US 20030177146A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0177146 A1

Zimowski (43) Pub. Date: Sep. 18, 2003

(54) METHOD, SYSTEM, AND PROGRAM PRODUCT FOR MIGRATING DATA FROM ONE DATA BASE MANAGEMENT SYSTEM TO ANOTHER DATA BASE MANAGEMENT SYSTEM

(75) Inventor: **Melvin Richard Zimowski**, San Jose, CA (US)

Correspondence Address:
INTERNATIONAL BUSINESS MACHINES CORP
IP LAW
555 BAILEY AVENUE , J46/G4
SAN JOSE, CA 95141 (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY

(21) Appl. No.: **10/102,385**

(22) Filed: **Mar. 18, 2002**

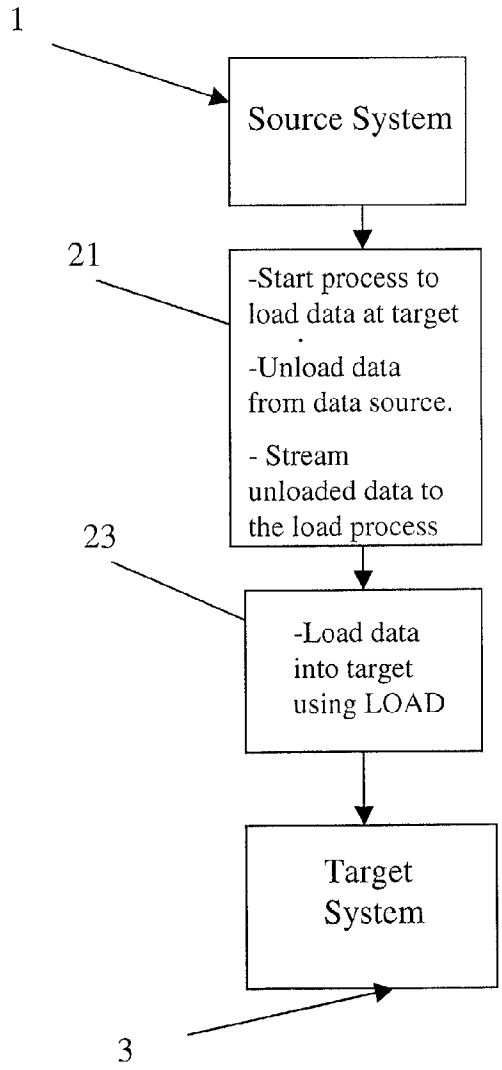
Publication Classification

(51) **Int. Cl.**[7] **G06F 17/30**

(52) **U.S. Cl.** **707/200**

(57) **ABSTRACT**

A method, system, and program product to migrate data from a data table in a data source to a data table in a target. This is done initiating processes, typically two parallel batch processes, with one of the processes unloading the data from the data source and uploading the data into the data target, and the other of the processes loading the source data into a database at the data target. Preferably the processes are parallel batch process. The processes are initiated from the data target.



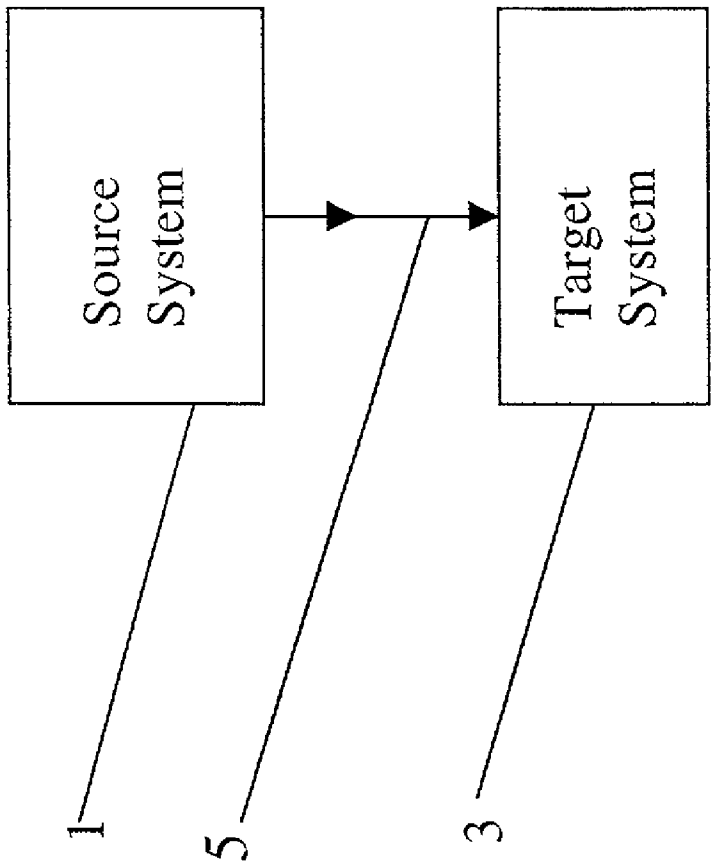
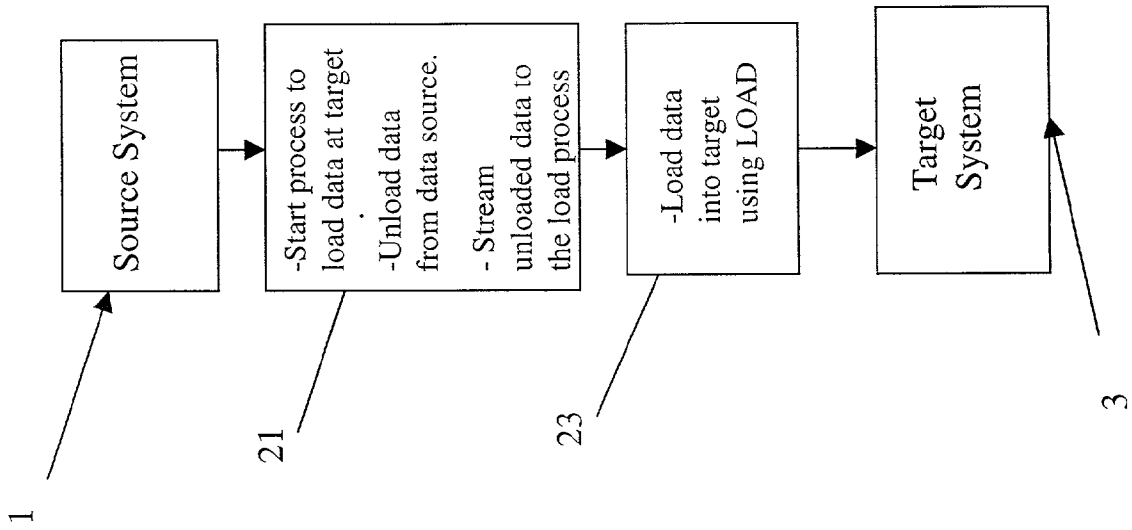


FIGURE 1

Figure 2


Source System
-Start process to load data at target
-Unload data from data source.
- Stream unloaded data to the load process
-Load data into target using LOAD
Target System
1
21
23
3

