **Network:** Speeding up transmission for network can reduce response time for users. Using only network protocols required will reduce load balancing of the operating systems.

  Obviously, enhancing the hardware can greatly improve the overall database system performance absolutely.

On the whole, we must rebuild indexes, adjust configuration according to DB and AP and so on which in order to improve the database application program performance. For more contents please refer to the DBA manual chapter “Performance Tuning”.

10. Appendix – Migration Samples

In this chapter, we will provide some real samples for both DBMaster and SQL Server. The content involves samples for not only some Table Schema and Data but also applications with different program languages. It provides a good demonstration of Migration from SQL Server to DBMaster.

The purpose is to help users quickly get to know the difference between DBMaster and SQL Server, and easily catch on the migration steps. It can reduce the migration costs.

In addition, these simple samples can not contain all of instances at present. And we will enhance all the features which the users care in this document continually.

10.1 Table Schema for all Types

In order to make users get to know Types Mapping between SQL Server and DBMaker, we give an example here. Users can write the SQL manually or generate the Script files automatically by ***JDatatransfer Tool*. (**We recommend using ***JDatatransfer tool*** because some Data Types in SQL Server are special and they are not so easily perceived or understood by users**)**.

In this section, we don't refer the migration of DATA, and we will demonstrate the samples for migration of ordinary types and special types data in next *chapter 10.2*.

10.1.1 Create Table with All Types in SQL Server

We create a table with all types which queried from SQL Server by "*select systypes.name from systypes*".

```
create table mssql_all_types(
  col_bigint           bigint,
  col_binary           binary(200),
  col_bit              bit,
  col_char             char(30),
  col_date             date,
  col_datetime         datetime,
  col_datetime2        datetime2(7),
  col_datetimeoffset   datetimeoffset(7),
  col_decimal          decimal(13, 3),
  col_float            float,
  col_geography        geography,
  col_geometry         geometry,
  col_hierarchyid      hierarchyid,
  col_image            image,
  col_int              int,
  col_money            money,
  col_nchar            nchar(40),
  col_ntext            ntext,
```

```
 col_numeric          numeric(8, 2),
 col_nvarchar         nvarchar(40),
 col_real             real,
 col_smalldatetime    smalldatetime,
 col_smallint         smallint,
 col_smallmoney       smallmoney,
 col_sql_variant      sql_variant,
 col_sysname          sysname,
 col_text             text,
 col_time             time(7),
 col_timestamp        timestamp,
 col_tinyint          tinyint,
 col_uniqueidentifier uniqueidentifier,
 col_varbinary        varbinary(200),
 col_varchar          varchar(30),
 col_xml              xml)
```

After the table being created, you can find the ***Default Value*** is ***NOT NULL*** for following two Types.

[col_sysname] [sysname] NOT NULL,

[col_timestamp] [timestamp] NOT NULL,

10.1.2 MIGRATE WITH JDATATRANSFER TOOL

If you *import from ODBC* with ***JDatatransfer Tool*** when the table "`mssql_all_types`" is not exist in DBMaster, you can choose ***Create destination table*** to migrate the Table Schema from SQL Server to DBMaster.

