Fall 12-1-2010

Database Merge

Bob Van Roekel
Dakota State University

DATABASE MERGE

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

December, 2010

By

Bob Van Roekel

Project Committee:

Dr. Stephen Krebsbach

Dr. Mark Moran

Dr. Ronghau Shan

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Bob Van Roekel

Master's Project Title: Database Merge

Faculty supervisor: Dr Stephen Krebsbach Date: 12/8/10

Committee member: Dr. Mark Moran Date: 12/8/2010

Committee member: Dr. Ronghua Shan Date: 12/8/10

ABSTRACT

GE Healthcare is a worldwide leader in healthcare solutions. They are involved in almost every aspect of healthcare from medical devices to software solutions to assist physicians. One of their software solutions captures data from the medical devices and then uses that data to formulate procedural reports for the physician. This enables the physician to spend less time dictating and transcribing their findings and more time to see additional patients. One problem GE Healthcare is faced with is when hospitals become acquired by larger hospitals and both hospitals are running GE Healthcare solutions. The hospitals do not want to have to maintain two separate systems, but rather want them combined into one central system that can be accessed by all facilities.

The data for these systems are stored in an Oracle database and there are no automated methods available to move the data from one database to another database. The databases use system generated identifiers to maintain the relationships between the data entities. This fact makes it very difficult to merge the data between two databases. The data cannot just be brought over from the source database into the destination database because the identifiers may already exist in the destination database. This requires that new identifiers be generated as the data is being brought over from the source database. This makes the processing of child tables very complex. The parent identifier of the child record needs to be synchronized with the new identifier of the parent entity. The complexity of this situation is compounded by the fact that a single record can point to multiple parent records and all these relationships will need to be maintained as the data is being brought over.

This project will involve a user-interface (UI) that will allow users to specify the databases to be merged and give the users the option to specify the data elements to be included in the merge. There will also be an Oracle package and supporting schema that will actually perform the task of moving the data from the source database to the destination database. The system will have to be able to be ran multiple times and only bring over data that does not already exist in the destination database. The destination database will also need

to be fully functional to other users of the system with no significant performance degradation.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Bob Van Roekel

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Background of the Problem

GE Healthcare has many different software solutions available for the medical community. One of their solutions is a product called Data Management System (DMS). DMS is essentially a data repository that contains data dealing with patients, admissions, and the medical procedures associated to those patients that occurred in the Catheterization lab (Cath Lab). The DMS software has various interfaces that feed it data. Some of this data comes from the Hospital Information Systems (HIS) and includes patient, admission, orders, scheduling, and inventory data. Other data comes from the medical devices used during the procedures in the Cath Lab. The DMS software captures all this data, and in-short, produces the physician report that includes all the detail information that occurred during the procedure.

Once the physician has completed the procedure in the Cath Lab, they use the DMS software to verify the generated report is accurate and then electronically signs off on the report. The report then gets routed to the various stakeholders – HIS, referring physician(s), etc. The report can be routed to a printer, fax, email address or mobile device. The ability to capture the data from the medical device allows the physician to be more productive as they no longer have to manually dictate the events of the procedure. This gives the physician time to see more patients and do more procedures during the course of the day and less time doing paperwork and dictation.

The DMS software is a great help to the physicians and to the hospital administrative staff. The problem comes in when one hospital is bought out by another hospital and both facilities have their own DMS systems. Obviously it would make sense for the newly combined IT departments to have to manage just one DMS system. Also, the users of the DMS system do not want to have to login to two different systems and maintain two separate systems. Each DMS system comes on its own server so there is also duplicated hardware involved and the cost of maintaining that hardware. GE Healthcare currently does not have a solution for their customers in this situation so they are forced to maintain the duplicated servers and software.

As more and more customers are being acquired by larger healthcare organizations this is becoming a bigger problem. GE Healthcare customers are requesting a solution to this problem and GE Healthcare is committed to resolving this problem for their current and future customers.

Statement of the problem

GE Healthcare customers that currently have multiple DMS systems need a way to migrate/merge these duplicate systems into one DMS system. This will simplify their operations and reduce cost in maintaining multiple systems. The problem of merging multiple databases together is that it is not a simple or straightforward process. The DMS system uses an Oracle database to store all of the data associated to the product. The Oracle database is a file based system and as such you cannot just copy the files from the source database to the destination database. Copying the files would result in a loss of data on the destination database as the existing data would be overwritten by the data from the source database file.

A systematic approach would be needed to ensure the integrity of the database is maintained. Oracle does not offer any solution to merging databases and there are no other Commercial, off-the-shelf (COTS) software available to assist in this matter. GE Healthcare has decided they need to develop a system that will allow their customers who have multiple DMS systems to merge those systems in one DMS system. This product will be a separate product offering to the customers and will allow GE Healthcare to gain a higher level of customer satisfaction with the added benefit of increased revenue.

This new product will be called DB Merge and will be a billable product to the customers that have multiple DMS systems and need them merged into one system. More and more hospitals are joining associations to reduce their costs and overhead, but by doing this they are sometimes duplicating their systems. Hospitals that find themselves with multiple DMS systems need a way to combine these systems into one system and GE Healthcare will give them that opportunity with the new DB Merge product offering.

Objectives of the project

The objectives of the DB Merge project will be to create a utility that can be used to merge the contents of one DMS database into another DMS database, ensuring the integrity of the data is maintained. The project will consist of a set of database tables used to store metadata as well as stored procedures that will actually merge the data between the source and destination databases. There will also be a User Interface (UI) that will allow the user to define the merge. The UI will allow users to specify the databases to be merged and give the user the option to specify the data elements to be included in the merge. The system will have to be able to be ran multiple times and only bring over data that does not already exist in the destination database. The destination database will also need to be fully functional to other users of the system with no significant performance degradation. Upon completion of the merge all the data relating to a location in the source server will reside in the destination server and will be available to use as it existed in the source server.

The UI portion of the merge should include a screen that can be used to create/define the merge. Here the user will specify the two servers that will be used in the merge as well as the other basic information about the merge. They will also be able to specify the procedural data they want brought over in the merge.

There are some basic data identifiers that could exist in both the source and destination databases. These elements such as Medical Record Number, Account Number, and Employee Id may be the same in both database but could reference different records. For example, in the source database you could have medical record number "ABC" that belongs to John Doe. The destination database could have the same medical record number but have it belong to Bill Smith. The medical record number is the unique key in the patient table and data will need to be queried by the unique key to determine if the element already exists. If it is assumed that ABC is the same record in the source and destination database, and procedure data is brought over for that patient, the system has just associated procedure data to the wrong patient. So to eliminate this problem there will need to be a screen in the UI where the user can identify these potential problems and either fix the issue or signify that it is indeed the same patient.

The final portion of the UI will be where the user initiates and monitors the merge. Because a merge could take several hours\days based on the amount of data, the user should be able to specify the various groups of the merge they want to run. These groups should be made up of Users, Employees, System Management, Patient, Admission and Procedure. This will allow the user to break the merge down into more manageable sections. Also, if there are errors in one group, it will more than likely have ramifications on the next group, so it is better to run one group and fix any problems before moving on to the next group. The UI will also have to enforce that the user run the groups in the specified order to ensure data integrity.

The Database Merge project will use database links to merge the data between the source and destination databases. This requires that the two databases be able to communicate to each other through the use of Oracle's network capabilities via the TNSNAMES.ora. The actual process of moving data between the servers will be initiated by a database job. This job will be created/started by the UI and will allow the UI to close without hindering the processing of the data. The database job will run in the background and so there will be no direct ties to the UI. This will allow the user to close the UI on long running portions of the merge and then to restart the UI to check the progress of the merge. There also needs to be an option to kill the job if the process appears to be hung or there are known issues. Once the job has been killed the UI should be notified and should also be able to restart the job and have it continue where it left off.

The database merge will need to be processed in a hierarchical manner to ensure parent data exists before child data is processed. There will need to be a series of tables that can group the data tables together by their functionality, i.e. system management, employee, patient, admission, and procedural. Within these groupings there will need to be a column that determines the order in which the individual tables within the group should be processed. This should ensure the parent tables are processed before the child tables and enforce the integrity of the data.

The database merge will be governed by a rule table. This table will house the rules needed to process the data between the two servers. There will be rules defined for various levels of data

processing, including table rules, column rules, before and after processing rules, and rules that determine how the data is pulled from the source server (SELECT/WHERE). Once processing begins the rule data should be stored in a PL/SQL table (array). The rules then will be in memory and will speed up the retrieval of the rules as there will be no need to query the database when processing the multiple rows of data for the same table.

There will also need to be a table that stores the old primary key and new primary key for each table\row that could have child rows associated to it. To facilitate the look up of these ID's, the ID's should also be stored in a PL/SQL table (array), this will speed up retrieval of the new ID's as the data is all stored in memory and a query will not be needed for the lookup. So when data is being processed and the column is a primary key, the process will need to get the new value and then create an entry in the lookup table that contains the old key and the new key. Then when processing columns that are foreign keys, the process will need to get the new value from the lookup table based on the old key value. This process will ensure the integrity of the data as it is being populated in the destination database.

As the merge is being processed, it will need to log the various activities and percentage complete of each process so the UI can be apprised of the current status. The UI will use this data to update the progress bar and inform the user of the current task being performed. Once the merge is complete, the database should send an email to the user to inform them that the process is finished and also inform them if any errors occurred during the run. This is essential for portions (procedure data) that could take many hours\days to complete. This will allow the user to go about their other activities and not have to manually monitor the merge. When the user gets the email notification they can fix any errors and then move on to the next phase of the merge.