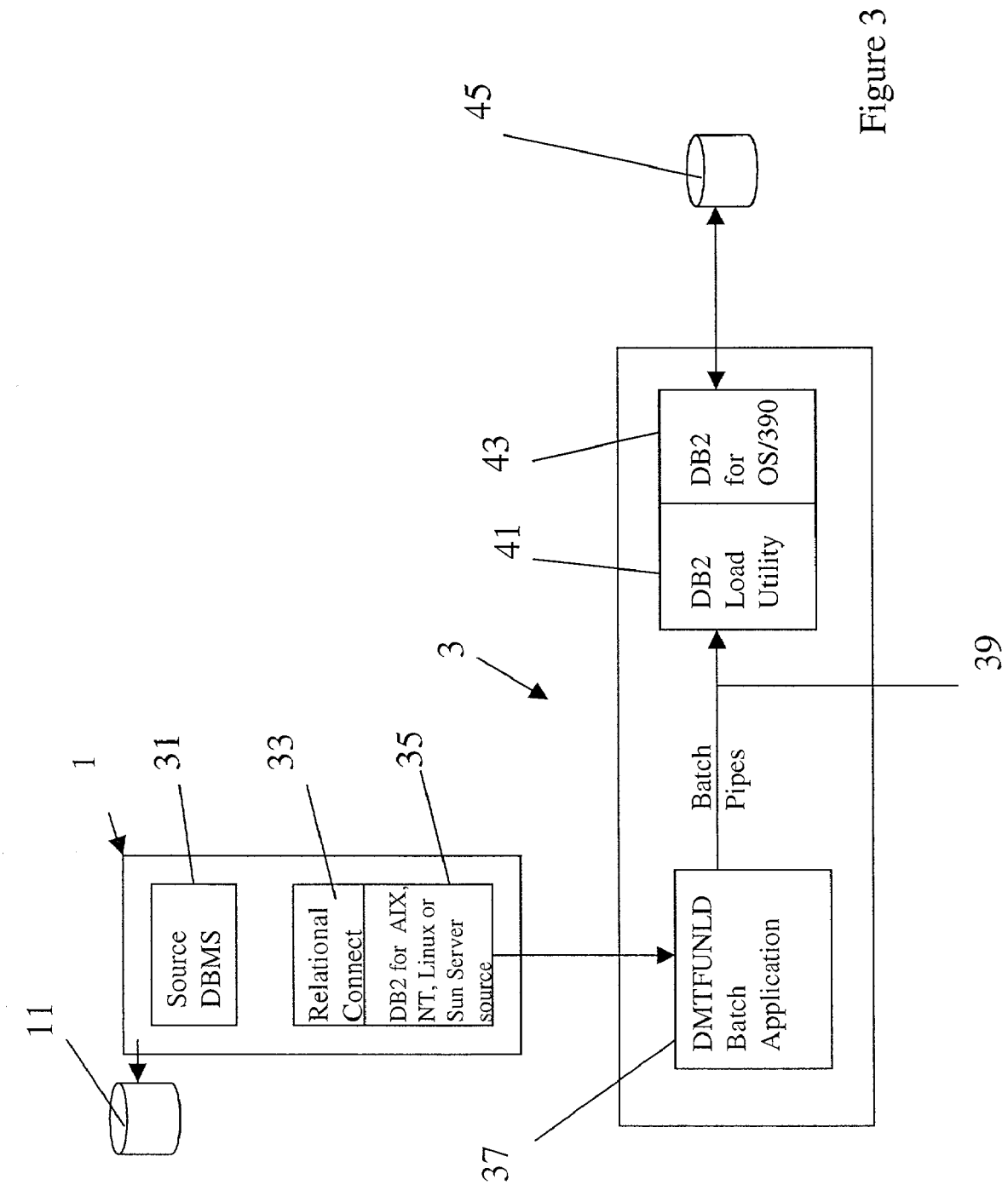

45
43
41
3
1
31
33
35
11
39
37
Source DBMS
Relational Connect
DB2 for AIX, NT, Linux or Sun Server source
DB2 for OS/390
DB2 Load Utility
Batch Pipes
DMTFUNLD Batch Application

Figure 3

METHOD, SYSTEM, AND PROGRAM PRODUCT FOR MIGRATING DATA FROM ONE DATA BASE MANAGEMENT SYSTEM TO ANOTHER DATA BASE MANAGEMENT SYSTEM

FIELD OF INVENTION

[0001] The method, system, and program product described herein are used for migrating large volumes of data from a data source to a data target, and include organizing and interrelating data or files, e.g., providing an association between different schema and metadata, with, for example, data structure conversion, for optimization of database and file migration and storage, and for data compatibility between different or multiple databases.

BACKGROUND OF THE INVENTION

[0002] Current approaches for migrating large quantities (for example, gigabytes) of data from relational data stores on various database platform sources, such as Windows NT, AIX, Linux, Solaris, and other database platforms, to databases on disparate target platforms, such as IBM "DB2 UDB for OS/390," do not perform well and are overly complex. Poor performance and complexity can be attributed to a variety of causes including (1) using SQL in a multi-step process to both extract the data from the source data store and place the data into the target data store, (2) storing the data in one or more intermediate data stores (e.g. local file systems) during data migration, (3) performing multiple datatype conversions as the data is moved between environments, (4) using less efficient and inefficient data transfer mechanisms, and (5) using data migration processes that are not fully automated.

SUMMARY OF THE INVENTION

[0003] The method, system, and program product described herein use bulk data loading interfaces to migrate the source data into a target data store. The method, system, and program product of the invention eliminate the use of intermediate data stores on the target system, execute the extraction of the data from the source data store and the placement of the data into the target data store in parallel, minimize the datatype conversions that are performed, transfer the data in large blocks, and perform the migration in a fully automated fashion. This approach reduces the complexity and the elapsed time required for performing these types of data migrations.

[0004] Specifically, the method, system, and program product migrates data from a relational table in a data source to a relational table in a data target. This is accomplished through the use of two parallel batch processes. One of these batch processes unloads the subject data from the data source and uploads the subject data to the data target. The other batch process loads the subject data into the data target. The "unloading" batch process operating on the data source may use an SQL "SELECT * FROM tablename" command to identify the data to be unloaded, where "tablename" is the name of the source relational table. The "loading" batch process operating on the data target may use a LOAD Utility or other bulk data loading mechanism to load the source data into the data target.

[0005] The unload of a particular database table at the source and the subsequent load of the database table at the target execute in parallel. The method, system, and program product allow for the parallel unload and load of the database table by eliminating the use of intermediate data files between the two batch processes. Data flows directly from the unload batch process to the load batch process, thereby reducing both the elapsed time and the complexity of the data migration. The method, system, and program product of the invention also permit a write of the data being migrated to disk on the target system, in parallel with the load of the database table. This copy allows the load of the data to be restartable from the last internal load utility checkpoint in the event of a load utility failure.