Certainly, you can modify the *Type, Precision, Nullable* for Destination Columns (Refer to 4.1.2.2 Execute steps Import from ODBC).

```
create table SYSADM.MSSQL_ALL_TYPES (
 COL_BIGINT           INTEGER          default null ,
 COL_BINARY           BINARY(200)      default null ,
 COL_BIT              SMALLINT         default null ,
 COL_CHAR             CHAR(30)         default null ,
 COL_DATE             DATE             default null ,
 COL_DATETIME         TIMESTAMP        default null ,
 COL_DATETIME2        TIMESTAMP        default null ,
 COL_DATETIMEOFFSET   CHAR(34)         default null ,
 COL_DECIMAL          DECIMAL(13, 3)   default null ,
 COL_FLOAT            REAL             default null ,
 COL_GEOGRAPHY        CHAR(1)          default null ,
 COL_GEOMETRY         CHAR(1)          default null ,
 COL_HIERARCHYID      CHAR(892)        default null ,
 COL_IMAGE            LONG VARBINARY   default null ,
 COL_INT              INTEGER          default null ,
 COL_MONEY            DECIMAL(19, 4)   default null ,
 COL_NCHAR            NCHAR(40)        default null ,
 COL_NTEXT            NCLOB            default null ,
 COL_NUMERIC          DECIMAL(8, 2)    default null ,
 COL_NVARCHAR         NVARCHAR(40)     default null ,
 COL_REAL             REAL             default null ,
 COL_SMALLDATETIME    TIMESTAMP        default null ,
```

```
COL_SMALLINT          SMALLINT         default null ,
COL_SMALLMONEY        DECIMAL(10, 4)   default null ,
COL_SQL_VARIANT       CHAR(10)         default null ,
COL_SYSNAME           NVARCHAR(128)    not null  ,
COL_TEXT              LONG VARCHAR     default null ,
COL_TIME              CHAR(16)         default null ,
COL_TIMESTAMP         BINARY(8)        not null  ,
COL_TINYINT           SMALLINT         default null ,
COL_UNIQUEIDENTIFIER  CHAR(36)         default null ,
COL_VARBINARY         BINARY(200)      default null ,
COL_VARCHAR           VARCHAR(30)      default null ,
COL_XML               CHAR(30)         default null )
in DEFTABLESPACE  lock mode row  fillfactor 100 ;
```

10.2 Table Schema and Data

In this section, we will divide all the Data Types into ***Ordinary Type*** and ***Special Type***.

The ***Ordinary Type*** data is ordinary characters and the numeric data type, which can be exported with *TEXT-Format* file from SQL Server via ***Import and Export Wizard***, and imported into DBMaster via ***Import from Text*** in ***JDataTransfer Tool*** (or via manual ***import*** command).

The ***Special Type*** data have different structures for different Databases, which must be converted by some built-in functions or ODBC Applications. For example: via ***Import from ODBC*** in ***JDataTransfer Tool***.

10.2.1 ORDINARY CHARACTER AND NUMERIC DATA TYPE

Step 1: Create table *ordinary_types* in Oracle.

```
create table ordinary_types(
  col_bigint          bigint,
  col_char            char(30),
  col_date            date,
  col_datetime        datetime,
  col_datetime2       datetime2(7),
  col_datetimeoffset  datetimeoffset(7),
  col_decimal         decimal(13, 3),
  col_float           float,
  col_int             int,
  col_money           money,
  col_numeric         numeric(8, 2),
  col_real            real,
  col_smalldatetime   smalldatetime,
  col_smallint        smallint,
  col_smallmoney      smallmoney,
  col_time            time(7),
  col_tinyint         tinyint,
  col_varchar         varchar(30));
```

Step 2: Insert Data by some Applications or by hand.

For example:

```
insert into ordinary_types values(10000,'COL_CHAR CHAR(30) null','2006-04-25','2006-04-25 10:23:30.123','2006-04-25 10:23:30.1234567','2007-05-08 12:35:29.1234567 +12:15',4359305.23,9849.34,1000,8888.88,12345.23,9849.34,'2006-04-25 10:23:30',10,83534.348,'23:24:51.1234567',10,'COL_VARCHAR VARCHAR(30) null');

insert into ordinary_types values(20000,'COL_CHAR CHAR(30) null','2006-04-26','2006-04-25 10:23:30.123','2006-04-25 10:23:30.1234567','2007-05-08 12:35:29.1234567 +12:15',4359305.23,9849.34,1000,8888.88,12345.23,9849.34,'2006-04-25 10:23:30',10,83534.348,'23:24:51.1234567',10,'COL_VARCHAR VARCHAR(30) null');

insert into ordinary_types values(30000,'COL_CHAR CHAR(30) null','2006-04-27','2006-04-25 10:23:30.123','2006-04-25 10:23:30.1234567','2007-05-08 12:35:29.1234567 +12:15',4359305.23,9849.34,1000,8888.88,12345.23,9849.34,'2006-04-25 10:23:30',10,83534.348,'23:24:51.1234567',10,'COL_VARCHAR VARCHAR(30) null');
```

Step 3: Create table *ordinary_types* in DBMaster.