SYSTEM DESIGN (RESEARCH METHODOLOGY)

Overview

The following section lists the process flow for a customer that requires a database merge for the DMS Application.

Once a customer has either been identified as potentially needing a DB merge or has requested a DB merge, the following process will need to occur:

1. A workout with the customer to determine if the merge being requested is supported is the first step. If the merge type is supported based on the configuration type of the source and destination servers, the customer will need to be presented with a Statement of Work.
2. A Statement of Work will then be drafted and presented to the customer in order to set the customers' expectations. The statement of work will identify areas in DMS that will and will not be merged during the merge. It will explain a merge can only be performed once and that all decisions made prior to the merge are final once the merge has been performed, with the exception of updating new patient/admission collisions encountered during the merge (see below). Once the statement of work has been accepted, the merge utility can be installed to start the merge process.
3. Ensuring systems are at the minimum version and are both at the same version is the first step needed to ensure the merge utility will function as intended and will even install. If the systems are not at the minimum version they must be upgraded. If the DMS versions are different, they must be upgraded to meet this requirement. The destination server must be verified that it is at least at Oracle version 10g, if not, it will need to up upgraded to the latest DMS supported version of Oracle.
4. Installing the merge utility is the next step. The merge utility needs to be installed on the both servers. Once the utility is installed, the merge specialist may proceed with selecting any options pertaining to the merge.
5. The Merge Specialist must select merge options in order to determine which data sets are processed during the merge. Selections include areas such as STS procedural data, ACC procedural data, Inventory, Cath procedural data, etc. Based on the merge type and the customer selections, the customer may have to perform additional mappings in order for the merge to proceed.
6. The Merge Specialist will kick off the job to convert the long data type columns in the Source-server to CLOB data type. This process must be completed prior to the Merge process. Customers that were initial customers at DMS 4.0 and after will not need to have this process ran as these data types were no longer used after 4.0.
7. If mappings are needed, the user will be presented with like data sets from both the Source-server and the Destination-server and they must match like items and potentially enter data needed to avoid collisions when the data is merged. Examples are covered in the Mapping Component section below.

8. Customer needs to run the Patient Identification (PID) Mismatch report on the source server to identify and fix PID mismatches that exist in the source server. They also need to run the Procedure Merge report to identify and fix procedures that exist on the source server and need to be merged within DMS. Procedures that have PID mismatches or procedures that need to be merged, will not be brought over until their issues have been resolved.
9. Move Source-server processing to Destination-server if the customer was still using the Source-server as the production server. If the customer has already started using the Destination-server as the production server this step can be skipped, otherwise the clients need to be re-directed to the Destination-server and the interfaces need to be reconfigured on the Destination-server.
10. An Electronic signoff is required for the merge. This signoff signifies the customer is satisfied with their selections and mappings.
11. The Merge Process will be initiated upon completion of the electronic sign-off. The actual data merged from one DMS database to another DMS database is contained within this process. An email notification will be sent to the Merge Specialist and the Customer Champion notifying them that the merge completed. At the end of the process, if any new collisions were detected that require user intervention/mappings, the user will have an opportunity to revisit mappings and start the merge again. The second round of processing only handles patient/admission, any system management data will only be processed one time for any given server/location combination.
12. A Merge Report will be generated after the merge is complete. The report will signify the merge is complete and will detail the work performed during the merge.
13. If some patients/cases were not merged, additional mappings may be needed, and then the merge process can be run again (starting from the signoff) for patients and cases only – System Management data will only process one time only.
14. If the merge process created a new location on the Destination server, that location will need to be associated to an Interface server
15. A Notice of Completion will be presented to the customer to finalize the process. Acceptance of this document by the customer will signify the merge completion.
16. An internal review of the database merge will be performed by the owner of the DB merge process after completing each individual customer's DB merge to identify best practices and lessons learned.
17. Repeat the above steps if an additional location on the Source-server needs to be processed or any additional servers that need to be merged to the Destination-server. The mapping and selection process for multiple locations can be done simultaneously (steps 1-6), but the merge process (steps 7-13) will only process one location at a time.