[0006] The method, system, and program product may be used to facilitate migration of disparate source database data, such as Oracle, Sybase, or Microsoft SQL Server, to, for example, IBM "DB2 for OS/390." In some situations data type conversion may be required. For example, Oracle long data may need to be mapped to IBM UDB DB2 varchar data.

[0007] Overall, according to the method, system, and program product of the invention, the complexity of generating SQL view mapping structures, generating batch jobs and coordinating and executing the migration task is reduced due to automated processes and tools.

[0008] The method, system, and program product of the invention are intended for use in both peer to peer and client server environments.

[0009] The program product of the invention is in the form of computer code on suitable media (such as magnetic or optical media that is machine readable for transmission, installation, instantiation, or execution). The code may be in the form of executable code, in the form of code that requires loading to memory for execution, or (physical or electronic) distribution code that requires one or both of decompression and decryption to be executable.

THE FIGURES

[0010] Aspects of the invention are illustrated in the FIGURES.

[0011] **FIG. 1** illustrates a system with two platforms configured for transferring data between the source and the target through a data link.

[0012] **FIG. 2** illustrates the method of transferring data from a data source to a data target, utilizing two target side steps in copying data from the source to the target.

[0013] **FIG. 3** is a high level illustration of software modules that comprise a preferred embodiment of the method, system, and program product of the invention.

DESCRIPTION OF THE INVENTION

[0014] The method, system, and program product described herein provides for the migration of relational tables from a data source to a data target. This is accomplished through the unloading of a particular database table at the data source and subsequent loading of the database table at a data target. Through the use of a data pipe between the parallel batch processes for (1) unloading the source data and (2) loading the target data, the data being migrated never needs an intermediate write to disk on the target, thereby reducing both the elapsed time and the complexity of the data migration.

[0015] **FIG. 1** illustrates a source-target system configured for transferring data between the data source, **1**, and the data target, **3**, through a data link, **5**. In the data source, **1**, SQL identifies the data to be extracted from the source system. The target system initiates the unload of the data at the source system, receives the data, and then loads it into the data target on the target system. It is understood that in a multi-data platform system, there can be more then one data source, and also more then one data target, and that data transfers between the data sources and the data targets may involve data type conversions. Previously, as described above, this was at best, a partially automated and inefficient data transfer system.

[0016] **FIG. 2** illustrates the two data target side steps, **21** and **23**, used to migrate data from the data source, **1**, to the data target, **3**. In the first data target side step, **21**, the target system, **3**, starts the process that will load the data in the second step, uses an unload process to issue an SQL request to unload selective data from the source system, uploads the unloaded data and streams the data to the second step, **23**, which, as mentioned earlier, comprises a process that loads the data into the target system.

[0017] The two steps, **21** and **23**, in **FIG. 2**, are actually two batch jobs executing on the target system. The first batch job, **21**, connects to the source system, issues an SQL statement to retrieve data from a source system table, uploads the selected data, and streams the data to the second batch job, **23** which loads the data into the target system using the DB2 UDB for z/OS Load utility.

[0018] **FIG. 3** is a high level illustration of the software modules used to implement the method and system of the invention. The source system, **1**, includes source data, **11**, a source database management system, **31**, for example, a relational database management system, and potentially a middleware product, such as DB2 UDB Relational connect, which allows access to various data sources.

[0019] The target system, **3**, is associated with a database management system, **43**, preferably a relational database management system, an associated database, **45**, preferably a relational database, application code to unload and upload data from the data source, and a utility program to load data into the source target.

[0020] As illustrated in **FIG. 3**, there is substantial code execution on the target system, **3**. The target system, **3**, initiates data transfer from source to target, for example, an IBM DB2 UDB target or similar target, with the application, DMTFUNLD, **37**, which both (1) unloads data from the source table and (2) uploads the data to the target system, **3**, using, for example, IBM's Distributed Relational Database Architecture (DRDA) protocol.

[0021] DMTFUNLD identifies the data to be unloaded through the execution of an SQL statement. Through the use of DRDA, the SQL statement can be executed against a remote DBMS source, and through the use of DB2 UDB Relational Connect, the remote DBMS source can be one of a variety of DBMS products. The SQL statement issued to unload data from the source DBMS is of the form SELECT * FROM tablename. Once DMTFUNLD issues the SQL statement, the data is uploaded to the target system using DRDA blocking.

[0022] DMTFUNLD drives the unload and load processes and executes as a batch program.

[0023] DMTFUNLD has the psuedocode representation:

[0024] Open the SYSIN dataset and the SYSP UNCH pipe

[0025] Read table name from SYSIN dataset

[0026] Build the SQL statement to be executed: SELECT * FROM <table_name>

[0027] Build the LOAD Utility control statement for the load of the data and write it to

[0028] SYSPUNCH

[0029] Close the SYSPUNCH pipe

[0030] Open the SYSREC pipe

[0031] Execute the SELECT statement to retrieve and upload the data to be migrated and write the data to SYSREC

[0032] Close the SYSREC pipe

[0033] Close the SYSIN dataset

[0034] As illustrated in **FIG. 3**, this batch application unloads the database table from the data source, **11**, using middleware, such as Relational Connect, **33**, and a relational database management system, such as IBM DB2 UDB, **35**. Unloading is initiated using the SQL expression

SELECT * FROM tablename.

[0035] The method, system, and program product upload the selected data to the data target using Distributed Relational Database Architecture (DRDA) flows, and then generate the LOAD control statement for the load of the extracted table from the source system, **1**, into the target system, **3**. The load control statement and the unloaded data are passed from DMTFUNLD, **37**, to the DB2 LOAD utility, **41**, using batch pipes, **39**. To be noted is that the first, **21**, and second, **23**, batch jobs execute in parallel. The use of batch pipes, **39**, permits (i) upload of data from the source system, **1**, and (ii) loading of data into the target system, **3**, to be substantially simultaneous. The use of batch pipes, **39**, also reduces I/O operations and the use of Direct Access Storage Devices (DASD) by transferring data through processor storage rather then through DASD.

[0036] Two batch jobs are required for each table to be migrated. The first batch job, **21**, performs the unload of the data from one or more tables of the source, **1**, and uploads the extracted data to the DB2 for OS/390 server, **43**, on the target, **3**. For each table, a second batch job performs the load of the data into a DB2 for OS/390 table using the LOAD Utility, **41**. The second batch job for each table, **23**, is automatically submitted for execution by the first batch job, **21**. The first, **21**, and second, **23**, batch jobs for an individual table execute in parallel. In addition, multiple first batch jobs can execute in parallel. The methodology also includes a process for generating the Job Control Language (JCL) for the batch jobs.

[0037] The unload environment on the source data platform, **1**, requires special setup and configuration. The system, method, and program product include a process for accomplishing this prior to the execution of the batch jobs that migrate the tables.

