Installing GUS at VBI

Brett Tyler's Lab
Virginia Bioinformatics Institute
Virginia Tech University
Blacksburg, Virginia
https://www.vbi.vt.edu

January 10, 2004

1 GUS Database Platform

The Genomics Unified Schema (GUS) database platform is composed of the GUS Application Framework and the GUS Web Development Kit. The Application Framework provides a strongly-typed relational schema and programs that facilitate the population of a genomics database. The Web Development Kit provides the foundation for implementing a query-based website for the database using the GUS schema.

Although the GUS platform should work on most computer platforms in use today, it has been tested and is supported only on the GNU/Linux platform. It currently works only with Oracle (8i or 9i).

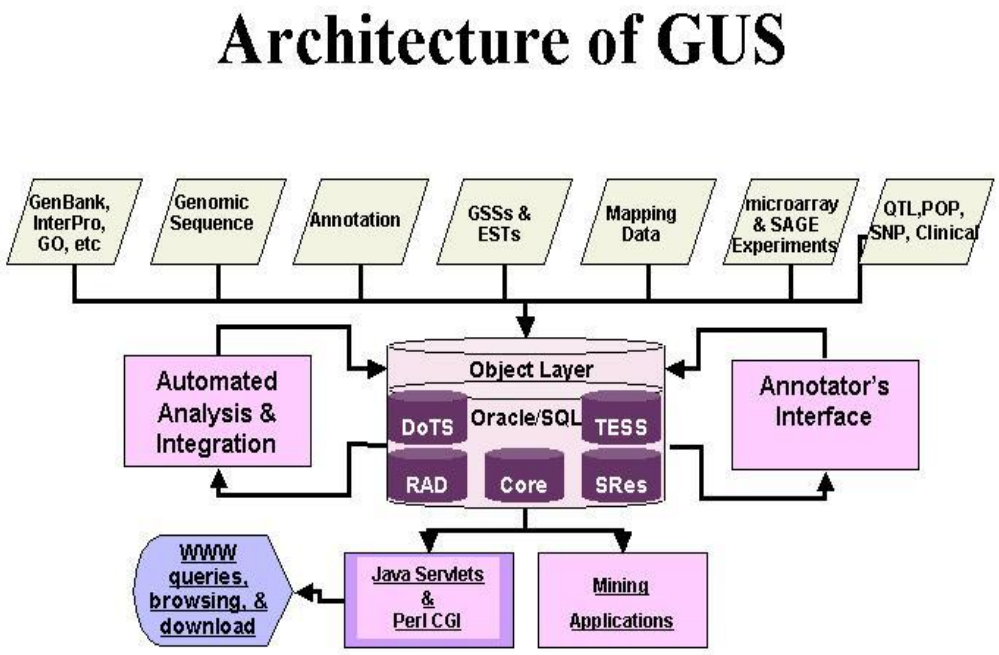

Figure 1: GUS Architecture (currently without permission from CBIL)

1.1 Input and Output Formats

The following are the types of input formats GUS3.0 can handle:

- DNA Sequence DBs: GenBank (main, dbEST, NRDB), TIGR
- Protein Sequence DBs: Swiss-Prot, CDD, ProDom, InterPro
- Expression: MAGE
- Ontologies: GO, SO, PATO

- Mapping Data: RH maps
- Gene Predictors: GLIMMER, GENSCAN, Phat, GeneFinder
- Sequence Alignment: BLAST, BLAT, SIM4
- Sequence Assembly: CAP4

The following are the types of format that may be exported out of a GUS database:

- FASTA
- MAGE
- DB table dumps
- DoTS assemblies

2 Requirements

2.1 Hardware

The GUS database platform hardware requirements depend primarily on the oracle hardware requirements and the intended size of the data warehouse to be implemented. The disk space requirements also have to take into account space for the OS, the GUS software, and all the ancillary software needed to run GUS. Thus, the minimum requirements are:

- Memory: 512 MB
- Swap Space: 1 GB
- Oracle Software Disk Space: 4.5 GB
- Oracle Working Space: 400 MB in the /tmp directory
- Oracle Data Warehouse:
- OS Software:
- Other Software:

2.2 Software

The following are the various software components needed to run the GUS database platform:

- Oracle 9i (or 8i)
- Perl 5.8.1 and the following modules:
  - DBI-1.38
  - DBD-Oracle-1.14
  - GD-2.11
  - Parse-Yapp-1.05
  - XML-Simple-2.09
  - XML-Parser-2.34 or XML-SAX-0.12
- Java SDKs:
  - J2SDK 1.4.2_02 (j2sdk-1_4_2_02-linux-i586.bin)
  - J2SDKEE 1.3.1 (1.3.1 FCS Release, January 31, 2002)
  - JWSDP 1.2
- Additional Java Packages: PerlTools 1.2
- Ant 1.5.4
- Apache 1.3.28
- Tomcat 5.0
- GUS Application Framework
- GUS Web Development Kit

3 Installing Oracle9i

An Oracle database instance was created on tuor (Sun server). Our Oracle client runs on Slackware 8.1 GNU/Linux systems that had been modified to fit the needs of the Virginia Bioinformatics Institute (VBI). It came with a 2.40GHz Pentium processor, 1.5G RAM, 3.9G swap, and runs the linux 2.4.22 kernel.