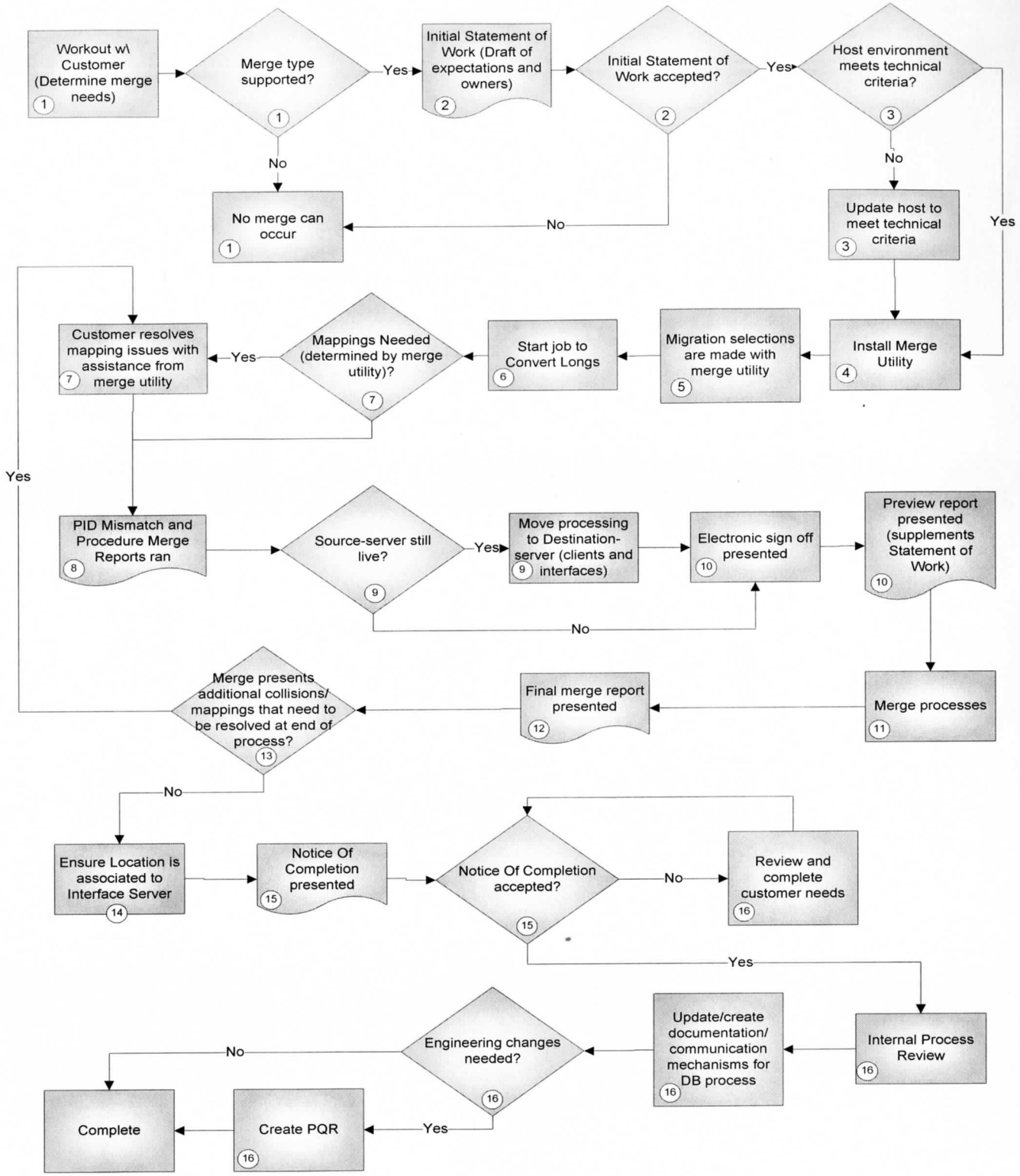


Figure 1 - Process Flow

Use Case Diagram

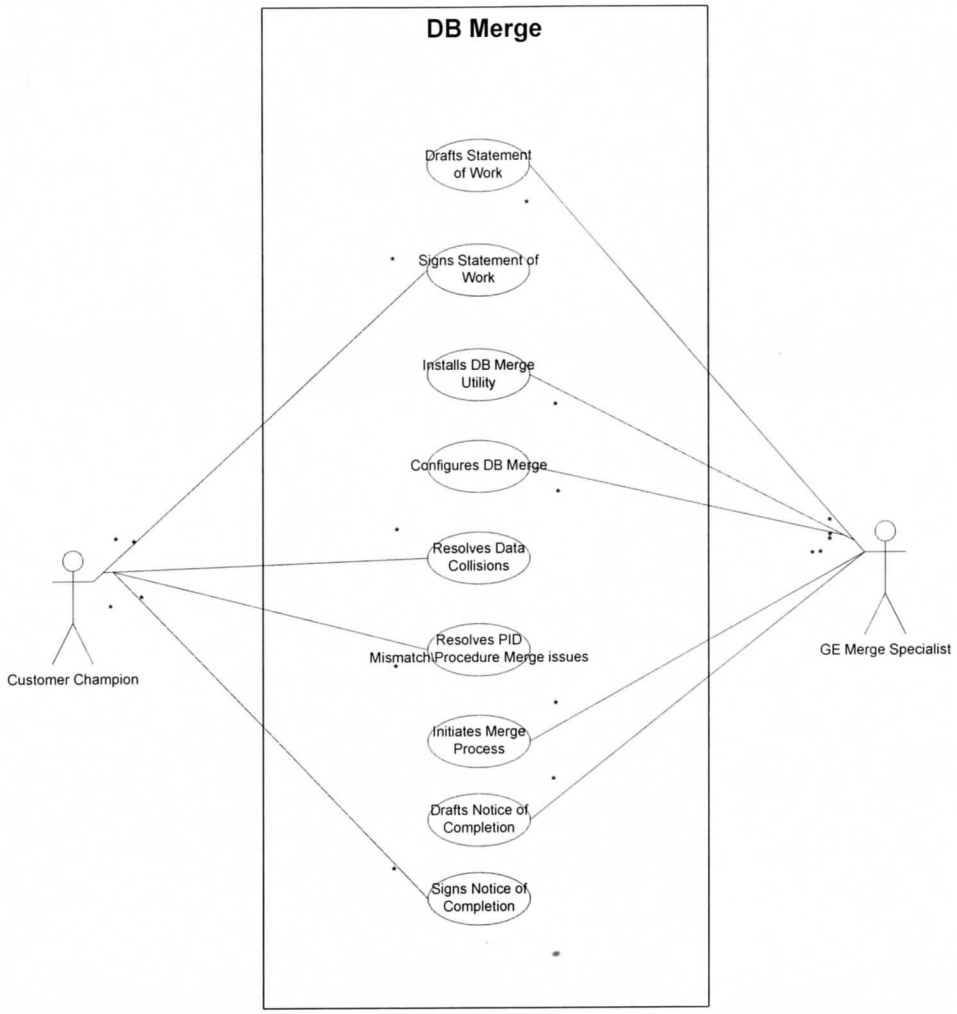


Figure 2 – Use Case Diagram

Database Design

Overview

There are 2 fundamental elements to the DB merge database design:

- Data model
- Stored Procedures

Data Model

The data model provides meta-data about the entire DMS data model in order to dynamically generate SQL to merge data between 2 DMS systems (Source-server and Destination-server).

The following diagram provides a list and description of the tables in DB merge data model (see appendix C for the ER Diagram):

SYSMAN_DD_TABLE	Stores information on every table and view in the DMS schema for documentation purposes, and for additional functions in which knowledge of valid tables are required.
SYSMAN_DD_MRG_RULE	Table used internally by DB MERGE to define rules for columns so the merge process knows how to insert/update the table/column. This table will be copied over to sysman_dd_mrg_rule_pr each time the merge process is ran.
SYSMAN_DD_MRG_RULE_PR	Table used internally by DB MERGE to define rules for columns so the merge process knows how to insert/update the table/column.
SYSMAN_DD_MRG_LKUP_KEY	Table used internally by DB MERGE to store new primary key id and their corresponding old primary key id. This is primarily used when inserting foreign keys so we can get the new id based on the old id.
SYSMAN_DD_MRG_GROUP	Lookup table that stores all group names used for DB MERGE.
SYSMAN_DD_MRG_GROUP_REL	Lookup table that stores group relationships for DB merges. An example being the REGISTRY group is needed to import the STS group).
SYSMAN_DD_MRG_TAB_GROUP	Stores which tables to process when processing a particular group.
SYSMAN_DD_MRG_AUDIT	Store audit trail of the merge process.
SYSMAN_DD_MRG_LOB_TMP	Temporary table used by merge process to move lob columns across the DB Link.

SYSMAN_DD_MRG_SELECTION	Used to specify what areas within DMS are to be merged (CATH, ECHO, ACC, STS, etc)
SYSMAN_DD_MRG_SELCTION_D	Used to store default data that is used to populate sysman_dd_mrg_selection
SYSMAN_DD_MRG_SOW_NOC	Used to store basic information about the merge – server/location names, status, statement of work and notice of completion dates, etc.
SYSMAN_DD_MRG_USER_MAPPED	Used to store historical mapping data – holds the person who created the mapping and the specific mapping that they made
SYSMAN_DD_MRG_COLLISION	Table used to store collision data that is presented to the user in the UI prior to the merge process.
SYSMAN_DD_MRG_MAPPED_ITM	Used by the merge process to store items(Patients, Admissions, Employees) that need to be processed.
SYSMAN_DD_MRG_PROCESS	Used by the user to specify what groups of data they want to process in the current execution of the merge (USERS, EMPLOYEES, SYS ADMIN, PATIENTS, ADMISSIONS, PROCEDURES, REGISTRIES)
SYSMAN_DD_MRG_PROCESS_D	Used to populate sysman_dd_mrg_process table with default values

Figure 3 – DB Merge Table Descriptions

Stored Procedures

The DB merge design consists of one package, mms.DBMerge, which contains all the stored procedures and functions utilized for the merge process. The package is the heart of the merge process. The package performs all the data merging and audit logging.

The following diagram lists the main callable stored procedures of the dbmerge package:

ResetDBMergeJob	Used to create the dbmerge job that will run the merge for the currently selected options
PopulateCollisionData	Used to populate the sysman_dd_mrg_collision table with the collision data for the specified merge
CreateDBLink	Creates the database link that will be used to merge data between source and destination databases
PopulateRptTotals	Used to populate the sysman_dd_mrt_rpt_total table with data that is used for the Completion report
KillRemoveJob	Used to kill the database job of the currently running db merge
ResetNoticeOfCompletionJob	Used to create the noticeOfCompletion database job that moves audit and lookup values to the archive tables
ResetConvertCPLongsJob	Used to create the ConvertCPLong database job that converts the LONG datatype columns to CLOB column
AuditMerge	Called by the UI to populate the progress bar and display current activity being performed by the merge

Figure 4 - DBMerge Package Specification

The following diagram lists the main stored procedures that get called when the user initiates a merge in the UI:

RestDBMergeJob	Called by the UI to begin merge process. Creates the database job that calls StartProcess for the specified row in sysman_dd_mrg_sow_noc
StartProcess	Entry point for the merge checks to ensure options are valid, creates log entries, handles pre and post cleanup logic
PreProcessUserSelect	Repopulates the sysman_dd_mrg_rule table with rows from sysman_dd_mrg_rule_pr replace holding variables with actual values specified by user i.e. ^MODULE^ with 'CATH'.'EP'
ProcessUserSelect	Handles the logic of calling processGroup for the current option(s) specified by user i.e. USERS, EMPLOYEES, SYSTEM MANAGEMENT etc.
ProcessGroup	Calls ProcessGroupTable for each sub group of the group being processed based on entries in sysman_dd_mrg_group_rel. For instance SYSTEM MANAGEMENT group is made of up Inventory, Billing, Registry etc.
ProcessGroupTable	Calls ProcessTableRows for each table for the current group being processed based on rows in sysman_dd_mrt_tab_group
ProcessTableRows	Handles the logic for processing the given table including constructing dynamic SELECT statement, running table level rules, commit logic, etc.
ProcessRow	Retrieves the data from the source server, calls CheckColumnForRule and creates the DML statement used to populate the row on the destination server
CheckColumnForRule	Runs the rule for the specified table\column, looks up new id for PK/FK in the sysman_dd_mrg_lkup_key, calculates new sequence ids, gets next counter value, etc.

Figure 5 – DB Merge Package Specification - Continued

For the current version of the DMS application there are over 7000 database objects that will need to be taken into consideration when creating the DB Merge application. The following table shows the breakdown of the Oracle database objects:

Object Type	Count
Tables	1681
Indexes	2514
Triggers	1291
Views	472
Sequences	658
Stored Programs (Packages, Procedures Functions)	318

Table 1 - DMS Database Objects

Merge Utility

Overview

The Merge utility is the UI component users will utilize to perform the merge. The utility will be a tab deck that houses and controls the execution of the following components:

- SelectionComponent
- MappingComponent
- MergeComponent
- DatabaseComponent
- ReportGenComponent

Dynamics

The Merge utility will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

The Merge utility manages the sequential execution of all components/processes by ensuring the following are met:

- Ensure merge selections are made before mappings
- Ensure mappings are made based on merge selections before sign-off
- Ensure user has completed sign-off before merge can start
- Allow mapping to be revisited after a merge complete if new mappings are needed (see MappingComponent and MergeComponent section for more details)
- Allow merge processing to be revisited after a merge complete if new mappings are needed (see MappingComponent and MergeComponent section for more details)
- Ensure systems are compatible and meet merge system requirements

The following are additional requirements the utility must meet:

- Logging and displaying errors to the user
- Allow the merge utility to be resumed at any given point – due to user shutting down the utility or any type of failure (reboot, loss of power, etc)
- The customer must be able to change the Merge password
- Save the current users specified server\location upon exit of the utility so when they reopen the utility that server\location will be defaulted for them

Sequence Diagrams

The diagram below is a high level depiction of the Merge utility as the controller.