You can create table manually or *Import from ODBC* (choose ***Create destination table*** and modify the *Type, Precision, Nullable* for Destination Columns) with ***JDataTransfer Tool***.

In addition, the Type Mapping may be different between the new version and the old version of ***JDataTransfer Tool***. Especially for Type NCHAR, NVARCHAR, DATE, DATETIME, DATETIME2, DATETIMEOFFSET, SMALLDATETIME, TIME, TIMESTAMP. Please take the following example for reference.

```
create table ordinary_types(
 col_bigint           integer,
 col_char             char(30),
 col_date             date,
 col_datetime         char(23),
 col_datetime2        char(27),
 col_datetimeoffset   char(34),
 col_decimal          decimal(13, 3),
 col_float            real,
 col_int              integer,
 col_money            decimal(19, 4),
 col_numeric          decimal(8, 2),
 col_real             real,
 col_smalldatetime    timestamp,
 col_smallint         smallint,
 col_smallmoney       decimal(10, 4),
 col_time             char(16),
 col_tinyint          smallint,
 col_varchar          varchar(30));
```

Step 4: Export Data separately from SQL Server.

Please use ***Import and Export Wizard*** in SQL Server (refer to *Chapter 4.1.1*), and choose the ***Flat File destination***. Operation steps as following:

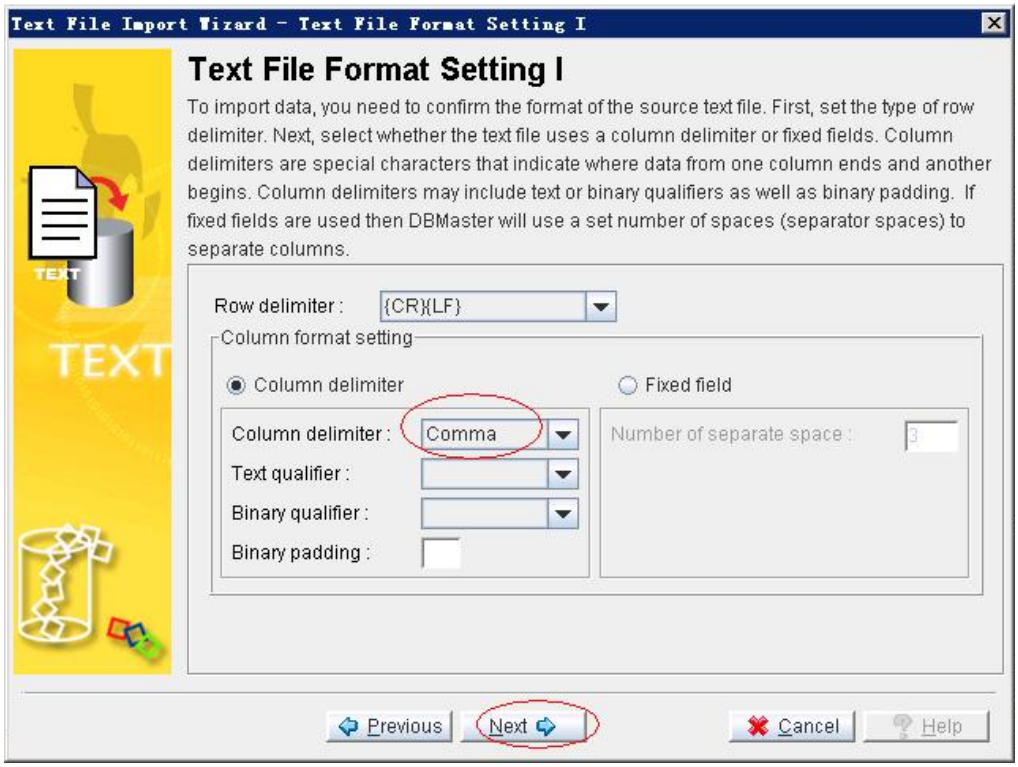
Input ***File Name*** (`c:\test\ordinary_types.txt`) → click ***Next*** → click ***Next*** → choose ***Source Table or View*** → click ***Next*** → click ***Next*** → ***Finished***.