3.1 Oracle Users Immediately After Oracle DB Creation

```
SQL> select * from all_users;

USERNAME                          USER_ID CREATED
------------------------------ ---------- ---------
SYS                                     0 24-OCT-03
SYSTEM                                  5 24-OCT-03
OUTLN                                  11 24-OCT-03
DBSNMP                                 19 24-OCT-03
ORDSYS                                 21 24-OCT-03
ORDPLUGINS                             22 24-OCT-03
MDSYS                                  23 24-OCT-03
CTXSYS                                 24 24-OCT-03

8 rows selected.
```

3.2 Changing Default Passwords for SYS and SYSTEM

```
SQL> select username, account_status, lock_date, created from dba_users;

USERNAME                       ACCOUNT_STATUS                   LOCK_DATE CREATED
------------------------------ -------------------------------- --------- ---------
SYS                            OPEN                                       24-OCT-03
SYSTEM                         OPEN                                       24-OCT-03
OUTLN                          OPEN                                       24-OCT-03
DBSNMP                         OPEN                                       24-OCT-03
ORDSYS                         EXPIRED & LOCKED                 24-OCT-03 24-OCT-03
ORDPLUGINS                     EXPIRED & LOCKED                 24-OCT-03 24-OCT-03
MDSYS                          EXPIRED & LOCKED                 24-OCT-03 24-OCT-03
CTXSYS                         OPEN                                       24-OCT-03

8 rows selected.
```

Change default passwords for sys, system, outln, and dbsnmp. Lock and expire the ctxsys account.

Change passwords by:

```
SQL> ALTER USER <username> IDENTIFIED BY <password>;
```

Lock and expire ctxsys account by:

```
SQL> ALTER USER ctxsys PASSWORD EXPIRE ACCOUNT LOCK;
```

To unlock accounts, if needed, need to use command:

```
ALTER USER <username> IDENTIFIED BY <password> ACCOUNT UNLOCK;
```

3.3 Assigning DBA Role to GUS Admin

```
SQL> GRANT dba TO gus identified by <password>;
```

4 Installing Perl Software

In order to find out what modules were installed, installed the pmtools suite, pmtools-1.00, which is obtained here. This tool includes the utility pmall, which will list all the modules installed on your system.

Found that the following modules had not been installed:

- GD-2.11
- Parse-Yapp-1.05
- XML-Simple-2.09
- XML-Parser-2.34 and/or XML-SAX-0.12

4.1 Installing Module GD

First had to install the C gd library. Obtain the source and follow the installation instructions.

Install module GD by:

```
$ perl -MCPAN -e 'install GD'
```

4.2 Installing Module Parse::Yapp

```
$ perl -MCPAN -e 'install Parse::Yapp'
```

4.3 Installing Module XML::Simple

Install module XML::Simple by:

```
$ perl -MCPAN -e 'install XML::Simple'
```

Installation process prompts to install required XML::SAX; say 'yes.' Installing XML::SAX prompts to install required XML::NamespaceSupport; say 'yes.'

4.4 Installing Module XML::Parser

```
$ perl -MCPAN -e 'install XML::Parser'
```

5 Installing the GUS Application Framework

These instructions assume that the following have already been installed or are running on your system:

- oracle
- oracle database instance
- java
- ant

5.1 Creating GUS Tablespaces

Here we create tablespaces for the GUS CORE, SRES, DoTS, TESS and RAD3 schemas, and index tables for each. We do this by modifying Terry Clark's tablespaces creation script. Our script can be found in Section Tablespaces Initialization Script.

5.2 Creating Required GUS Users in Oracle

There are two required users:

- GUSRW - needs read/write permissions
- GUSdevReadOnly - needs read pemissions

We create the user accounts by running a script (see Section Creating Required GUS Users Script) containing the create statements taken from Terry Clark's installation notes plus some modifications.

5.3 Granting Priviliges to GUS Users

To assign the required privileges to users GUSRW and GUSdevReadOnly, we execute a script (see Section Granting Privileges to GUS Users Script). This script is the same as that found in Terry Clark's installation notes.

5.4 Obtaining and Unpacking the GUS Distribution

Three projects (compressed tar files) are needed for the GUS installation; two comprise the GUS distribution and the third one is a required CBIL project. The projects and places to download them from are:

- gus.tar.gz from the Sanger Institute GUS CVS root page **or** get code using cvs by:

```
$ mkdir checkouts
$ cd checkouts/
$ cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/GUS login
$ cvs -z 3 -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/GUS co GUS
```

- install.tar.gz also from the Sanger Institute
- cbil.tar.gz from CBIL at the University of Pennsylvania

The three compressed tar files should be placed in the same directory and uncompressed and untarred by issuing the following command:

```
$ tar xvfzp <filename>.tar.gz
```

5.5 Setting Up Environment Variables to Install GUS

In order to install GUS, the GUS installation user needs to set a few environment variables. These include the locations of the places where the GUS distribution and CBIL project were unpacked (this location is $PROJECT_HOME), where GUS will be installed ($GUS_HOME), and of a gus configuration file, and the paths to some required language libraries, oracle home, and some required executables. The values of these environment variables for the VBI installation are found in Section Environment Variables Settings for Building GUS.

5.6 Configuring the GUS Properties and Install Files

The GUS properties file (.gus.properties) contains some settings that the GUS software requires in order to execute commands. A copy of the VBI file is found in Section GUS Properties File. The GUS install project contains a sample file and it is found at $PROJECT_HOME/install/gus.properties.sample.

If $GUS_HOME has not been created, create it now and create the .gus.properties file by either copying the sample file and editing it, or by cutting and pasting the file contents found in Section GUS Properties File. Place the .gus.properties file in $GUS_HOME.