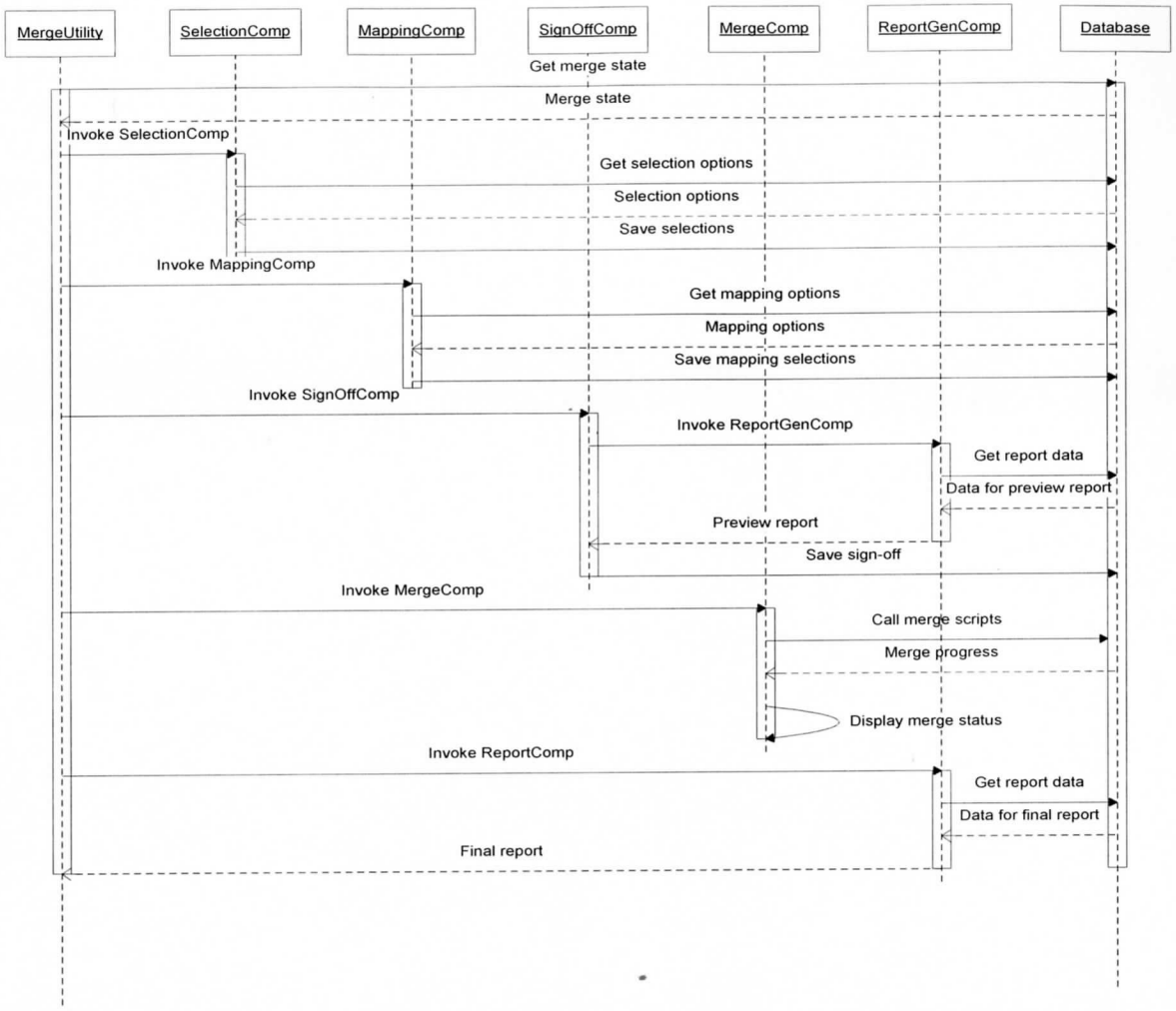


Figure 6 – Sequence Diagram

Serviceability

The Merge utility will log all exceptions that occur for troubleshooting. The Merge utility will display any exceptions to the user deemed necessary.

Selection Component

Overview

The selection component allows the users to specify the server and locations used in the merge, as well as the areas of DMS that will be merged

Dynamics

The selection component will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

- Allow users to specify the Source and Destination Servers
- Ensure Destination-server and Source-server can communicate
- Check DMS Versions to ensure they are the same before any merge steps
- Check Server types to ensure they are a supported merge type before any merge steps
- Populates Location dropdowns based on available locations from each server
- User must be able to select location to process the merge from Source-server
- User must be able to select location on Destination-server to process to or create a new location
- User must be allowed to specify the Statement of Work has been signed
- All other areas of the utility must be disabled until the Statement of Work has been signed and specified in this component for these servers/locations
- Users must be able to enter the Customer Champion and DMS Migration Champion and enter their corresponding email address that will be used by the merge application to send out status updates
- Users must be allowed to save these entries
- Users must be able to switch between multiple Server/location/Merge combinations
- When user selects a server\location combination that has a statement of work signed, the system must enable the remaining parts of the utility. If the statement of work has not been signed and specified the remaining parts of the utility must be disabled.
- System will check if the current user has been in the utility previously and if so, default the server\location they used the last time they were in the utility
- Display possible selections in DMS to be merged

- Allow users to specify if they want unconfirmed procedures to be auto confirmed by the merge utility
- Allows users to specify which functional areas in DMS they want merged
- Record user selections to sysman_dd_mrg_selection table using current servers\locations
- Utilize MMS_MODULES to aid in determining selection criteria and server types.
- Make selections dynamic - use DB Model to display selection options.
- Must provide on-screen descriptions of the options

Serviceability

The Selection component will throw exceptions in the case of an error to be logged by a containing component – the Merge Utility.

Mapping Component

Overview

Customers will utilize the MappingComponent after the merge sections are made. Mappings are necessary in order to rectify potential data collisions that could occur during the merge of data from one server to another. The component will present like data sets (Patient Information, Employees, etc) from both the Source-server and the Destination-server. Once presented with the data needing mapping, users can associate corresponding data between the two servers. The user may also be required to enter additional data in order to avoid collisions. These mappings will be utilized during the merge process to allow for collision-free data merges.

Example 1:

Data in Source-server		Data in Destination-server	
MRNO/Location	Patient Name	MRNO/Location	Patient Name
123/SVH	John Doe	123/SVH	Jane Doe

In example 1, if the merge would try to import John Doe into the new system, Jane Doe's attributes such as DOB, sex, SSN, etc would be updated with John Doe's attributes. In this instance we would want the user to perform one of the following actions:

1) Verify the 2 patients are the same
2) In DMS on Source-server – perform merge MRNO if the MRNO is incorrect on the Source-server.

3) In DMS on Destination-server – perform merge MRNO if the MRNO is incorrect on the Destination-server.

Example 2:

Data in Source-server		Data in Destination-server	
MRNO/Location	Patient Name	MRNO/Location	Patient Name
123/SVH	John Doe	456/SVH	John Doe

In example 2, John Doe would be created as a new patient without the mapping to 456/SVH. In this instance we would want the user to perform one of the following actions:

1) Verify the 2 patients are the same
2) If patients are same and 123 is the MRNO the customer wants to keep, the customer would utilize DMS to perform MRNO merge of 456 to 123.

Another important function of the MappingComponent is to record the expected processing state of patients and admissions. The states are simply INSERT or EXISTS (no updates occur if the record exists). The DatabaseComponent utilizes these states to determine if addition user mappings are needed after the merge completes – see DatabaseComponent section for details.

Dynamics

The MappingComponent will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

- Records mappings to sysman_dd_mrg_lkup_key table
  - Check for collisions and ensure mappings are up to date - give warning if there are still collisions
- Must provide on-screen descriptions of the mapping options
- Provide default filtering to narrow selections
- Indicate matches that are considered mismatches (PID mismatch, Employee, etc)
- Allow user to change filters to find like selections
- Log all mappings
  - Indicate the mismatches in the log
  - Mark patients, admissions and procedures with a state (insert or exists)

- Must be able to print currently displayed data, so user can take the report back to DMS and make the appropriate modifications (Merge MRNO, etc)

The following table lists the different areas needing mapping based on merge type:

Area	Merge Type
MRNO	Standalone to Express/Same Location
Admission	Standalone to Express/Same Location
Employees	ALL
Registry cut-over dates (Present user with warning if cutover dates do not match they will have to perform sync)	ALL

Table 2 – Mapping Areas

Serviceability

The MappingComponent will throw exceptions in the case of an error to be logged by a containing component – the Merge Utility.

Merge Component

Overview

The MergeComponent presents users with a summary of the areas to be merged and a notification that this is the final step before starting the merge process - any selections and mappings are final, and the merge process is a onetime event – with the exception of updating new patient/admission collisions encountered during the merge. It is also the area where users can initiate the actual data merge and to indicate the Notice of Completion has been signed – the final step in the process.

There can only be one active merge going on in the database at one time. The user can have several Selections/mappings going on at any given time for different servers/locations, but there can only be one merge.