[0038] The data being migrated always resides in memory and never needs to be written to an intermediate file or any other type of intermediate data store on the target machine, e.g., an IBM OS/390 machine. Source to target datatype conversions, for example, Oracle to DB2 datatype conversions, occur once, at the time the data is unloaded from the source database. The DRDA protocol is used to transfer the data.

[0039] The method, system, and program product of the invention also has the capability to create a copy of the data being loaded—in parallel with the execution of the extraction from the source and the load to the target. This copy is useful in situations where the amount of data being migrated is large and the LOAD Utility, **41**, for whatever reason, fails to execute to completion, as the copy can be used to restart the load of the data from the last internal LOAD Utility checkpoint rather than from the start of the load. When used, this optional facility does create a copy of the data in an intermediate data store, the local target file system, for example, an IBM OS/390 file system.

[0040] The data migration process is initiated by the batch application (DMTFUNLD), **37**, that executes under the target's operating system, for example IBM z/OS. Each invocation of the application:

[0041] (1) unloads a table from a source database, **11**, using, for example, IBM DB2 UDB for AIX with Relational Connect by executing SELECT * FROM tablename,

[0042] (2) uploads the table to the target machine using DRDA flows

[0043] (3) generates the target machine's LOAD Utility, **41**, control statement for the subsequent load of the table into the target machine's database management system, for example, IBM DB2 UDB for OS/390, **43**.

[0044] The LOAD Utility control statement has the form:

[0045] LOAD DATA LOG NO INDDN ddname ENFORCE NO RESUME YES

[0046] SORTKEYS 150000000 INTO TABLE into_table_specification

[0047] The batch application passes the LOAD Utility control statement and the data to the target's database management system's LOAD utility using Batch Pipes, **39**. The use of Batch Pipes:

[0048] (1) permits the upload and the load of the data to proceed in parallel

[0049] (2) reduces the number of I/O operations and the use of DASD by transferring data through processor storage rather than to and from DASD

[0050] The bulk data migration method, system, and program product makes the following assumptions about the source and target schemas:

[0051] (1) The source DBMS system and target DBMS system contain identical tables or the source DBMS tables can be mapped to the target DBMS tables using a mechanism such as a relational view.

[0052] (2) The source and target tables have the same table and column names.

[0053] (3) The datatypes associated with columns of the source and target tables are identical or the datatypes of the source table will be replaced with equivalent target datatypes.

[0054] These assumptions permit the existing middleware applications to access the tables that have been migrated to the target system, for example, DB2 for OS/390 tables, and preserve the referential integrity relationships that are introduced and managed by those middleware applications.

EXAMPLE

[0055] This example illustrates the creation, modification, and execution of various commands, jobs, and SQL statements used to establish and configure the execution environment for the data migration tool.

[0056] The execution environment had the following characteristics:

[0057] 1. OS/390 Database Server: stplex4a.stl.ibm.com

[0058] 2. DB2 for OS/390 Subsystem: SBL1 (DB2 for OS/390 V6)

[0059] 3. Source library for DMTFUNLD: USRNDO1.SBL1.SDSNSAMP

[0060] 4. JCL library: USRNDO1.SBL1.JCL

[0061] 5. REXX EXEC library: USRNDO1.SBL1.REXX

[0062] 6. AIX Database Server: yoda13_tr0.stl.ibm.com

[0063] 7. DB2 UDB for AIX database: DMTDB271 (DB2 for UDB V7.1 fix pack 2)

[0064] 8. Oracle Server: orac8i (Oracle Version 8.1.6 using Net8)

[0065] The data migration tool used the federated database support provided by DB2 UDB V7 and the federated database support for Oracle data sources provided by DB2 Relational Connect. DB2 UDB and Relational Connect were configured for access to Oracle data. The steps included installing Relational Connect and adding an Oracle data source to a federated system, as described in "Setting Up a Federated System to Access Oracle Data Sources" in the chapter "Configuring a Federated Database System" of the DB2 UDB and DB2 Connect Installation and Configuration Supplement.

[0066] In order to add Oracle data sources to the federated system, the following steps were performed:

[0067] Step 1: Install and configure the Oracle client software on the DB2 federated server using the documentation provided by Oracle. The documentation for this example was from Oracle Version 8.1.6 and Net8.

[0068] Step 2: Set data source environment variables by modifying the DB2 DJ.ini file and issuing the db2set command. The db2set command updates the DB2 profile registry with the specified settings. The db2dj.ini file was edited to set the ORACLE_HOME, ORACLE_BASE, ORA_NLS, and TNS_ADMIN environment variables, as shown in the Appendix.

[0069] Step 3: Insure that the SQL*Net or Net8 tnsnames.ora file is updated for each Oracle server to which communications are configured, and update the tnsnames.ora file for the server yoda13_tr0.stl.ibm.com.

[0070] Step 4: Recycle the DB2 instance:

[0071] Step 5: Use the CREATE WRAPPER statement to define the wrapper library that will be used to access Oracle data sources. Wrappers are the mechanism that federated servers use to communicate with and retrieve data from data sources.

[0072] Step 6: Set the DB2_DJ_COMM environment variable to include the wrapper library that corresponds to the wrapper module created in the previous step.

[0073] Step 7: Next, use the CREATE SERVER statement to define each Oracle server to which communications are configured. We created the Oracle server orac8i.

[0074] Step 8: If a user ID or password at the federated server was different from a user ID or password at an Oracle data source, use the CREATE USER MAPPING statement to map the local user ID to the user ID and password defined at the Oracle data source. We used the Oracle user ID SYS to access the Oracle data on server orac8i. SYS is one of the DBA users that is created when a database is installed and initialized. The DBA role has all system privileges and the ability to grant all privileges to other users. The Oracle user SYS owns all base tables and user-accessible views of the data dictionary.

[0075] Step 9: For Oracle tables in the Oracle data source that do not contain long columns, use the CREATE NICKNAME statement to assign a nickname to the table. Note that the nickname must be the same as the table name specified for the DB2 for OS/390 target table.

[0076] Note: Due to schema restrictions, for the tables migrated in this example, Oracle Long data types had to be cast to DB2 UDB for z/OS varchar(length) data types, where length was the max actual length of the data contained in the Oracle Long column.

[0077] Step 10. For Oracle tables in an Oracle data source that do contain long columns, use the CREATE NICKNAME statement to assign a nickname to each table located in the Oracle data source and the CREATE VIEW statement to cast the datatype of each long column to varchar. Note that the view name must be the same as the table name specified for the DB2 for OS/390 target table.

[0078] Next, it was necessary to modify the configuration for the DB2 UDB Database.

[0079] Step 11: Increase the APPLHEAPSZ configuration parameter setting to 1024. The APPLHEAPSZ configuration parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent. The default APPLHEAPSZ value is inadequate for applications that retrieve data from Oracle tables that contain one or more columns having a datatype of long.

[0080] To update the database configuration for the target database, we used the DB2 Command Line Processor to set the default application heap parameter (APPLHEAPSZ) for this database to 1024:

[0081] Step 12: Next insure that a temporary tablespace exists of adequate size for the transfer of the largest table stored in Oracle.