The GUS install properties file (install.prop) contains a pointer to the location of the Perl executable on your system. See Section Install Property File for a copy of VBI's file's contents. The GUS install project contains a sample file in $PROJECT_HOME/install/config/install.prop.sample.

In the $GUS_HOME directory, create a directory called 'config' and create the install.prop file in that directory.

5.7 Building GUS

At this point, we are ready to build GUS. This consists of installing the install and GUS projects. We do this by executing a build perl script, which calls ant to do the actual build. As Terry Clark's installation notes point out, there is a bug in the GUS project build process executed by ant. The bug is due to the fact that there are two clauses in the $PROJECT_HOME/GUS/build.xml file that call for generating GUS objects before the database tables are installed. To get around this, the build process requires that one execute the build command three times as follows.

5.7.1 Running Build the First Time

Before executing the build command for the first time, do the following:

```
$ cd $PROJECT_HOME/GUS
$ cp build.xml build.xml-original
```

Now, edit build.xml to remove the following two clauses:

```
<exec executable="generateGusObjects"
       failonerror="true">
```

```
   <arg value="--javaOrPerl=java"/>
 </exec>
```

and

```
<exec executable="generateGusObjects"
       failonerror="true">
   <arg value="--javaOrPerl=perl"/>
 <exec>
```

In the version of our GUS project, the first clause starts on line 117 and the second one on line 193.

Now type:

```
$ build GUS install -append
```

5.7.2 Running Build the Second Time

During the process of running the build command the first time, $GUS_HOME and contents were created. At the end of the build run, we are asked to create the file $GUS_HOME/config/schema.prop and edit it appropriately.

Create the schema.prop file by typing:

```
$ cp $GUS_HOME/config/schema.prop.sample $GUS_HOME/config/schema.prop
```

Now, edit schema.prop with sid, host, password, table space, and index table space values appropriate for your site. A copy of our schema.prop file is found in Section Schema Property File.

Finally, execute the build command a second time by typing:

```
$ build GUS install -append
```

5.7.3 Running Build the Third Time

Before running the build command the third and last time, we have to do two things:

1. execute the $GUS_HOME/schema/oracle/create-db.sh script by:

   ```
   $ cd $GUS_HOME/schema/oracle
   $ create-db.sh | tee create-db.sh.out
   ```

   NOTE: In order to run the create-db.sh, you need to know the oracle SYS password as the first task in the script requires SYS as SYSDBA to run it.

   The create-db.sh script performs a number of tasks. The first few tasks involve running sql scripts to create new users, sequences, tables, and views. The next few tasks involve calling a Perl script several times to grant various permissions. Finally, a few more sql scripts are executed to insert some required data, and create key constraints and indexes.

Therefore, this step takes a long time to complete. It may be confusing once the first set of sql scripts complete and the grant permission Perl scripts start executing because it may appear like nothing is happening, especially because the shell displays a message reading that we have been disconnected from oracle. Please be patient and wait until the shell prompt comes back.

2. restore original $PROJECT_HOME/GUS/build.xml file:

```
$ cd $PROJECT_HOME/GUS
$ mv build.xml build.xml-without-objects-creation
$ mv build.xml-original build.xml
```

Now, execute the build command a third time by typing:

```
$ build GUS install -append
```

5.8 Updating Oracle to Match Values in GUS Properties File

During the build process, a number of rows were inserted with default values in various tables including the core.userinfo, core.groupinfo, and core.projectinfo. Before going on, the default values in these last three tables have to be updated to match the correponding values in the .gus.properties file (see Section GUS Properties File). Specifically, the values that need to be updated are:

- the value for login in core.userinfo has to match the userName value in .gus.properties
- the value for name in core.groupinfo has to match the group value in .gus.properties
- the value for name in core.projectinfo has to match the project value in .gus.properties

Additionally, the password in core.userinfo has to be updated accordingly.

NOTE: Although oracle does not distinguish case for login, password, and sql commands, the values in oracle that have to match the values in the .gus.properties file also have to match in case. That is because it is the Perl code in GUS that does a comparison of the values in oracle and the .gus.properties file not oracle.

5.9 Populating the Machine Table

One table that does not get populated with default values during the build process is the core.machine table. We need to populate it manually before we go on to the next step. The insert statement used at VBI to populate the machine table is found in Section Populating Machine Table.

5.10 Registering GA

An application programmer connects to GUS3.0 by using the GusApplication (GA; see Section GA Usage for information on how to use GA) Perl program. Specific tasks, such as loading taxonomy or genbank data, etc., are performed by individual plugins (see Section Plugins for more information on plugins), which are invoked by GA. Before using GA or any of the plugins, they have to be registered with the database. The registration process consists of storing the plugin's version number, its checksum, and other info in the database.

Before actually registering GA, we need to tell the GUS system the location of the program md5sum, used to calculate checksums. We do this by writing the md5sum location in the file $GUS_HOME/config/GUS-PluginMgr.prop. Create the file as follows:

```
$ cp $GUS_HOME/config/GUS-PluginMgr.prop.sample $GUS_HOME/config/GUS-PluginMgr.prop
```

Now, edit the file, if needed, to point to the correct location of md5sum on your system.

Finally, GA can now be registered with the command:

```
$ ga +meta --commit.
```

The output of a successful registration can be found in Section Output from Registering GA.

Note: From this point on, any insertions into the GUS database **must** be done using a plugin.