Dynamics

The MergeComponent will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

- Utilize ReportGenComponent to generate Completion report
- Notify user they need to shutdown clients and interfaces – redirect processing to Destination-server
- Display a summary of:
    - User selected areas to merge
    - Present mismatches (PID, Employees, etc)
    - Present which procedures will not process in preview report (StudyInstance UID collisions – there is no way to rectify)
    - Patients that have not been resolved and will not be merged – must resolve at some point and re-run merge process
- Display notification (final step, all selections and mappings are final, sign-off coincides with Statement of Work, etc)
- Must update TNSnames file with Source-server selection
- Must create/update Source Server database link
- Ensure all work queues are clear before starting merge (work queues are not merged)
- Stop all services on Source Server (once queues are clear – ADT, ML, etc)
- Creates database job to perform merge - calls mms.DBMerge.ResetDBMergeJob
- Displays status of merge process to user – updates on a regular basis (not real time)
    - Time estimation
    - Table name and number (EX: Demog 5 of 245)
    - Row number (EX: 455 of 54073)
- Allow users to indicate Notice of Completion was signed. This moves the current set of mappings/selections/audits, etc to a Legacy status and permits the next merge to begin
- Ensure that another merge job cannot be created if there is already a merge job in the database
- Ensure that a merge job can only be created for the Servers/Locations that have the status of CURRENT in the sysman_dd_mrg_audit table

The following table lists the available sections that can be specified for the current run of the merge:

Selection	Description
Users	Creates users and their corresponding securities (roles, privileges)
Employees	Creates all employee information records
System Management	Creates all system management records including inventory, billing, list management etc.
Patients	Creates all patient records i.e. demographic, medications, allergies, etc.
Admission	Creates all admission records including physician, room, bed, etc

Procedures	Creates all procedural records, including reports and images
Registry	Creates all registry records including ACC, ICD, and STS

Table 3 - Merge Selection Options

Serviceability

The MergeComponent will throw exceptions in the case of an error to be logged by a containing component – the Merge Utility.

Database Component

Overview

The DatabaseComponent will be called after the sign-off is complete. The database component is designed to process all system management data and then process patients and admissions. After the patients and admission have been created, it then loops through each patient that was successfully brought over and brings their related procedural and registry data over one by one. If the state of a patient/admission changed during the merge process as indicted by the mapping process, the information will not be processed. A state may change from "INSERT" to "EXISTS" due to newly imported data as the Destination-server can be live and requires no down time. After the database completes for a server/location, system management is considered complete and NO further processing will take place for system management data. If a patient or admission was unable to process due to a state change, mapping can be resumed and the merge process can be attempted again until all patients are processed.

Dynamics

The DatabaseComponent will be a PL/SQL Package written in the Oracle 10g framework.

Specific Functionality

- Indicate status of merge process to UI – updates on a regular basis (not real time)
  - Table name and number (EX: Demog 5 of 245)
  - Row number (EX: 455 of 54073)

- Inform UI the interfaces can be turned on for the Destination-server even though the merge has not completed. The Destination-server for standalone merges can be fully operational during the entire merge process.
- Processes patients and all related information (admission and cases) one patient at a time
- System Management data can only be processed once for a Source-server/location
- Can be run over and over for patients that have been unable to process due to conflicts for a Source-server/location
- Patient and Admission level data will only be inserted if they do not exist, no updates of this data will occur in the Destination server.
- Patients that have a temporary MPI(TEMP321) assigned will get a new temporary mpi created when the data is brought over
- Admission records that have a temporary account number (TEMP21) will get a new temporary account number created when the data is brought over
- Create a database link that will point to the source DMS database and will be used to query the data that will be brought over to the destination database. All processing will occur on the destination database and the database link will only be used to query the data on the source database.
- The merge process needs to be initiated by a database job so the UI can close and not kill\halt the process
- Provide a procedure that the UI can call to kill the database job if the job appears to be hung or is creating numerous errors.
- Recreate the statistics for the merged tables after they have been merged to ensure query performance is not affected.
- Data will be processed based on any rules that have been defined for the table and or column. If no rule exists for the current column being processed than that column will be brought over to the destination database as it is in the source database.

Rule	Description
UK	Columns that make up the unique key for
WHERE CLAUSE	Use the specified where clause to pull the data from the source database
PRECONDITION	Run the specified code before processing this table
POSTCONDITION	Run the specified code after processing this table
ORDER By	Use the specified order by clause to pull the data from the source database
INSERT ONLY	Insert rows that do not exist, do not update any existing data
DISABLE TRIGGERS	Disable\enable triggers before\after processing this table
COMMIT ROW COUNT	Commit after specified number of rows have been processed for this table
SAVE NEW PK	Save the new primary key(s) for the rows in this table
CHECK ROW EXISTS	Specifies if the code needs to check the row exists - used to speed up processing

Table 4 - Table Level Rules

Rule	Description
PK	Specifies the column as the primary key
FK	Specifies the column is a foreign key
Ignore	The column should not be processed
Get Max + 1	Get the next available number for this column in the destination database
Null When No Lkup	Set the column to Null if the lookup Key is not found
Proc ID Lkup	Get the procedure ID for the procedure being processed
HardCode Value	Field needs to be populated with specified value i.e. process_flag
Sequence	Use the specified sequence generator to populate this column
IGNORE LOB	Do not process this LOB column

Table 5 - Column Level Rules

Serviceability

The DatabaseComponent will throw exceptions in the case of an error to be logged by a containing component – the Merge Utility.

ReportGen Component

Overview

The ReportGenComponent is utilized to generate the preview report selections and the final report displayed once the merge completes.

Dynamics

The ReportGenComponent will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

- Generates both preview (at signoff) and final reports (after merge completes)
- Reports are grouped by merge date/process time – This should be a single time that coincides with a signoff time – need to be able to link what was merged back to what was selected.
- Reads sysman_dd_mrg_audit table for summary
- Displays final summary report of the merge actions
- All reports must display the user that made the selections/mappings
- Final report must display to the user:

- Tables and records succeeded in merge
- Tables and records that failed during merge – failure reason

Serviceability

The ReportGenComponent will throw exceptions in the case of an error to be logged by a containing component – the Merge Utility.

Merge Utility Install

Overview

The merge utility install will install the merge tool.

Dynamics

The Merge Utility install will be a C# assembly written in the .Net 2.0 framework.

Specific Functionality

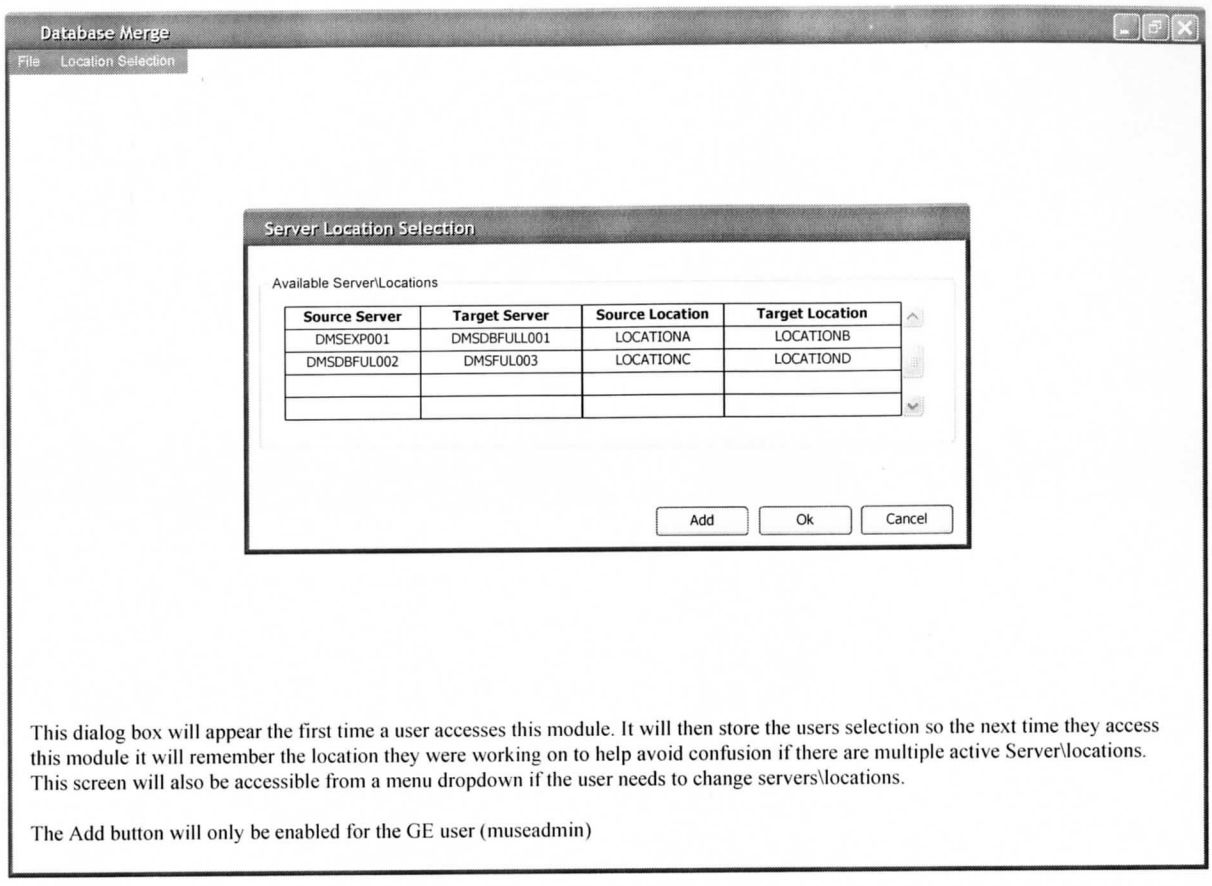
- Merge utility can be installed on Destination-server or client
- Update the Oracle network configuration files with Destination-server information
- Install merge database scripts
- Install merge utility

Serviceability

The install will throw exceptions in the case of an error

Screen Mockups

Selection Component – Select Servers\Locations



This dialog box will appear the first time a user accesses this module. It will then store the users selection so the next time they access this module it will remember the location they were working on to help avoid confusion if there are multiple active Server\locations. This screen will also be accessible from a menu dropdown if the user needs to change servers\locations.