Step 5: Import Data into DBMaster.

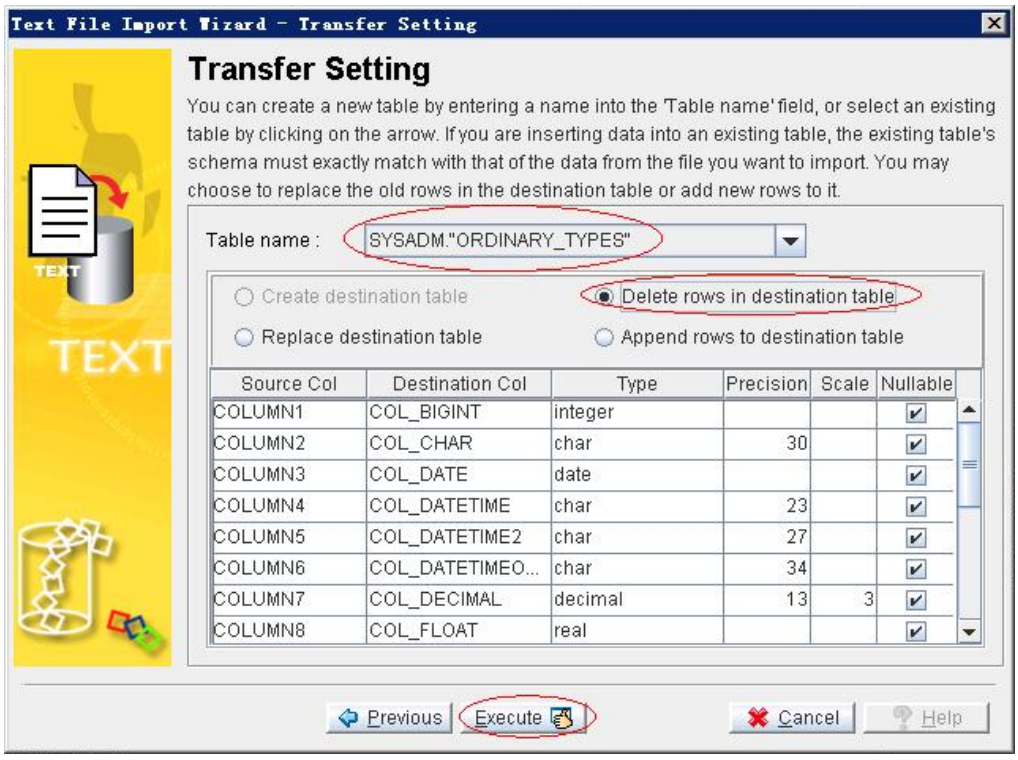
Please use ***Import from Text*** with ***JDataTransfer Tool*** and you must choose the uniform separator character to export TEXT format data. For example: ***Comma*** (***Semicolon*** by default) for *Column Delimiter*, {CR}{LF} for *Row Delimiter*. Please take following steps and Charts for reference.

Open ***Import from Text*** →click ***Next*** → Input ***Text file*** (`c:\test\ordinary_types.txt`) → click ***Next*** → (***Chart 1***) → click ***Next*** → click ***Next*** → choose ***Database***, ***User Name*** and ***Password*** → click ***Next*** →(***Chart 2***).