5.11 Registering the SubmitRow Plugin

In this step, we register the commonly used plugin SubmitRow to demonstrate how we use GA to do it. Type:

```
$  ga +create GUS::Common::Plugin::SubmitRow
```

The output of a successful Submit registration is found in Section Output from Registering SubmitRow. Now, to commit, type:

```
$ ga +create GUS::Common::Plugin::SubmitRow --commit
```

The output of a successful commit is, again, found in Section Output from Registering SubmitRow.

6 GA and GUS Plugins

6.1 GA Usage

`ga [<mode>] [<plugin_class_name>] [<plugin_class_options>] [<ga_options>]`

- <mode> is one of:
  - +meta - create Core.Algorithm and Core.AlgorithmImplementation for self
  - +create - creates Algorithm, AlgorithmImplementation, and AlgorithmParamKey
  - +update - creates AlgorithmImplementation, and AlgorithmParamKey
  - +history - lists invocations
  - +run - runs the plugin (default)
- <plugin_class_name> - from hierarchical namespace, e.g., GUS::Common::Plugin::SubmitRow
- <plugin_class_options> - defined by plugin
- <ga_options> - generic GA options (see Section Plugins Documentation for info on what these options mean)
  - commit
  - debug
  - verbose
  - veryVerbose
  - sqlVerbose
  - user
  - group
  - project
  - comment
  - algoinvo
  - gusconfigfile
  - help
  - helpHTML

6.2 Plugins

A plugin is simply a package that inherits from GUS::PluginMgr::Plugin. It must implement two methods:

- new - to instantiate and initialize the plugin object
- run - to carry out plugin tasks

Note 1: Before running a plugin for the first time, it has to be registered. This is done by running ga in create mode. For an example of how to register a plugin, please see Registering the SubmitRow Plugin.

Note 2: All data inserted into GUS must be done through a plugin. Doing this ensures that all insertions into the database are recorded along with paramaters used to enter the data, etc.

In order to find out what options to use with a plugin, you may get help by issuing a command of the form:

```
ga <plugin_class_name> --help
```

In GUS, version 3.0, there are three Plugin groups:

1. GUS::Common::Plugin
2. GUS::GOPredict::Plugin
3. GUS::RAD::Plugin

6.2.1 The GUS::Common::Plugin Group

- AnnotatorsInterfaceSubmitter
- AssembliesWithflCD
- DeleteSimilarities
- GBParser (see GBParser) example:

```
ga GUS::Common::Plugin::GBParser --file=/usr/local/src/pathport-related/agrobac.gbk \
--db_rel_id=1  --gbRel=138 --debug --commit
Reading properties from /home/apps/GUS/dev/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/GUS/dev/.gus.properties
Mon Nov 17 13:29:06 2003        COMMIT  commit on
Maximum number of  objects set to 50000
Mon Nov 17 13:29:06 2003        RELEASE 138
Mon Nov 17 13:29:06 2003        FILE    /usr/local/src/pathport-related/agrobac.gbk
Mon Nov 17 13:33:14 2003        INSERTED        NC_003305       ; N=1
Mon Nov 17 13:33:15 2003        Genbank entries inserted= 1;  updated= 0;
                                total #(inserted::updated::deleted)=17572::::
Mon Nov 17 13:33:15 2003        RESULT  Genbank entries inserted= 1;  updated= 0;  failed= 0
```

- IdentifyAndLoadSnpsFromAssemblies
- ImportPlasmoDBAAFeatures
- ImportPlasmoDBPrediction
- InsertDbRefAndDbRefNASequence
- InsertNewExtDbRelease
- InsertNewExternalSequences

```
ga GUS::Common::Plugin::InsertNewExternalSequences \
   --external_database_release_id=138 \
   --table_name=DoTS::ExternalNASequence \
   --sequencefile=soybean-one  \
   --debug --sqlVerbose --commit >& soybean-one.out
```

- LoadBLATAlignments
- LoadBlastSimFast
- LoadBlastSimilaritiesPK
- LoadDocumentation
- LoadGeneFeaturesFromXML
- LoadNRDB
- LoadPfam
- LoadTaxon (see LoadTaxon for documentation and Uploading Taxonomy Data for example of how to use it)
- MakeBestSimilarityPair
- MakeIndexWordLink
- MakeIndexWordSimLink
- OrthologGroupsMCL
- SubmitRow (see SubmitRow) example:

```
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType \
--attrlist sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,\
parent_sequence_type_id,name,description \
--valuelist "1^^^DNA^^^null^^^null^^^1^^^1^^^DNA^^^'DNA, unkown strandedness'" \
--commit
Reading properties from /home/apps/GUS/dev/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/GUS/dev/.gus.properties
Tue Nov 11 13:05:14 2003        ALGINVID        38
Tue Nov 11 13:05:14 2003        COMMIT  commit on
Tue Nov 11 13:05:14 2003        Row updated
Tue Nov 11 13:05:14 2003        RESULT  Updated one row
```

- UpdateGusFromCla
- UpdateGusFromXML
- UpdateNASequences
- dbEST

6.2.2 The GUS::GOPredict::Plugin Group

- CopyAASeqGoFuncToProtGoFunc
- DeleteGoPredictions
- FixDeletedProteins
- GoPlugin
- LoadGoAssoc
- LoadGoOntology
- MakeGoPredictionRules
- MakeGoPredictions

6.2.3 The GUS::RAD::Plugin Group

- AnalysisResultLoader
- ArrayLoader
- ArrayResultLoader
- AssayControlLoader
- ControlLoader
- ProcessedResultLoader

7 Uploading Data Into GUS

This section documents the steps needed in order to upload data into GUS. When uploading data from external community databases, essentially, the protocol for data upload consists of:

- Possibly inserting a row into the SRes::ExternalDatabase table (using the SubmitRow plugin). This is needed to be done once per external database.
- Possibly inserting a rwo into the SRes::ExternalDatabaseRelease table (using the SubmitRow plugin). This is done if the data you are about to upload comes from an external database release from which you have not uploaded data before (i.e., there is not a row in the database corresponding to the external database release from which you are about to upload data).
- Uploading the desired data using the appropriate plugin.