The Add button will only be enabled for the GE user (museadmin)

Figure 7 – Server Location Screen Mock-up

Selection Component – Add Servers/Locations

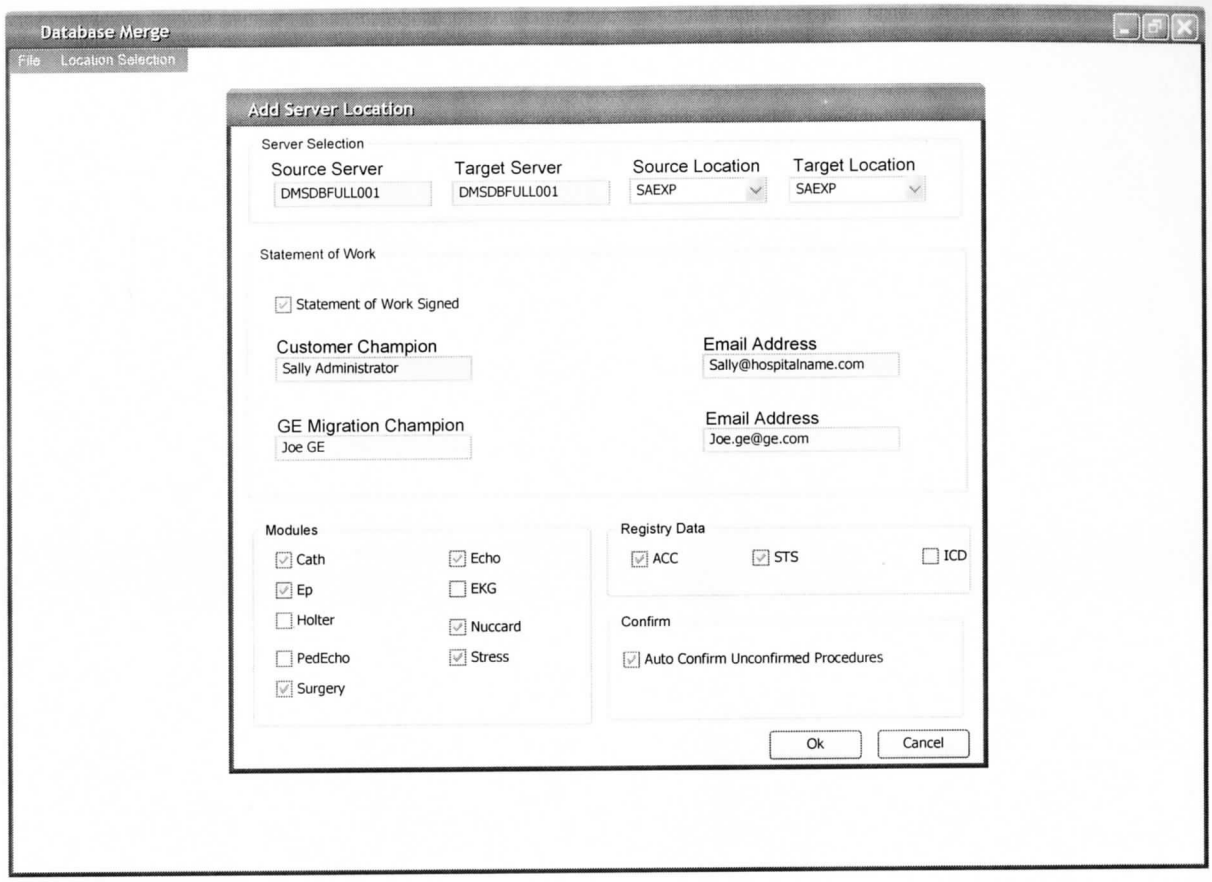


Figure 8 – Add Server/Location Screen Mock-Up

Mapping Component

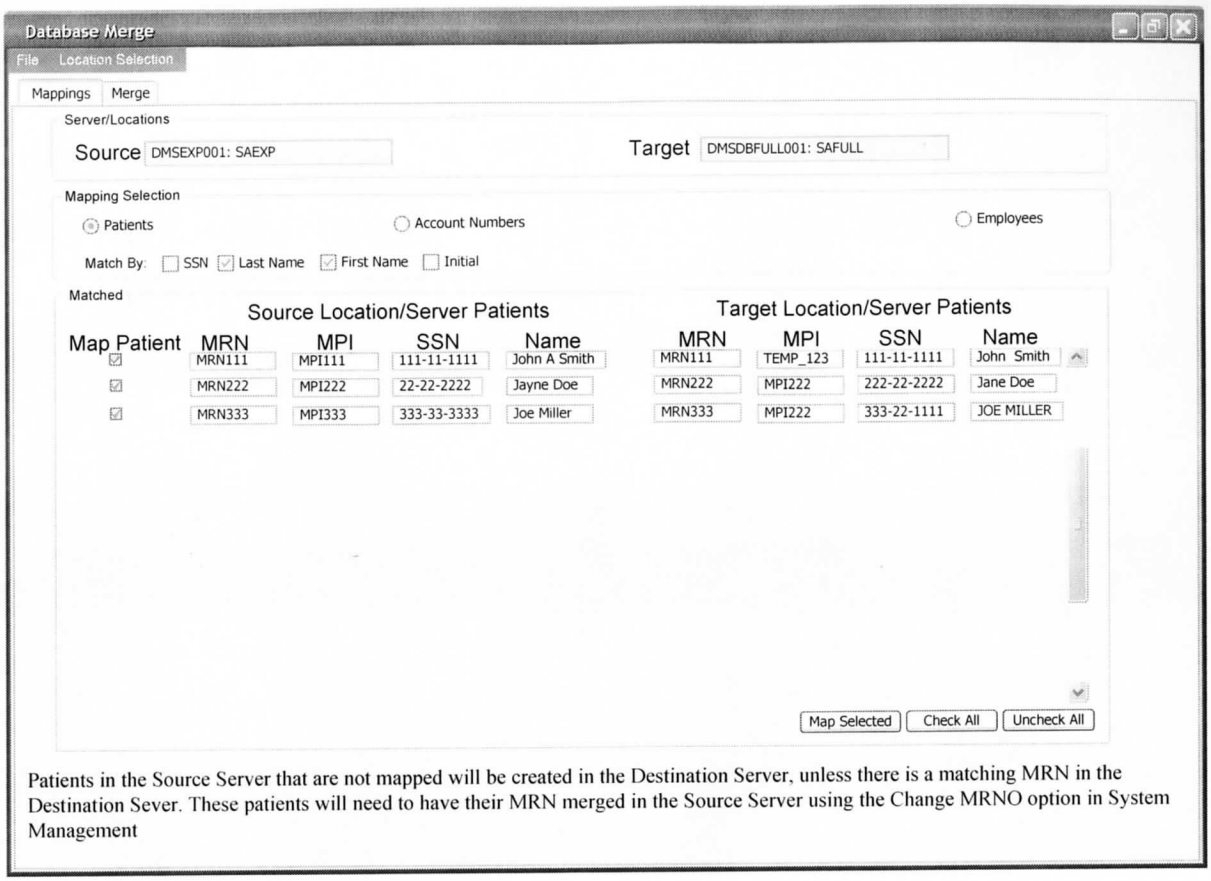


Figure 9 – Mapping Screen Mock-Up

Merge Component

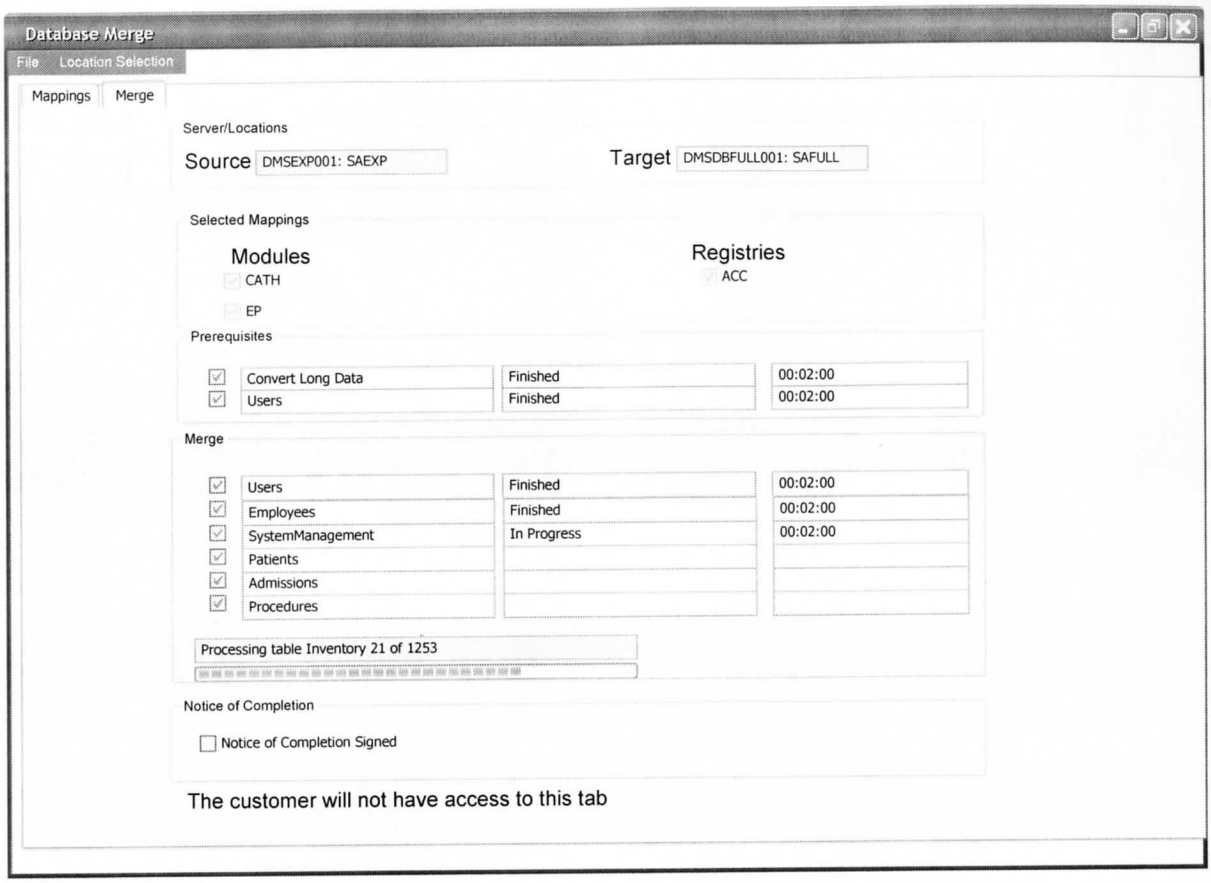


Figure 10 – Merge Screen Mock-Up

Statement of Work and Notice of Completion

Statement of Work

A Statement of work will be presented to the customer at the beginning of the merge process – before the merge utility is installed. The Statement of work will set the customers' expectations and will outline the merge process. It is not meant to detail the tasks of the merge, but rather explain the merge process and explain the details will be presented as the selections are made in the Merge Utility, and that the sign-off/preview report portion of the merge is the electronic extension to the Statement of Work.

All parts of the Statement of Work need to be addressed in individual merges to ensure all aspects of the merge are documented and agreed-upon.

Notice of Completion

Once the database merge is complete, a Notice of Completion will be presented to the customer. If the customer is satisfied with the merge then their acceptance will be captured through the Notice of Completion and the database merge will then be complete. If the customer is not satisfied, then the customer's needs will need to be reviewed along with the Statement of Work, Preview Report and Final Reports to see where the gaps are and how they can be filled.

All parts of the template need to be addressed in individual merges to ensure all aspects of the merge are documented and agreed-upon.

CASE STUDIES

Desert Springs Case study

Desert Springs Hospital located in Las Vegas, NV was founded in 1971 and currently has 286 beds. The hospital is known by local residences as the "Heart Hospital" because of its solid reputation as the leader in cardiac care. It has also been nationally recognized for its leadership in diabetes, cardiac, stroke and bariatric surgery programs. The hospital is accredited by the Joint Commission and is affiliated with the Federation of American Hospitals and the Association of Western Hospitals as well as the Nevada Hospital Association (http://www.desertspringshospital.com/About-the-Hospital). Desert Springs has been a long-time GE Healthcare customer and user of the DMS application.

Desert Springs Hospital was recently acquired by The Valley Health System which currently is comprised of the Centennial Hills Hospital, Spring Valley Medical Center, Summerlin Medical Center, and Valley Hospital. These hospitals were also GE Healthcare customers of the DMS product and each had their own DMS server and database. Since the formation of The Valley Health System, they have requested that the DB Merge application be run for each hospital so they can combine the data from the separate databases into one database\server located in their newly formed data center.

Desert Springs is the smallest of the hospitals in terms of DMS data and was selected to be the pilot hospital to be merged into the main DMS system using the DB Merge application. It was decided that each of the hospitals would have its own location in the DMS application which will allow them to separate the data easier for reporting purposes. Since each hospital would be merged in their own location, there would be no issues of data collisions in regards to employees, patients or admissions. This made the merge very easy on the hospital staff since they would not need to reconcile any of the data collisions. The data for employee, patients and admissions would come in very cleanly and would not require any intervention on the hospital staff.

The DMS Merge specialist met with the Desert Springs staff and formulated the plan for the merge. They agreed upon the terms of the merge and the Statement of Work was signed by both the hospital staff and the merge specialist. The hospital staff wanted all of their CATH and EP procedural data to be merged over to the new destination server as well as their ACC Registry data. The merge specialist then installed the DB Merge utility on the new destination server and ran the DB Merge utility and configured the merge to the customer's specifications.