[0082] Configuring DB2 for OS/390

[0083] Step 13: Migrate the schema from Oracle to DB2 for OS/390, as shown in the Appendix.

[0084] Step 14: Install the source code for DMTFUNLD, the JCL for the precompile, assemble, link edit, and bind of DMTFUNLD, and the UNLOAD and RELOAD REXX EXECs, in your favorite assembler source code, JCL, and REXX libraries.

[0085] Step 15: Configure the DB2 for OS/390 Communications Database so that DB2 for OS/390 can function as a DRDA Application Requester to access the remote Oracle server via DB2 UDB, as shown in the Appendix.

[0086] This step included inserting a row into the SYSIBM.USERNAMES table with a column TYPE value of "O" for outbound translation, an AUTHID value of blank indicating that translation applies to all authorization ids, a LINKNAME value identical to the SYSIBM.LOCATIONS LINKNAME column value, and a userid and password values respectively for the NEWAUTHID and PASSWORD columns that specify a user ID and password of a user that has the authority to access the Oracle tables via the nicknames and views created above.

[0087] While the invention has been described and illustrated with respect to certain preferred embodiments and exemplifications, it is not intended to limit the scope of the invention thereby, but solely by the claims appended hereto.

Appendix

Characteristics of Environment in which AIX Prototype was Developed

During the prototyping effort, various commands, jobs, and SQL statements were executed in order to establish and configure the execution environment for the tool. Many of these commands, the JCL for the jobs, and the SQL statements are included below as examples. In order to understand these examples, you need to be aware of the following characteristics of the execution environment:

OS/390 Database Server: stplex4a.stl.ibm.com
DB2 for OS/390 Subsystem: SBL1 (DB2 for OS/390 V6)
Source library for DMTFUNLD: USRND01.SBL1.SDSNSAMP
JCL library: USRND01.SBL1.JCL
REXX EXEC library: USRND01.SBL1.REXX

AIX Database Server: yoda13_tr0.stl.ibm.com
DB2 UDB for AIX database: DMTDB271 (DB2 for UDB V7.1 fix pack 2)
Oracle Server: orac8i (Oracle Version 8.1.6 using Net8)

Configuring Relational Connect and DB2 UDB

The data migration tool uses the federated database support provided by DB2 UDB V7 and the federated database support for Oracle data sources provided by DB2 Relational Connect. The section "Setting Up a Federated System to Access Oracle Data Sources" in the chapter "Configuring a Federated Database System" of the DB2 UDB and DB2 Connect Installation and Configuration Supplement describes the steps that must be followed to configure DB2 UDB and Relational Connect for access to Oracle data. The steps include installing Relational Connect and adding Oracle data source to a federated system.

Adding Oracle Data Sources to a Federated System

When developing the AIX prototype, we performed the following steps for adding Oracle data sources to our federated system:

Step1: Install and configure the Oracle client software on the DB2 federated server using the documentation provided by Oracle.

We used Oracle Version 8.1.6 and Net8.

Step 2: Set data source environment variables by modifying the DB2DJ.ini file and issuing the db2set command. The db2set command updates the DB2 profile registry with the user settings settings.

We edited the db2dj.ini file to set the ORACLE_HOME, ORACLE_BASE, ORA_NLS, and TNS_ADMIN environment variables as follows:

```
ORACLE_HOME=/Oracle/ora8i
ORACLE_BASE=/Oracle/ora8i
ORA_NLS=/Oracle/ora8i/ocommon/nls/admin/data
TNS_ADMIN=/Oracle/ora8i/network/admin/tnsnames.ora
```

Step 3: Ensure that the SQL*Net or Net8 tnsnames.ora file is updated for each Oracle server to which communications are configured.

We updated the tnsnames.ora file for our server yoda13_tr0.stl.ibm.com.

Step 4: Recycle the DB2 instance:

```
$db2stop
$db2start
```

Step 5: Use the CREATE WRAPPER statement to define the wrapper library that will be used to access Oracle data sources. Wrappers are the mechanism that federated servers use to communicate with and retrieve data from data sources.

```
db2 => CREATE WRAPPER NET8;
```

Step 6: Set the DB2_DJ_COMM environment variable to include the wrapper library that corresponds to the wrapper module that you created in the previous step.

```
$ db2set DB2_DJ_COMM = libnet8.a;
```

Step 7: Use the CREATE SERVER statement to define each Oracle server to which communications are configured.

We created the Oracle server orac8i:

```
db2 => CREATE SERVER orac8i TYPE ORACLE VERSION 8.1.6 WRAPPER NET8
db2 (cont.) => OPTIONS  (NODE 'ora8i.stl.ibm.com');
```

Step 8: If a user ID or password at the federated server is different from a user ID or password at an Oracle data source, use the CREATE USER MAPPING statement to map the local user ID to the user ID and password defined at the Oracle data source.

We used the Oracle user ID SYS to access the Oracle data on server orac8i. SYS is one of the DBA users that is created when a database is installed and initialized. The DBA role has all system privileges and the ability to grant all privileges to other users. The Oracle user SYS owns all base tables and user-accessible views of the data dictionary.

```
db2 => CREATE USER MAPPING FOR USER SERVER orac8i
db2 (cont.) => OPTIONS
db2 (cont.) => (REMOTE_AUTHID 'sys', REMOTE_PASSWORD
'change_on_install');
```

Step 9: For Oracle tables in your Oracle data source that do not contain long columns, use the CREATE NICKNAME statement to assign a nickname to the table. Note that the nickname must be the same as the table name specified for the DB2 for OS/390 target table.

For example:

```
db2 => create nickname mrz.s_employee for orac8i.system.s_employee;
```

Note: Due to schema restrictions, for the tables migrated in this example, Oracle Long data types had to be cast to DB2 UDB for z/OS varchar(length) data types, where length was the max actual length of the data contained in the Oracle Long column.

Step 10: For Oracle tables in the Oracle data source that do contain long columns, use the CREATE NICKNAME statement to assign a nickname to each table located in the Oracle data source and the CREATE VIEW statement to cast the datatype of each long column to varchar. Note that the view name must be the same as the table name specified for the DB2 for OS/390 target table.

For example:

```
db2 => create nickname nick.s_note_opty for orac8i.system.s_note_opty;
db2 =>  create view mrz.s_note_opty (row_id,created,created_by,last_upd,
db2 (cont.) => last_upd_by,dcking_num,modification_num,conflict_id,
db2 (cont.) => priv_flg,src_row_id,note_type,note) as select
row_id,created,
db2 (cont.) =>
created_by,last_upd,last_upd_by,dcking_num,modification_num,
db2 (cont.) =>
conflict_id,priv_flg,src_row_id,note_type,varchar(note,26)
db2 (cont.) => from nick.s_note_opty;
```

Modify the Configuration for your DB2 UDB Database

Step 11: Increase the APPLHEAPSZ configuration parameter setting to 1024.

The APPLHEAPSZ configuration parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent. The default APPLHEAPSZ value is inadequate for applications that retrieve data from Oracle tables that contain one or more columns having a datatype of long.