7.1 Inserting Initial Data into DoTS::SequenceType

Initial Population of DoTS::SequenceType.

7.2 Inserting Initial Data into SRes::ExternalDatabase and SRes::ExternalDatabaseRelease

Since there are a few databases we know upfront we will be uploading data from, we populate SRes::ExternalDatabase and SRes::ExternalDatabaseRelease with the appropriate information in one batch. The statements used are found in Initial Population of SRes::ExternalDatabase and SRes::ExternalDatabaseRelease.

The following are the databases for which a row was inserted into SRes::ExternalDatabase at this juncture:

- GO Function
- GO Component
- GO Process
- EMBL
- DDBJ
- NBRF PIR
- PRF
- PDB
- Patents
- GenInfo Backbone Id
- General Database Identifier
- NCBI Reference Sequence
- Local Sequence Identifier

- GenBank
- Taxonomy

Therefore, when uploading data from the above databases, there is no longer a need for inserting data into SRes::ExternalDatabase. As for inserting data into SRes::ExternalDatabaseRelease, that depends on whether

7.3 Uploading Taxonomy Data

Download taxonomy file taxdump.tar.gz from NCBI site and save in /usr/local/src/gus-related and unpack by typing:

```
$ cd /usr/local/src/gus-related
$ mkdir taxonomy
$ cd taxonomy
$ tar xvfzp ../taxdump.tar.gz
```

Note: NCBI does not use release numbers (versions) for the taxonomy database.

Now, register the LoadTaxon plugin:

```
$ ga +create GUS::Common::Plugin::LoadTaxon --commit
```

Load file by issuing command:

```
$ ga GUS::Common::Plugin::LoadTaxon --gencode=/usr/local/src/gus-related/taxonomy/gencode.dmp\
                                    --names=/usr/local/src/gus-related/taxonomy/names.dmp\
                                    --nodes=/usr/local/src/gus-related/taxonomy/nodes.dmp\
                                    --commit
```

7.4 Uploading GO Ontology

Download GO ontology data from NCBI site. The files needed are:

- process.ontology
- component.ontology
- function.ontology

Save files in /usr/local/src/gus-related/ontology/.

Register the LoadGoOntology plugin:

```
$ ga +create GUS::GOPredict::Plugin::LoadGoOntology --commit
```

Add data to the SRes::GORelationshipType table using the SubmitRow plugin. See Section Inserting Initial Rows into SRes::GORelationshipType for the commands used.

Finally, upload the ontology data with command:

```
ga GUS::GOPredict::Plugin::LoadGoOntology --file_path=/usr/local/src/gus-related/ontology/
   --id_file=/home/apps/GUS/dev/uploads/go-ids
   --process_db_id=3 --process_ext_db_rel=3
   --function_db_id=1 --function_ext_db_rel=1
   --component_db_id=2 --component_ext_db_rel=2
   --commit
   >& /home/apps/GUS/dev/uploads/go-ontology-upload.out
```

NOTE: IF YOU –COMMIT AT THE FIRST INSTANCE IT WORKS. OTHERWISE IF YOU WANT TO MAKE IT TWICE, CHOOSE ANOTHER NAME FOR THE ID_FILE PARAMETER e.g., IF YOU CHOSE id_go FIRST TIME THEN CHOOSE id_go1.

7.5 Uploading GenBank Data

Download the nucleotide data from NCBI and save the file in GenBank format locally in /usr/local/src/gus-related/genbank/. For our example below, we downloaded data for *Trypanosoma cruzi*.

Prior to uploading GenBank data for the first time, we need to insert information into the SRes::ExternalDatabase in order to avoid violating constraints when trying to upload the GenBank data.

Register the GBParser plugin:

```
$ ga +create GUS::Common::Plugin::GBParser --commit
```

The Genbank data was finally uploaded to the dots tables using the following command:

```
ga GUS::Common::Plugin::GBParser --file=/usr/local/src/gus-related/tcruzi.genbank --debug
   --gbRel=137 --db\_rel\_id=1 >\& /home/apps/oracle/err4 --commit
```

(script available at /home/apps/oracle/scripts/upload_genbank_data.sh)

7.6 Uploading NRDB Data

```
Uploading data using LoadNRDB.pm
================================

(The script LoadNRDB.pm is a modified version downloaded from the CVS site on 8th Dec 2003)

1. The data downloaded from ftp://ftp.ncbi.nih.gov/blast/db/nr.Z
2. The taxonomy data was downloaded from ftp://ftp.ncbi.nih.gov/pub/taxonomy(gi-taxid-prot.dmp)
3. The data were stored at /usr/local/src/gus-related/nrdb/nr.Z and gi_taxid_prot.dmp
4. The NCBI README file containing the list of databases were obtained from
        ftp://ftp.ncbi.nih.gov/blast/db/README.txt

Appendix:
=========
        List of databases:

        Database Name                   Identifier Syntax
  ============================      ========================
  GenBank                           gb|accession|locus
  EMBL Data Library                 emb|accession|locus
  DDBJ, DNA Database of Japan       dbj|accession|locus
```

```
  NBRF PIR                          pir||entry
  Protein Research Foundation       prf||name
  SWISS-PROT                        sp|accession|entry name
  Brookhaven Protein Data Bank      pdb|entry|chain
  Patents                           pat|country|number
  GenInfo Backbone Id               bbs|number
  General database identifier       gnl|database|identifier
  NCBI Reference Sequence           ref|accession|locus
  Local Sequence identifier         lcl|identifier
```