The actual merge was scheduled to begin on the following Monday morning. The merge specialist began the merge of the Desert Springs hospital data to the new destination server by merging the System Management data, which included 4500 inventory records. The System Management portion of the merge finished in around 7 minutes and no errors were encountered. After the System Management data was successfully merged, the merge specialist moved on to the Patient data. The Desert Springs DMS Database contained over 16,000 patient records and they were merged over successfully in 10 minutes. Next the merge specialist began the merge of 39,000 Admission level data records. The admission data was merged in a little over 15 minutes with no errors. Now that the system management, patient and admission data was successfully merged to the new destination server, the merge specialist could move on to the procedure information. The procedure data contained over 3,000 records and was completed in around 15 minutes. The reason the procedure data takes more time is that there is much more data related to a procedure than there is to a patient. Also, there are images and reports (PDF files) that are stored as BLOB's and BFILE's in the database and those LOB objects take time to merge between databases. Once the procedural data merge completed, the merge specialist kicked off the final portion of the merge, the Registry data. Desert Springs had a little over 2,500 registry records and they were merged to the new database in around 6 minutes. The merge specialist then ran the Completion report and gave it to the hospital administrative staff for their review and approval.

All the data from the Desert Springs DMS database was now located in the new destination database in the data center. The administration team from Desert Springs logged into the new

system and verified the data and found no issues with the merge. All their employees, patients, admission and procedural data were there as it was in the Desert Springs database as was specified by the Completion report. They were very happy with the results and pleased with how quickly the merge was able to finish. While the merge was running there was no disruption in service on the destination database. The hospital staff was able to go about their day to day activities using the DMS system and did not notice any issue or performance problems. The Desert Springs DMS system was taken off-line and they have begun using the DMS application on the new destination server located in the data center. The Notice of Completion form was signed by both the hospital administrator and the Merge specialist. They have since begun planning for merging the remaining hospitals DMS systems and expect the same satisfactory results.

CONCLUSIONS

The DB Merge utility has been successfully ran at several customer sites since it has been released to production. All of the sites have been smaller in data size so the merge was able to complete in a day or two. GE Healthcare has hired a Merge Specialist and they work with the customer and actually perform the merge. The Merge Specialist has worked closely with the developer on the first few merges to iron out any difficulties encountered. Most of the issues encountered have had to do with network issues. The main problems encountered were dealing with connectivity between the two servers. Once the network issues were ironed out, the application was able move the data between the two servers successfully.

The Merge Specialist has had a couple of suggestions to improve the UI and at the present time the suggestions are on hold as they were deemed minor annoyances. Also, since the customer is not affected by the changes, there is no real benefit to changing the code at this time. But all in all, the merge specialist is very happy with the application. They like the fact that they can start the merge and not have to leave the application open to monitor the progress. They can go about other business and once the merge completes, they get the email notification and then they can go back to the site and proceed with the next step. They also like the flexibility that application provides. They can run just the portions of the application that relate to the customer's site. Also, they can run one section at a time as it fits into their schedule. That is why the merges that have been run so far have spanned more than one day. They are just running portions during the work day, and starting again on the second day. If they were to run the merge straight through, it would have finished in around 8-10 hours. Customers are also happy with the results and the fact that they no longer have to support and maintain multiple systems, and can get at all their data\reports from one server.

Obviously as the underlying structure of the database changes, new entries will need to be added in the SYSMAN_DD_MRG_RULE table to facilitate the merging of the new tables\columns. This should be all that is needed to handle merging new data for new schema

entries that get added to the DMS system. This will make the application easy to maintain as there should be no code changes required to handle the new schema, only data changes. This makes the application very flexible and allows for faster turnaround times with each new release of the DMS product.