To update the database configuration for your target database, use the DB2 Command Line Processor to set the default application heap parameter (APPLHEAPSZ) for this database to 1024:

```
db2 => update database configuration for dmtdb271 using applheapsz 1024;
```

Step 12: Ensure that a temporary tablespace exists of adequate size for the transfer of the largest table stored in Oracle.

Configuring DB2 for OS/390

Step 13: Migrate the schema from Oracle to DB2 for OS/390.

a. For install validation, document the number of tables, indexes, triggers, stored procedures, and row counts for each table on the source system. NOTE: UDFs will not be migrated.
b. Perform the physical database design for DB2 UDB for OS/390, based on the sizing information obtained. Determine the number of databases, the number and type of table spaces required, and the primary quantity and secondary quantity values for the table space and index spaces.
c. Extract a description of all of the source database objects and place them in a text file that can be used to recreate all the database objects on the OS/390 system.
d. Using the information from the physical database design step, create the required stogroups, databases, and tablespaces.
e. Modify the text file to place the source tables in their designated tablespaces and to specify the space quantities for the tablespaces and indexspaces.

f. Using the text file, re-create the source tables and indexes on the target OS/390 system.

Step 14: Install the source code for DMTFUNLD, the JCL for the precompile, assemble, link edit, and bind of DMTFUNLD, and the UNLOAD and RELOAD REXX EXECs, in your favorite assembler source code, JCL, and REXX libraries.

During our prototyping effort, we used PDS's named 'USRND01.SBL1.SDSNSAMP', 'USRND01.SBL1.JCL', and 'USRND01.SBL1.REXX' respectively.

Step 15: Configure the DB2 for OS/390 Communications Database so that DB2 for OS/390 can function as a DRDA Application Requester to access the remote Oracle server via DB2 UDB:

a. insert a row into the SYSIBM.LOCATIONS table with a LOCATION column value that identifies the source DB2 UDB database, a LINKNAME column value that identifies a row in the SYSIBM.IPNAMES table, and a PORT column value of the port number associated with the source DB2 UDB database server.
b. Insert a row into the SYSIBM.IPNAMES table with a LINKNAME column value identical to the SYSIBM.LOCATIONS LINKNAME column value, an IPADDR column value of the IP address or domain name of the source DB2 UDB database server, a SECURITY_OUT column value of "P" and a USERNAMES column value of "O" indicating that a userID and password will be sent during connect process.
c. Insert a row into the SYSIBM.USERNAMES table with a column TYPE value of "O" for outbound translation, an AUTHID value of blank indicating that translation applies to all authorization ids, a LINKNAME value identical to the SYSIBM.LOCATIONS LINKNAME column value, and an userid and password values respectively for the NEWAUTHID and PASSWORD columns that specify a user ID and password of a user that has the authority to access the Oracle tables via the nicknames and views created in steps 9 and 10 above under Adding Oracle Data Sources to a Federated System.

Update of Communications Database

```
INSERT INTO SYSIBM.LOCATIONS
(LOCATION,LINKNAME,IBMREQD,PORT,TPN)
VALUES ('DMTDB271','DMTDB271','','10010','');

INSERT INTO SYSIBM.IPNAMES
(LINKNAME,SECURITY_OUT,USERNAMES,IPADDR)
VALUES ('DMTDB271','P','O','YODA13_TR0.STL.IBM.COM');

INSERT INTO SYSIBM.USERNAMES (TYPE,AUTHID,LINKNAME,NEWAUTHID,PASSWORD)
VALUES ('O',' ','DMTDB271','DB2INST7','DB2RCTST')
```

Step 4: Precompile, assemble, link edit, and bind the plan for DMTFUNLD. Grant EXECUTE authority on the plan for DMTFUNLD to the user or userids that will be performing the data migration.

The plan for DMTFUNLD consists of two packages, one for DB2 UDB and another for DB2 for OS/390. Specify the location name for DB2 for UDB as the value for the CURRENTSERVER option of the BIND PLAN command.

The JCL that we used during our prototyping effort for these tasks follows. In our effort, we granted EXECUTE authority to PUBLIC.

```
//BINDDMT  JOB ,
//                    TIME=1440,CLASS=A,MSGCLASS=A,
//                    REGION=8M,USER=USRND01
//JOBPROC   JCLLIB ORDER=(DSN.SBL1.SDSNPROC,SYS1.DBDC.PROCLIB)
//JOBLIB       DD  DSN=DSN.SBL1.SDSNLOAD,DISP=SHR
//*******************************************************************
*
//*
//* PRECOMPILE, ASSEMBLE, AND LINK EDIT DMTFULND
//*
//*******************************************************************
**
//PREPUNL EXEC DSNHASM,MEM=DMTFUNLD,
//         PARM.PC='HOST(ASM),STDSQL(NO)',
//         PARM.ASM='RENT,OBJECT,NODECK',
//         PARM.LKED='RENT,XREF'
//PC.DBRMLIB   DD DSN=USRND01.SBL1.DBRMLIB.DATA(DMTFUNLD),
//             DISP=SHR
//PC.SYSLIB    DD DSN=USRND01.SBL1.SDSNSAMP,
//             DISP=SHR
//PC.SYSIN     DD DSN=USRND01.SBL1.SDSNSAMP(DMTUNLDD),
//             DISP=SHR
//ASM.SYSLIB   DD
//             DD DSN=DSN.SBL1.SDSNMACS,
//             DISP=SHR
//             DD DSN=USRND01.SBL1.SDSNSAMP,
//             DISP=SHR
//             DD DSN=DSN.SBL1.SDSNSAMP,
//             DISP=SHR
//LKED.SYSLMOD DD DSN=USRND01.SBL1.RUNLIB.LOAD(DMTFUNLD),
//             DISP=SHR
//LKED.SYSIN   DD *
  INCLUDE SYSLIB(DSNELI)
  NAME DMTFUNLD(R)
//*
//* BIND DMTFUNLD AND GRANT EXECUTE AUTHORITY TO PUBLIC
//*
//BINDUNL EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//DBRMLIB  DD  DSN=USRND01.SBL1.DBRMLIB.DATA,
//             DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(SBL1)
  BIND PACKAGE(DMTDB271.DMTPKG)                                  +
       MEMBER(DMTFUNLD) ACTION(REP) VALIDATE(BIND)               +
       ISOLATION(CS) LIB('USRND01.SBL1.DBRMLIB.DATA');
  BIND PACKAGE(DMTPKG)                                           +
       MEMBER(DMTFUNLD) ACTION(REP) VALIDATE(BIND)               +
       ISOLATION(CS) LIB('USRND01.SBL1.DBRMLIB.DATA');
  BIND PLAN(DMTFUNLD)                                            +
```

```
      PKLIST(DMTPKG.DMTFUNLD,DMTDB271.DMTPKG.DMTFUNLD)     +
      ACTION(REP) ISOLATION(CS) CURRENTSERVER(DMTDB271)    +
      VALIDATE(BIND) DEFER(PREP);
  RUN PROGRAM(DSNTIAD)  PLAN(DSNTIA61) -
      LIB('USRND01.SBL1.RUNLIB.LOAD')
  END
//SYSIN    DD  *
   GRANT EXECUTE ON PLAN DMTFUNLD TO PUBLIC;
```

Step 5: Build the JCL for the unload and load jobs.