(README.txt is placed at :tyler-lab$/usr/local/src/gus-related/nrdb/README.txt)

1. 178 sequences were fetched from the nr.Z file and were compressed using the command compress mini_nr.
2. Corresponding taxon-ids were fetched from the gi-taxid-prot.dmp.(This file contains the gi number of the protein sequences as the first column and the corresponding taxon id as the second column(tab separated).
3. Both the files were named as mini_nr.Z and mini_nr.dmp respectively and were stored in /usr/local/src/gus-related/nrdb.
4. The database names(from the first column of the readme files) were inserted into the sres.externaldatabase table.(using the submitrow plugin: script is at /home/apps/oracle/scripts/populate.sh)
5. The sres.externaldatabase.external_database_id was inserted into the sres.externaldatabaserelease table as external_database_id attribute and some numbers were assigned to sres.externaldatabaserelease.external_database_release_id.(see script populate.sh).
6. Data was uploaded into the externalaasequence view using the LoadNRDB.pm module. (Script is at /home/apps/oracle/scripts/Load_NRDB.sh).

NOTE: In the --SourceDB option the externaldatabasereleaseid:database identifier is written. (The database identifier is taken from the first value of the identifier syntax from the NCBI readme.txt file)

The --maketemp option does the following things:
1) making the temp table(gusrw.nrdbtemp).
2) inserting into the NRDBEntry and ExternalAASequences tables.
3) deleting obsolete rows in those tables.
ref: sub makeTempTable();

Data to NRDBEntry table:

In the above example there were 178 records uploaded to the aasequenceIMP table and externalaasequences view. However, there were 335 records inserted into the nrdbentry and gusrw.nrdbtemp table.

In nrdbentry table: The primary key nrdb_entry_id auto increments. The aa_sequence_id represents the seriel number of the amino acid uploaded, and hence is repetitive and referes to the aa_sequence_id of the aasequenceImp table.

#Update the ga plugin
#---

```
ga +update GUS::Common::Plugin::LoadNRDB --commit

ga GUS::Common::Plugin::LoadNRDB --algoinvo=1 --comment='Loading NRDB data' --db
i_str='dbi:Oracle:gusdev.bioinformatics.vt.edu' --extDbRelId=137 --temp_login='g
usrw' --temp_password='gusdevrw' --gitax='/usr/local/src/gus-related/nrdb/mini_n
r.dmp' --SourceDB='137:gb,167:sp,170:emb,171:dbj,172:pir,173:prf,174:pdb,175:pat
,176:bbs,177:gnl,178:ref,179:lcl' --nrdb='/usr/local/src/gus-related/nrdb/mini_n
r.Z' --maketemp --plugin --delete --debug >& '/home/apps/oracle/nrdb.err1' --com
mit
```

7.7 Uploading pFAM Data

```
Using Load_pfam module:
=======================
Load_pfam.pm module inserts data into dots.pfamentry, dots.dbrefpfamentry and sres.dbref.
dots.dbrefpfamentry table is the child of dots.pfamentry and sres.dbref. The db_ref_id and
the pfam_entry_id of the dots.dbrefpfamentry table are derived from dots.pfamentry and
sres.dbref tables.

Data for Load_pfam plugin:
==========================

The data can be downloaded from http://www.sanger.ac.uk/Software/Pfam/ftp.shtml.

About pfam:
===========

pfam is a collection of protein family alignments which were constructed semi-automatically using HMMs.
Sequences that are not covered by pfam are clustered and aligned automatically, and released as pfamB.
pfamA families have permanent accession numbers and contain functional annotation and cross-references
to other databases, while pfamB families are re-generated at each release and are un-annotated.

Construction of pfam:
=====================

pfamA is based on a sequence database called pfamseq - pfamseq 11 is based on swissprot 41.25
and SP-TrEMBL 24.14.
pfamB is constructed from PRODOM 2002.1.


Commandline options for Load_Pfam
---------------------------------

ga +create GUS::Common::Plugin::LoadPfam --commit


Prior to uploading the pfam data the following databases were registered in sres::externaldatabase
and sres::externaldatabaserelease. The corresponding release_ids and the dates were obtained from
the respective web_sites which were finally incorporated into the sres::externaldatabaserelease.

Data for sres::externaldatabase:
================================
```

```
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "16^^^PfamA^^^pfama" --ref
resh --verbose --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "17^^^MIM medline^^^mim me
dline" --refresh --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "18^^^Interpro^^^interpro"
 --refresh --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "19^^^Prodom^^^prodom" --r
efresh --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "26^^^Expert^^^expert" --c
ommit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "27^^^PFAMB^^^pfamb" --com
mit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "28^^^PRINTS^^^prints" --c
ommit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "29^^^PROSITE^^^prosite" -
-commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "30^^^PROSITE_PROFILE^^^pr
osite_profile" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "31^^^SCOP^^^scop" --commi
t

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "32^^^SMART^^^smart" --com
mit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "33^^^URL^^^url" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "34^^^LOAD^^^load" --commi
t

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase --attrlist
 external_database_id,name,lowercase_name --valuelist "35^^^HOMSTRAD^^^homstrad"
 --commit
```

```
Data for sres::externaldatabaserelease:
=======================================

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "16^^^16^^^16-DEC-03^^^11.0" --refresh --verbose --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "17^^^17^^^16-DEC-03^^^unknown" --refresh --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "26^^^26^^^27-MAY-03^^^16.0" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "27^^^27^^^16-DEC-03^^^unknown" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "28^^^28^^^11-APR-03^^^36.0" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "29^^^29^^^15-DEC-03^^^18.18" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "30^^^30^^^15-DEC-03^^^18.18" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "31^^^31^^^1-DEC-03^^^1.65" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "32^^^32^^^17-DEC-03^^^unknown" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "33^^^33^^^17-DEC-03^^^unknown" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "34^^^34^^^17-DEC-03^^^unknown" --commit

ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease --a
ttrlist external_database_release_id,external_database_id,release_date,version -
-valuelist "35^^^35^^^17-DEC-03^^^unknown" --commit

Running Load_pfam plugin:
```

```
========================

ga GUS::Common::Plugin::LoadPfam --flat_file=/usr/local/src/gus-related/pfam/sma
ll_pfam.gz --release='11.0' --commit
```