REFERENCES

APPENDICES

APPENDIX A: WORK BREAKDOWN

ID	WBS	Task Name	Duration	Start	Finish
1	**DBMerge1**	**DBMerge**	**36 days?**	**Wed 5/12/10**	**Wed 6/30/10**
2	**DBMerge1.1**	**Create Database Schema**	**1 day?**	**Wed 5/12/10**	**Wed 5/12/10**
3	DBMerge1.1.1	Create Selection tables and objects	1 day	Wed 5/12/10	Wed 5/12/10
4	DBMerge1.1.2	Create Process tables and objects	1 day?	Wed 5/12/10	Wed 5/12/10
5	DBMerge1.1.3	Create Group tables and objects	1 day?	Wed 5/12/10	Wed 5/12/10
6	DBMerge1.1.4	Create Rule tables and objects	1 day?	Wed 5/12/10	Wed 5/12/10
7	DBMerge1.1.5	Create Log tables and objects	1 day?	Wed 5/12/10	Wed 5/12/10
8	**DBMerge1.3**	**Create Stored Program Units**	**29 days**	**Wed 5/12/10**	**Mon 6/21/10**
9	DBMerge1.3.1	Setup Procedures	5 days	Wed 5/12/10	Tue 5/18/10
10	DBMerge1.3.2	Process User Selection procedures	2 days	Wed 5/19/10	Thu 5/20/10
11	DBMerge1.3.3	Process Group procedures	2 days	Fri 5/21/10	Mon 5/24/10
12	DBMerge1.3.4	Process Group Table procedures	2 days	Tue 5/25/10	Wed 5/26/10
13	DBMerge1.3.5	Process Table procedures	3 days	Thu 5/27/10	Mon 5/31/10
14	DBMerge1.3.6	Process Data Row procedures	5 days	Tue 6/1/10	Mon 6/7/10
15	DBMerge1.3.7	Process data column rule procedures	10 days	Tue 6/8/10	Mon 6/21/10
16	DBMerge1.3.8	Information/Error Logging procedures	3 days	Wed 5/12/10	Fri 5/14/10
17	DBMerge1.3.9	System Status procedures	3 days	Wed 5/12/10	Fri 5/14/10
18	DBMerge1.3.10	Process cleanup procedures	2 days	Wed 5/12/10	Thu 5/13/10
19	DBMerge1.3.11	DB Job creation/kill procedures	2 days	Wed 5/12/10	Thu 5/13/10
20	DBMerge1.3.12	Data Collision procedures	5 days	Wed 5/12/10	Tue 5/18/10
21	**DBMerge1.5**	**Create UI**	**36 days**	**Wed 5/12/10**	**Wed 6/30/10**
22	DBMerge1.5.1	Logon screen	3 days	Wed 5/12/10	Fri 5/14/10
23	DBMerge1.5.2	Merge Selection Screen	3 days	Mon 5/17/10	Wed 5/19/10
24	DBMerge1.5.3	Merge Creation Screen	5 days	Thu 5/20/10	Wed 5/26/10
25	DBMerge1.5.4	Data Collision Screen	10 days	Thu 5/27/10	Wed 6/9/10
26	DBMerge1.5.5	DB Merge Management Screen	15 days	Thu 6/10/10	Wed 6/30/10

Table 6 - Work Breakdown Structure

APPENDIX B: GANTT CHART

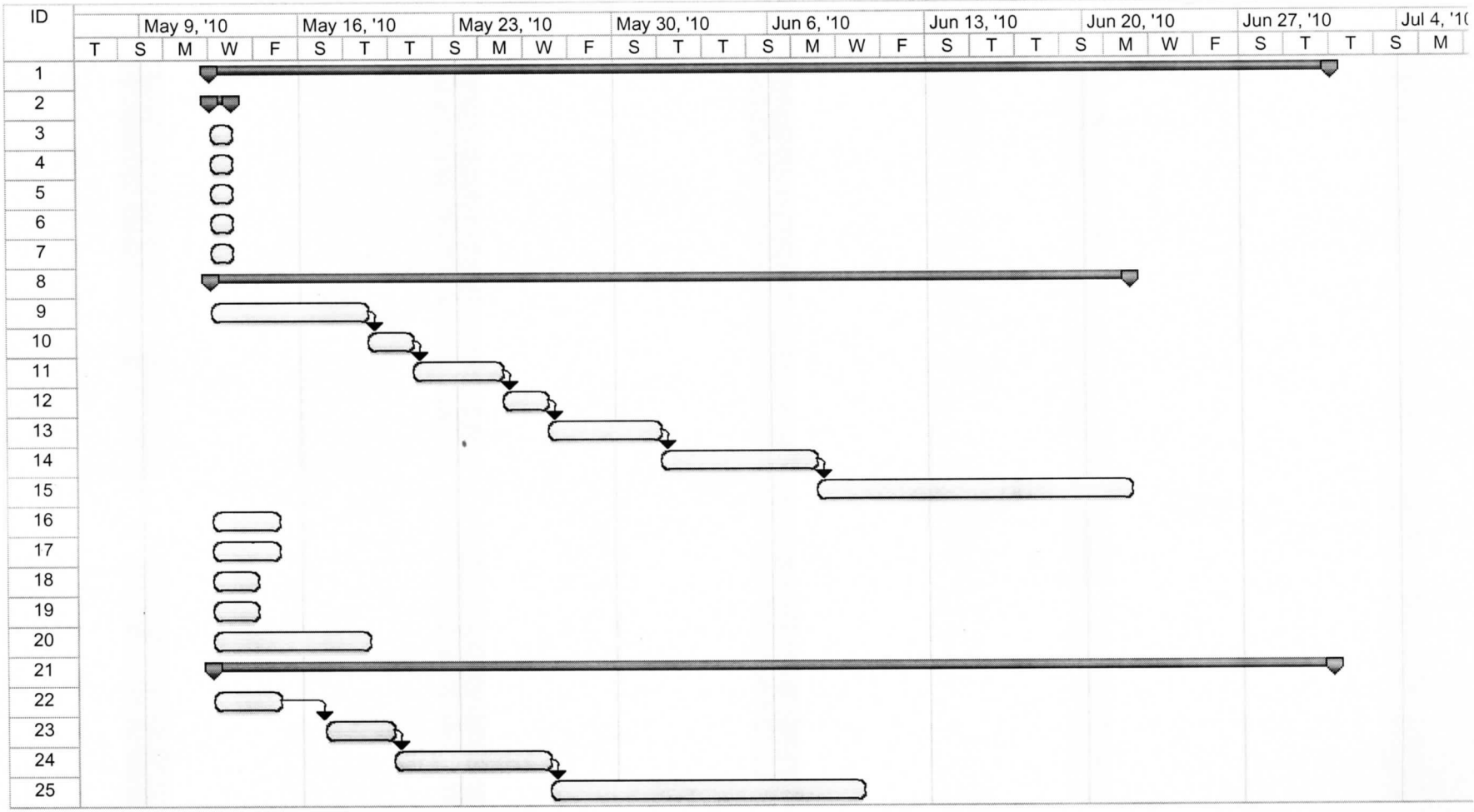


Figure 11 - Gantt Chart

APPENDIX C: ER DIAGRAM

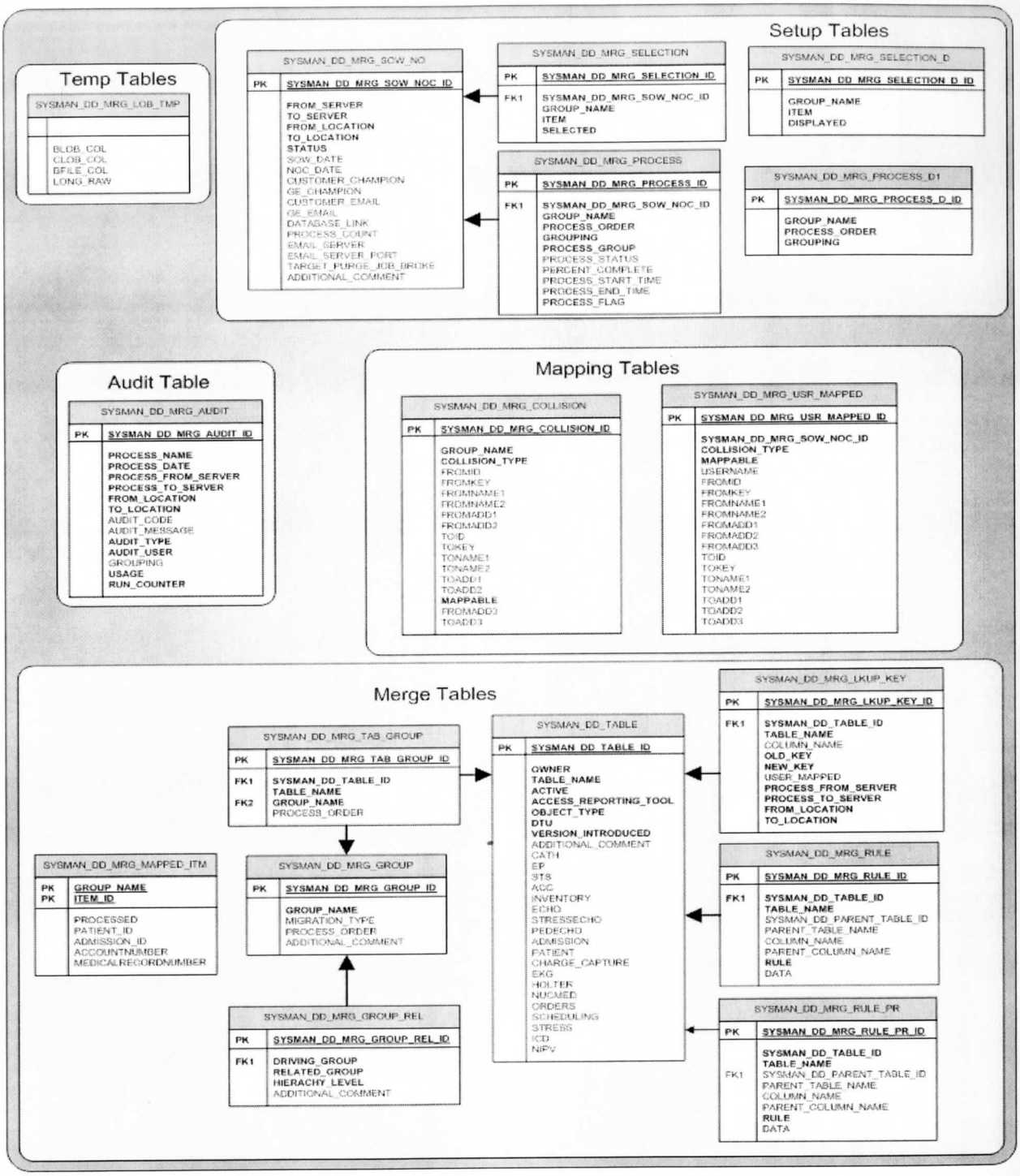


Figure 12 – ER Diagram