Use the REXX program DMT to generate the JCL for the unload and load jobs. This program requires a sequential input file in which each record contains the following five fields:

```
dbname -  target database name
tsname -  target tablespace name
ownername -  owner name for table
tablename -  table name
maxrowlength -  maximum length of rows within the table
```

DMT generates one unload job that unloads each of the tables listed in the input file, and one load job for each table unloaded by the unload job.

For an input file with the following records:

```
DMTTSTDB DMTTST NICK     S_EMPLOYEE   3583
DMTTSTDB DMTTST NICK     S_ZIPCODE    1000
```

DMT generates one unload job and two load jobs:

Unload Job for NICK.S_EMPLOYEE and NICK.S_ZIPCODE

```
//UNLD001      JOB ,
//   MSGLEVEL=(1,1),MSGCLASS=A,
//   TIME=1440,REGION=8M,
//   CLASS=A,
//   NOTIFY=USRND01,USER=USRND01
/*JOBPARM S=SY4A,LINES=9999
/*ROUTE PRINT STLVM14/ZIMOWSKI
//JOBLIB    DD DISP=SHR,
//          DSN=DSN.SBL1.SDSNLOAD
//*-------------------------------------------------
//*
//* Submit LOAD job
//*
//*-------------------------------------------------
//S001   EXEC PGM=IEBGENER
//SYSPRINT  DD DUMMY
//SYSUT1    DD DSN=USRND01.SBL1.LOAD(LOAD001),
//          DISP=SHR
//SYSUT2    DD SYSOUT=(A,INTRDR),DCB=BLKSIZE=80
//SYSIN     DD DUMMY
//*-------------------------------------------------
//*
```

```
//* Unload the data
//*
//*-------------------------------------------------
//U001   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(SBL1)
 RUN  PROGRAM(DMTFUNLD) PLAN(DMTFUNLD) -
      LIB('USRND01.SBL1.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPUNCH DD DSN=USRND01.SBL1.UNLD001.SYSPUNCH,
//  DCB=(RECFM=FB,LRECL=80,BUFNO=10),
//  SUBSYS=(BP01)
//SYSREC00 DD DSN=USRND01.SBL1.UNLD001.SYSREC00,
//  DCB=(RECFM=FB,LRECL=3583,BUFNO=255),
//  SUBSYS=(BP01,'OPENNOW')
//SYSIN    DD *
  NICK.S_EMPLOYEE
//*-------------------------------------------------
//*
//* Submit LOAD job
//*
//*-------------------------------------------------
//S002   EXEC PGM=IEBGENER
//SYSPRINT  DD DUMMY
//SYSUT1    DD DSN=USRND01.SBL1.LOAD(LOAD002),
//          DISP=SHR
//SYSUT2    DD SYSOUT=(A,INTRDR),DCB=BLKSIZE=80
//SYSIN     DD DUMMY
//*-------------------------------------------------
//*
//* Unload the data
//*
//*-------------------------------------------------
//U002   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(SBL1)
 RUN  PROGRAM(DMTFUNLD) PLAN(DMTFUNLD) -
      LIB('USRND01.SBL1.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPUNCH DD DSN=USRND01.SBL1.UNLD002.SYSPUNCH,
//  DCB=(RECFM=FB,LRECL=80,BUFNO=10),
//  SUBSYS=(BP01)
//SYSREC00 DD DSN=USRND01.SBL1.UNLD002.SYSREC00,
//  DCB=(RECFM=FB,LRECL=1000,BUFNO=255),
//  SUBSYS=(BP01,'OPENNOW')
//SYSIN    DD *
  NICK.S_ZIPCODE
//*

Load Job for NICK.S_EMPLOYEE

//LOAD001      JOB ,
//   MSGLEVEL=(1,1),MSGCLASS=A,
```

```
//    TIME=1440,REGION=8M,
//    CLASS=A,
//    NOTIFY=USRND01,USER=USRND01
//JOBPROC JCLLIB ORDER=(DSN.SBL1.SDSNPROC)
/*JOBPARM S=SY4A,LINES=9999
/*ROUTE PRINT STLVM14/ZIMOWSKI
//JOBLIB  DD  DSN=DSN.SBL1.SDSNLOAD,DISP=SHR
//*-------------------------------------------------
//*
//* Load the data using the LOAD utility
//*
//*-------------------------------------------------
//LOAD    EXEC DSNUPROC,PARM='SBL1,DSNTEX'
//DSNTRACE DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SYSIN    DD  DSN=USRND01.SBL1.UNLD001.SYSPUNCH,
//         DCB=(RECFM=FB,LRECL=80,BUFNO=10),
//         SUBSYS=(BP01,'OPENNOW')
//SYSREC00 DD  DSN=USRND01.SBL1.UNLD001.SYSREC00,
//         DCB=(RECFM=FB,LRECL=3583,BUFNO=255),
//         SUBSYS=(BP01,'OPENNOW')
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK01 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK02 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK03 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK04 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK05 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK06 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK07 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK08 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK09 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK10 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK11 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK12 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK13 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK14 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK15 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK16 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK17 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK18 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK19 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK20 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK21 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK22 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK23 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK24 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK25 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK26 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK27 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK28 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK29 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK30 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK31 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK32 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK33 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
```

```
//SW00WK34 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK35 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK36 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK37 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK38 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK39 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK40 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK41 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK42 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK43 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK44 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK45 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK46 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK47 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK48 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK49 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK50 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//*-------------------------------------------------
//*
//* Run the REPAIR utility
//*
//*-------------------------------------------------
//R001   EXEC DSNUPROC,PARM='SBL1,DSNTEX'
//DSNTRACE DD  SYSOUT=*
//SYSIN    DD  *
  REPAIR OBJECT
  SET TABLESPACE DMTTSTDB.DMTTST NOCOPYPEND
//*


Load Job for NICK.S_ZIPCODE

//LOAD002      JOB ,
//   MSGLEVEL=(1,1),MSGCLASS=A,
//   TIME=1440,REGION=8M,
//   CLASS=A,
//   NOTIFY=USRND01,USER=USRND01
//JOBPROC JCLLIB ORDER=(DSN.SBL1.SDSNPROC)
/*JOBPARM S=SY4A,LINES=9999
/*ROUTE PRINT STLVM14/ZIMOWSKI
//JOBLIB  DD  DSN=DSN.SBL1.SDSNLOAD,DISP=SHR
//*-------------------------------------------------
//*
//* Load the data using the LOAD utility
//*
//*-------------------------------------------------
//LOAD    EXEC DSNUPROC,PARM='SBL1,DSNTEX'
//DSNTRACE DD  SYSOUT=*
//UTPRINT  DD  SYSOUT=*
//SORTLIB  DD  DSN=SYS1.SORTLIB,DISP=SHR
//SYSIN    DD  DSN=USRND01.SBL1.UNLD002.SYSPUNCH,
//         DCB=(RECFM=FB,LRECL=80,BUFNO=10),
//         SUBSYS=(BP01,'OPENNOW')
//SYSREC00 DD  DSN=USRND01.SBL1.UNLD002.SYSREC00,
//         DCB=(RECFM=FB,LRECL=1000,BUFNO=255),
//         SUBSYS=(BP01,'OPENNOW')
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
```

```
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK01 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK02 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK03 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK04 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK05 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK06 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK07 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK08 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK09 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK10 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK11 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK12 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK13 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK14 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK15 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK16 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK17 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK18 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK19 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK20 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK21 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK22 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK23 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK24 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK25 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK26 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK27 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK28 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK29 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK30 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK31 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK32 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK33 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK34 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK35 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK36 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK37 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK38 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK39 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK40 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK41 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK42 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK43 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK44 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK45 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK46 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK47 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK48 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK49 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//SW00WK50 DD UNIT=SYSDA,SPACE=(CYL,(1100,1100))
//*-------------------------------------------------
//*
//* Run the REPAIR utility
//*
//*-------------------------------------------------
//R002    EXEC DSNUPROC,PARM='SBL1,DSNTEX'
```

```
//DSNTRACE DD  SYSOUT=*
//SYSIN    DD  *
  REPAIR OBJECT
  SET TABLESPACE DMTTSTDB.DMTTST NOCOPYPEND
//*
```