8 Deleting Entries from GUS

```
|-------------------------------------------------------------------------------------
|The deleteEntries.pl is located at $GUS_HOME/bin directory.                          |
|                                                                                     |
|The lines 38 and 39 were modified from $gusconfig->getReadOnlyDatabaseLogin() and
 $gusconfig->getReadOnlyDatabasePassword() to $gusconfig->getDatabaseLogin() and     |
|$gusconfig->getDatabasePassword().                                                   |
|                                                                                     |
|The above functions fetches the user from the .gus.properties file that has add,
     ||modify,delete privileges.(it is gusrw).                                       |
|                                                                                     |
|-------------------------------------------------------------------------------------
```

Syntax:

```
        deleteEntries.pl --idSQL 'select nrdb_entry_id from dots.nrdbentry where
                          nrdb_entry_id=1' --table DoTS::NRDBEntry
```

(The query will delete a row from the nrdbentry table having nrdb_entry_id=1.
If one wants to execute a query like 'delete from dots.nrdbentry' then use option:
--idSQL 'select nrdb_entry_id from dots.nrdbentry' --table dots::nrdbentry)

9 Appendix 1

9.1 Tablespaces Initialization Script

The script is /home/apps/oracle/gusdev/scripts/create_tblsp.sql:

```
-- 27th Oct 2003:
--This script is used for creating tablespaces

--11th Dec 2003:
--This script is modified because the data in the dots namespace used up
--the 200M very fast.  The modification (autoextend on next 10M maxsize 5000M)
--will automatically allocate more space if the quota of 200M gets full.

create tablespace gus_core
datafile '/tyler-lab/oradata/gusdev/dynmgus_core.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_sres
datafile '/tyler-lab/oradata/gusdev/dynmgus_sres.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_dots
datafile '/tyler-lab/oradata/gusdev/dynmgus_dots.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_tess
datafile '/tyler-lab/oradata/gusdev/dynmgus_tess.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_rad3
datafile '/tyler-lab/oradata/gusdev/dynmgus_rad3.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_core_idx
datafile '/tyler-lab/oradata/gusdev/dynmgus_core_idx.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_sres_idx
datafile '/tyler-lab/oradata/gusdev/dynmgus_sres_idx.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_dots_idx
datafile '/tyler-lab/oradata/gusdev/dynmgus_dots_idx.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_tess_idx
```

```
datafile '/tyler-lab/oradata/gusdev/dynmgus_tess_idx.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);

create tablespace gus_rad3_idx
datafile '/tyler-lab/oradata/gusdev/dynmgus_rad3_idx.dbf' size 200M
autoextend on next 10M maxsize 5000M
default storage (initial 4M next 1M pctincrease 0);
```

9.2 Creating Required GUS Users Script

The script is /home/apps/oracle/gusdev/scripts/create_users.sql:

```
--This script is to creat the users.

DROP USER GUSRW CASCADE;

CREATE USER GUSRW IDENTIFIED BY <password>
  TEMPORARY TABLESPACE TEMP
  DEFAULT TABLESPACE users
  QUOTA UNLIMITED ON TEMP
  QUOTA UNLIMITED ON users;

DROP USER GUSdevReadOnly CASCADE;

CREATE USER GUSdevReadOnly IDENTIFIED BY <password>
  TEMPORARY TABLESPACE TEMP
  DEFAULT TABLESPACE users
  QUOTA UNLIMITED ON TEMP
  QUOTA UNLIMITED ON users;
```

9.3 Granting Privileges to GUS Users Script

The script is /home/apps/oracle/gusdev/scripts/grant_users.sql:

```
--This is to grant permission to the users
GRANT CREATE SESSION  TO GUSrw;
GRANT CREATE TABLE    TO GUSrw;
GRANT CREATE VIEW     TO GUSrw;
GRANT CREATE SEQUENCE TO GUSrw;
GRANT CREATE TRIGGER  TO GUSrw;
GRANT CREATE SYNONYM  TO GUSrw;
GRANT SELECT ANY TABLE    TO GUSrw;
GRANT INSERT ANY TABLE    TO GUSrw;
GRANT UPDATE ANY TABLE    TO GUSrw;
GRANT DELETE ANY TABLE    TO GUSrw;
GRANT CREATE SESSION    TO GUSdevReadOnly;
GRANT SELECT ANY TABLE  TO GUSdevReadOnly;
GRANT SELECT ANY TABLE  TO GUSdevReadOnly;
```

