MGI DB Migration

Author: Lori E. Corbani

Created: June 27, 2001

Last Modified: October 2, 2001 12:23

1 Purpose of Document

This document describes the design of the MGI DB Migration product, *mgidbutilities*.

2 Introduction

The MGI DB Migration product is responsible for converting a Sybase database schema from one version to another.

MGI DB Migration depends on the MGI DB Schema and MGI DB Permission products for various databases. An installation of a MGI DB Schema or Permission product is configured for a specific database instance residing on a specific database server (for example, database *mgd* residing on server *MGD*).

MGI DB Migration covers the schema migration for *any* MGI Sybase database.

3 Definitions

See the MGI Database Administration Overview's Definitions section.

4 Overview

A "migration" is the series of steps necessary to convert a database schema from one version to another. A migration contains one or more of the following:

1. changes to existing database objects (tables, stored procedures, views, triggers, defaults, rules, keys, indexes)
2. addition of new database objects
3. changes to existing data (data cleanup, adding data for new tables, etc.)

There are a couple of ways in which we can migrate from one schema version to another:

1. Apply migration scripts directly to the production database.
2. Apply migration scripts to a copy of the production database, backup the migrated copy and load the backup onto the live production database (thereby replacing the "old" production database with a "new" copy).

99% of the time we utilize the first method of applying the migration scripts directly to the production database.

NOTE: We always backup the production database prior to running any migration.

5 Design and Implementation

1. There is one MGI Migration product. This product addresses migration needs for all MGI databases.
2. This product provides functionality for migrating from one schema version to another for a given set of MGI Databases. In all cases the migration is from a current production schema version to a new schema version and is usually part of a MGI software release (which includes changes to other MGI products).
3. Each migration is separate and distinct from previous and future migrations. That is, the migration for MGI2.6 to MGI2.7 is distinct from the migration for MGI2.5 to MGI2.6 and from MGI2.7 to MGI2.8.
4. The goal of each migration is to move a schema from point A to point B. Point A is defined by an existing DB Schema installation and point B is defined by a new DB Schema installation (for example, mgddbschema-1-0-0 vs. mgddbschema-2-0-0).
5. The applied migration results in a schema which exactly matches the new MGI DB Schema product for which it is intended. If we are migrating from MGI2.6 to MGI2.7, then once the MGI2.7 migration is applied to the MGI2.6 schema, the resulting schema should exactly match the MGI2.7 DB Schema product.[1]
6. Each migration may apply to one or more MGI Databases (MGD, Nomen, Strains, WTS).
7. The name of the CVS product is *mgidbmigration*.
8. *Configuration* file

   A c-shell file with the following environment variables. This file is source-ed by any c-shell script in the product to initialize the runtime environment. It is located in the root directory of the product.

   - SYBASE - path of the Sybase installation (ex. /opt/sybase)
   - PYTHONPATH - path of MGI Python libraries (ex. /usr/local/mgi/lib/python)
   - PATH - path of system executables; includes $SYBASE/bin
   - DBUTILITIESPATH - path of the DB Utilities product
9. A directory for each distinct migration. These directories are named by either the public version number (MGI2.7) or by the WTS Tracking Record Number (TR2618). For example, migration directory MGI2.7 contains scripts for migrating from MGI2.6 to

1. In the future, we will provide an automated procedure for verifying a migrated schema.

MGI2.7. Migration directory TR2618 contains scripts for migrating data for TR2618 which is not part of a comprehensive MGI release.

10. Each migration directory contains c-shell scripts for each WTS Tracking record involved in the software release which requires schema migration. For example, the MGI2.6 migration directory contains scripts *tr1467.csh* and *tr2193.csh* which apply migration logic for TRs 1467 and 2193 respectively. These TR scripts may include calls to Python or C programs and execute BCP commands or SQL commands.
11. Each migration directory contains a wrapper script, *MGI.csh*, which performs the following functions:
    - Initializes the runtime environment by sourcing the *Configuration* file.
    - Sends all SQL and output generated by the migration to the *MGI.log* file.
    - Using the appropriate MGI DB Schema product(s), drops and re-creates all stored procedure, view and trigger objects. This is a good idea even if these objects aren't being modified, because there are inter-dependencies between objects. By dropping and re-creating these objects, we are sure to catch any inconsistencies or errors introduced by modifying any database object.
    - Using the appropriate MGI DB Permission products, revokes and grants the appropriate permissions on all database objects.
    - Executes the necessary TR scripts to perform the migration.
    - Sets the new schema version values using the DB Utility script *updateSchemaVersion.csh*.
    - Sets the new public version values using the DB Utility script *updatePublicVersion.csh*.
12. TR scripts can be executed individually or by the *MGI.csh* wrapper script.
13. All scripts log their output into a log file which resides in the migration directory.
14. All scripts can be executed by a *cron* or *at* job.
15. All scripts can be executed by any user in group *progs* on development, but only by user *sybase* on production.

6 Using the Product

1. If a CVS branch is needed (due to concurrent development on different schema versions), contact the CVS Administrator to help set up a branch tag.
2. Checkout the entire CVS product. This is necessary in order to add a new directory at the top level of the product for the new migration. It also provides you with examples of previous migrations. And, when tagging the product, it is necessary to have the entire product checked out.
3. Use the DB Schema and DB Perms products in the migration scripts. If you are altering the schema, then use the new DB Schema version that contains the schema

changes. If you need to refer to an existing schema, then use a DB Schema installation of the existing schema version.