(***Chart 1***)



(***Chart 2***)



In addition, you can also use the *IMPORT* command in dmSQL tools as following:

```
dmSQL> import ordinary_types from c:\test\ordinary_types.txt description c:\test\desc.txt;
```

desc.txt

```
FORMAT=VARIABLE
COLUMN_DELIMITER=',
ROW_TERMINATOR="\r\n"
```

Step 6: Check the Data in DBMaster.

```
dmSQL> def table ordinary_types;
dmSQL> select * from ordinary_types;
```

10.2.2 SPECIAL DATA TYPE

Step 1: Create table *special_types* in SQL Server.

```
create table special_types(
 col_binary           binary(200),
 col_bit              bit,
 col_geography        geography,
 col_geometry         geometry,
 col_hierarchyid      hierarchyid,
 col_image            image,
 col_nchar            nchar(40),
 col_ntext            ntext,
 col_nvarchar         nvarchar(40),
 col_sql_variant      sql_variant,
 col_sysname          sysname,
 col_text             text,
 col_timestamp        timestamp,
 col_uniqueidentifier uniqueidentifier,
 col_varbinary        varbinary(200),
 col_xml              xml);
```

Step 2: Insert Data by some Applications or by hand.

For example:

```
insert into special_types values(cast('4142434445464748494a'as binary(50)),10,null,null,CAST('/1/1/' AS
hierarchyid),cast('4142434445464748494a' as
binary(50)),'4142434445464748494a','4142434445464748494a','4142434445464748494a','IANTCHAR10','4142434445464748494
a','COL_TEXT LONG VARCHAR default null',null,'6F9619FF-8B86-D011-B42D-00C04FC964FF',cast('4142434445464748494a' as
binary(50)),'COL_XML CHAR(30) default null');

insert into special_types values(cast('4142434445464748494a'as binary(50)),10,null,null,CAST('/1/1/' AS
hierarchyid),cast('4142434445464748494a' as
binary(50)),'4142434445464748494a','4142434445464748494a','4142434445464748494a','IANTCHAR10','4142434445464748494
a','COL_TEXT LONG VARCHAR default null',null,'6F9619FF-8B86-D011-B42D-00C04FC964FF',cast('4142434445464748494a' as
binary(50)),'COL_XML CHAR(30) default null');

insert into special_types values(cast('4142434445464748494a'as binary(50)),10,null,null,CAST('/1/1/' AS
hierarchyid),cast('4142434445464748494a' as
binary(50)),'4142434445464748494a','4142434445464748494a','4142434445464748494a','IANTCHAR10','4142434445464748494
a','COL_TEXT LONG VARCHAR default null',null,'6F9619FF-8B86-D011-B42D-00C04FC964FF',cast('4142434445464748494a' as
binary(50)),'COL_XML CHAR(30) default null');
```

Note: It's hard to demonstrate the migration for type GEOGRAPHY and GEOMETRY, and we only insert *NULL* values.

Step 3: Create Table in DBMaster manually or ***Export from ODBC***.

You can only export table schema from ODBC via JDataTransfer Tool, Please refer to 4.1.2.2 Execute steps Import from ODBC which include choosing ***Create destination table*** and modifying the *Type, Precision, Nullable* for Destination Columns.

```
create table special_types(
 col_binary           binary(200),
 col_bit              smallint,
 col_geography        char(1),
 col_geometry         char(1),
 col_hierarchyid      char(892),
 col_image            long varbinary,
 col_nchar            nchar(40),
```

```
col_ntext            nclob,
col_nvarchar         nvarchar(40),
col_sql_variant      char(10),
col_sysname          nvarchar(128)  not null,
col_text             long varchar,
col_timestamp        binary(8)      not null,
col_uniqueidentifier char(36),
col_varbinary        binary(200),
col_xml              char(30));
```

Note: *Please modify the **Default Value** to **NOT NULL** for column COL_SYSNAME and column COL_TIMESTAMP.*

Step 4: Import the special type Data into DBMaster.