Step 6: Execute the RUNSTATS utility to gather statistical information on the tables and indexes within the migrated database.

We claim:

1. A method of migrating data from a table in a source system to a table in a target system comprising the steps of:

a. initiating two parallel batch processes,

b. one of said batch processes unloading the data from the source system and uploading the data into the target system, and

c. the other of said batch processes loading the source data into a database table at the target system.

2. The method of claim 1 comprising using an SQL "SELECT * FROM tablename" command to unload the data from the source system, where "tablename" is the name of the source table.

3. The method of claim 1 comprising using a LOAD Utility to load the data into a database at the target system.

4. The method of claim 1 comprising

a. initiating the two parallel batch processes on the target system,

b. first initiating the said batch process to load the source data into a database at the target system; and

c. thereafter initiating the other of one of said batch processes to unload the data from the source system and uploading the data into the target system.

5. The method of claim 1 wherein the tables are relational database tables.

6. The method of claim 1 comprising generating on the target system control languages to submit and execute batch jobs for data transfer.

7. A computer system comprising at least one database management system configured and controlled as a target system and adapted for communication with a second database management system functioning as a source system, said computer system comprising at least one database management system configured and controlled as a target system, said computer system being further configured and controlled to execute two batch processes from the target system,

a. one of said batch processes unloading data from the source system and uploading the data to the target system, and

b. the other of said batch processes loading the data into a database at the target system.

8. The system of claim 7 comprising using an SQL "SELECT * FROM tablename" command to unload the data from the source system, where "tablename" is the name of the source table.

9. The system of claim 7 comprising using a LOAD Utility to load the source data into a database at the target system.

10. The system of claim 7 further adapted to:

a. initiate the two parallel batch processes on the target system,

b. first initiate the said batch process to load the data into a database at the target system; and

c. thereafter initiate the other of one of said batch processes to unload the data from the source system and upload the data into the target system.

11. The system of claim 7 wherein the tables are relational database tables.

12. The system of claim 7 further adapted and controlled to generate on the target system control languages to submit and execute batch jobs for data transfer.

13. A program product comprising computer readable instructions on a media, said instructions being capable of controlling and configuring a computer to carry out the method of migrating data from a data base in a source system to a data base in a target system and comprising the steps of:

a. initiating two parallel batch processes,

b. one of said batch processes unloading data from the source system and uploading the data to the target system, and

c. the other of said batch processes loading the data into a data table at the target system.

14. The program product of claim 13 wherein the program product further comprises computer readable instructions for using an SQL "SELECT * FROM tablename" command to unload the data from the source system, where "tablename" is the name of the source table.

15. The program product of claim 13 wherein the program product further comprises computer readable instructions for using a LOAD Utility to load the data into the table at the target system.

16. The program product of claim 13 wherein the media is magnetic or optical media.

17. The program product of claim 13 wherein the computer readable instructions are readable for at least one of transmission, installation, instantiation, or execution.

18. The program product of claim 13 wherein the computer readable instructions are in the form of executable code.

19. The program product of claim 13 wherein the computer readable instructions are in the form of code that requires loading to memory for execution.

20. The program product of claim 13 wherein the computer readable instructions are in the form of distribution code that requires one of both of decompression or decryption to be executable.

21. The program product of claim 13 further containing instructions to:

a. initiate the two parallel batch processes on the target system,

b. first initiate the said batch process to load the data into a database at the target system; and

c. thereafter initiate the other of one of said batch processes to unload the data from the source system and upload the data into the target system.

22. The program product of claim 13 wherein the tables are relational database tables.

23. The program product of claim 13 further containing instructions to generate on the target system control languages to submit and execute batch jobs for data transfer.

24. A program product comprising computer readable instructions on a media, said instructions being capable of controlling and configuring a computer to carry out the method of migrating data from a data base in a source system to a data base in a target system and comprising the steps of:

a. initiating two processes on the target system,

b. one of said processes unloading data from the source system and uploading the data to the target system, and

c. the other of said processes loading the data into a table at the target system.

25. The method of claim 24 wherein the processes are parallel processes.

26. The method of claim 24 wherein the processes are batch processes.

27. The program product of claim 24 wherein the program product further comprises computer readable instructions for using an SQL "SELECT * FROM tablename" command to unload the data from the source system, where "tablename" is the name of the source table.

28. The program product of claim 24 wherein the program product further comprises computer readable instructions for using a LOAD Utility to load the data into a table at the target system.

29. The program product of claim 24 wherein the media is magnetic or optical media.

30. The program product of claim 24 wherein the computer readable instructions are readable for at least one of transmission, installation, instantiation, or execution.

31. The program product of claim 24 wherein the computer readable instructions are in the form of executable code.

32. The program product of claim 24 wherein the computer readable instructions are in the form of code that requires loading to memory for execution.

33. The program product of claim 24 wherein the computer readable instructions are in the form of distribution code that requires one of both of decompression or decryption to be executable.

34. The program product of claim 24 further containing instructions to:

a. initiate the two parallel batch processes on the target system,

b. first initiate the said batch process to load the data into a database at the target system; and

c. thereafter initiate the other of one of said batch processes to unload the data from the source system and upload the data into the target system.

35. The program product of claim 24 wherein the tables are relational database tables.

36. The program product of claim 24 further containing instructions to generate on the target system control languages to submit and execute batch jobs for data transfer.

* * * * *