Please use ***Import from ODBC*** with ***JDataTransfer Tool*** (refer to 4.1.2.2 Execute steps Import from ODBC).

10.3 Applications (Source Code segment)

We provide some parts of Source Code segments in this section. And the issue is focusing mainly on the different usage of ***Connection*** between DBMaster and SQL Server.

In addition, we will demonstrate the different usage of ***placeholder*** in JAVA and C# Language Samples. The placeholder in JAVA is "**?**" when users pass parameters; the placeholder in C# is same "**?**" for DBMaster, and is "@xxxxx" for SQL Server (*SQL Server Data Provider for .NET*).

10.3.1 JAVA LANGUAGE

- SQL Server

```
try{
    ......
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver").newInstance();
    Connection conn =
DriverManager.getConnection("jdbc:sqlserver://192.168.0.8:1433;DatabaseName=testdb","sa","pw123");
    PreparedStatement pstmt = conn.prepareStatement("insert into ordinary_types(col_varchar, col_int,
col_float) values(?,?,?)");
    pstmt.setString(1, "varchar-abcbc");
    pstmt.setInt(2, 1000);
    pstmt.setFloat(3, (float)32322555.3332);
    pstmt.executeUpdate();
    ......
    }
}catch(Exception ex){
    ex.printStackTrace();
}
```

- DBMaster

```
try{
    ......
    Class.forName("dbmaster.sql.JdbcOdbcDriver").newInstance();
    Connection conn = DriverManager.getConnection("jdbc:dbmaster:testdb","SYSADM","test123");
    PreparedStatement pstmt = conn.prepareStatement("insert into ordinary_types(col_varchar, col_int,
col_float) values(?,?,?)");
    pstmt.setString(1, "varchar-abcbc");
```

```
    pstmt.setInt(2, 1000);
    pstmt.setFloat(3, (float)32322555.3332);
    pstmt.executeUpdate();
    ......
    }
}catch(Exception ex){
    ex.printStackTrace();
}
```

Note: DBMaster only supports JDBC Type2 at present, users need to install native DLL for JDBC Driver. In addition, don't forget to set IP and Port in Dmconfig.ini.

10.3.2 C# LANGUAGE

- SQL Server (*SQL Server Data Provider for .NET*)

```
String connStr = "Data Source=localhost;Initial Catalog=testdb;User Id=sa;Password=pwd123;";
SqlConnection conn = new SqlConnection(connStr);
conn.Open();
SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "insert into ordinary_types(col int, col char, col numeric) values(@p1,@p2,@p3)";
cmd.Parameters.Add(new SqlParameter("@p1",SqlDbType.Int));
cmd.Parameters.Add(new SqlParameter("@p2",SqlDbType.Char, 30));
cmd.Parameters.Add(new SqlParameter("@p3",SqlDbType.Decimal));
cmd.Parameters["@p1"].Value = 1001;
cmd.Parameters["@p2"].Value = "Li Ping";
cmd.Parameters["@p3"].Value = 2345.34;
cmd.ExecuteNonQuery();
cmd.Parameters.Clear();
cmd.CommandText = "select col_int, col_char, col_numeric from ordinary_types where col_int=1001";
cmd.CommandType = CommandType.Text;
SqlDataReader reader = cmd.ExecuteReader();
while(reader.Read()){
   for(int i=0;i<reader.FieldCount;i++){
      Console.WriteLine(reader[i]);
   }
}
conn.Close();
```

- DBMaster (*ADO.NET ODBC Provider*)

```
String connStr = "Driver={DBMaster 5.1 Driver}; Database=testdb; Uid=SYSADM; Pwd=test123";
OdbcConnection conn = new OdbcConnection(connStr);
conn.Open();
OdbcCommand cmd = new OdbcCommand();
cmd.Connection = conn;
cmd.CommandText = "insert into ordinary_types(col int, col char, col numeric) values(?,?,?)";
cmd.Parameters.Add(new OdbcParameter("p1",OdbcType.Int));
cmd.Parameters.Add(new OdbcParameter("p2",OdbcType.Char, 30));
cmd.Parameters.Add(new OdbcParameter("p3",OdbcType.Numeric));
cmd.Parameters[0].Value = 1001;
cmd.Parameters[1].Value = "Li Ping";
cmd.Parameters[2].Value = 2345.34;
```

```
cmd.ExecuteNonQuery();
cmd.Parameters.Clear();
cmd.CommandText = "select col_int, col_char, col_numeric from ordinary_types where col_int=1001";
OdbcDataReader reader = cmd.ExecuteReader();
while(reader.Read()){
   for(int i=0;i<reader.FieldCount;i++){
      Console.WriteLine(reader[i]);
   }
}
conn.Close();
```

Note: We don't provide the ***.NET Provider*** at present, so we only can use ***ADO.NET ODBC Provider*** or ***ADO.NET OLEDB Provider*** to connect DBMaster (The following *Source Code segment* is for *ADO.NET OLEDB Provider*).

```
String connStr = "Provider=DMOLE51; Data Source=testdb; User Id=SYSADM; Password=test123";
OleDbConnection conn = new OleDbConnection(connStr);
conn.Open();
OleDbCommand cmd = new OleDbCommand();
cmd.Connection = conn;
cmd.CommandText = "insert into ordinary_types(col int, col char, col numeric) values(?,?,?)";
cmd.Parameters.Add(new OleDbParameter("p1",OleDbType.Integer));
cmd.Parameters.Add(new OleDbParameter("p2",OleDbType.Char, 30));
cmd.Parameters.Add(new OleDbParameter("p3",OleDbType.Numeric));
cmd.Parameters[0].Value = 1001;
cmd.Parameters[1].Value = "Li Ping";
cmd.Parameters[2].Value = 2345.34;
cmd.ExecuteNonQuery();
cmd.Parameters.Clear();
cmd.CommandText = "select col_int, col_char, col_numeric from ordinary_types where col_int=1001";
OleDbDataReader reader = cmd.ExecuteReader();
while(reader.Read()){
   for(int i=0;i<reader.FieldCount;i++){
      Console.WriteLine(reader[i]);
   }
}
conn.Close();
```

10.3.3 PHP LANGUAGE

We demonstrate the PHP PDO samples. If users don't adopt PDO, please refer to our PHP samples in Installed Directory which use the PHP ODBC API.

- SQL Server

```
<?php
try{
   $dbh = new PDO ("mssql:host=localhost;dbname=testdb","as","pwd123");
   /*** echo a message saying we have connected ***/
   echo 'Connected to database';
}catch(PDOException $e){
   echo $e->getMessage();
}
?>
```

- DBMaster

```
<?php
 try{
    $dbh = new PDO("odbc:Driver={DBMaster 5.1 Driver};Database=testdb", "sysadm", "test123");
    /*** echo a message saying we have connected ***/
    echo 'Connected to database';
 }catch(PDOException $e){
    echo $e->getMessage();
 }
?>
```

Note: If users didn't use the PDO in SQL Server as following:

```
<?php
$dbhandle =mssql_connect("localhost","sa","pwd123") or die("Couldn't connect to SQL Server ");
$dbselected=mssql_select_db("testdb", $dbhandle) or die("Couldn't open database");
$result = mssql_query("select sysobjects.name from sysobjects");
$numRows = mssql_num_rows($result);

?>
```

Please use PHP ODBC API for DBMaster.

```
<?php
$conn=odbc_connect("dbsample5","SYSADM","");
$rs_count=odbc_exec($conn,"select count(*) from SYSUSER");
?>
```