9.4 Environment Variables Settings for Building GUS

The environment variables are found in the *.profile* file for the GUS installation user, gus (whose home directory is /home/gususer) in VBI's case. Here are the contents of that file:

```
export ANT_HOME=/usr/local/ant/apache-ant-1.5.4
export JAVA_HOME=/usr/lib/j2sdk1.4.2_01
export PROJECT_HOME=/home/apps/GUS/dev/src
export GUS_HOME=/home/apps/gus/GUS/dev/gushome
export GUS_CONFIG_FILE=/home/apps/GUS/dev/.gus.properties
export ORACLE_HOME=/home/oracle/products/9.2.0
export PERL5LIB=$GUS_HOME/lib/perl
PATH=$GUS_HOME/bin:$PROJECT_HOME/install/bin:$ANT_HOME/bin:/home/oracle/products/9.2.0/bin:${PATH}
export PATH
```

9.5 GUS Properties File

The GUS properties file is /home/apps/gus/.gus.properties:

```
databaseLogin=GUSrw
databasePassword=<password>
readOnlyDatabaseLogin=GUSdevReadOnly
readOnlyDatabasePassword=<password>
coreSchemaName=Core
userName=gus
group=dba
project=gus
dbiDsn=dbi:Oracle:gusdev.bioinformatics.vt.edu
```

9.6 Install Property File

The GUS install properties file is /home/apps/GUS/dev/gushome/config/install.prop:

```
perl=/usr/bin/perl
```

9.7 Schema Property File

The file is /home/apps/GUS/dev/gushome/config/schema.prop:

```
# Password for the Oracle 'SYS' user; this is used only to
# create the new Oracle users that will own the GUS tables.
#
oracle_systemPassword=<password>

# SID for the Oracle server on which to install GUS.
#
oracle_SID=gusdev

# Hostname for the Oracle server on which to install GUS
#
oracle_host=tuor

# Default tablespace and quotas for the new users.
#
oracle_tempTablespace=TEMP
oracle_tempQuota=UNLIMITED
oracle_defaultQuota=UNLIMITED

# The names, passwords, and tablespaces for the new users.
# It is recommended that you do NOT change the default usernames
```

```
# as this feature has not been fully tested (the username is the
# first parameter in each group of four.)
#
oracle_core=Core
oracle_corePassword=<password>
oracle_coreTablespace=gus_core
oracle_coreIndexTablespace=gus_core_idx

oracle_corever=CoreVer
oracle_coreverPassword=<password>
oracle_coreverTablespace=gus_core
oracle_coreverIndexTablespace=gus_core_idx

oracle_sres=SRes
oracle_sresPassword=<password>
oracle_sresTablespace=gus_sres
oracle_sresIndexTablespace=gus_sres_idx

oracle_sresver=SResVer
oracle_sresverPassword=<password>
oracle_sresverTablespace=gus_sres
oracle_sresverIndexTablespace=gus_sres_idx

oracle_dots=DoTS
oracle_dotsPassword=<password>
oracle_dotsTablespace=gus_dots
oracle_dotsIndexTablespace=gus_dots_idx

oracle_dotsver=DoTSVer
oracle_dotsverPassword=<password>
oracle_dotsverTablespace=gus_dots
oracle_dotsverIndexTablespace=gus_dots_idx

oracle_tess=TESS
oracle_tessPassword=<password>
oracle_tessTablespace=gus_tess
oracle_tessIndexTablespace=gus_tess_idx

oracle_tessver=TESSVer
oracle_tessverPassword=<password>
oracle_tessverTablespace=gus_tess
oracle_tessverIndexTablespace=gus_tess_idx

oracle_rad3=RAD3
oracle_rad3Password=<password>
oracle_rad3Tablespace=gus_rad3
oracle_rad3IndexTablespace=gus_rad3_idx

oracle_rad3ver=RAD3Ver
oracle_rad3verPassword=<password>
oracle_rad3verTablespace=gus_rad3
oracle_rad3verIndexTablespace=gus_rad3_idx
```

9.8 Populating Machine Table

```
insert into core.machine values(0, 'unknown', NULL, 1, 0, SYSDATE, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1);
```

9.9 Output from Registering GA

```
<Core::Algorithm>
  <name>GA-Plugin</name>
  <description>GUS application framework for plugins</description>
  <Core::AlgorithmImplementation>
    <cvs_revision>1.47</cvs_revision>
    <cvs_tag> </cvs_tag>
    <executable>GUS::PluginMgr::GusApplication</executable>
    <executable_md5>912ad90decf966dc67d4c217adca0020</executable_md5>
    <description>update for GUS 3.0</description>
    <Core::AlgorithmInvocation>
      <start_time>SYSDATE</start_time>
      <end_time>SYSDATE</end_time>
      <machine_id>0</machine_id>
      <cpus_used>1</cpus_used>
      <result>meta</result>
    </Core::AlgorithmInvocation>
  </Core::AlgorithmImplementation>
</Core::Algorithm>
<Core::Algorithm>
  <algorithm_id>2</algorithm_id>
  <name>GA-Plugin</name>
  <description>GUS application framework for plugins</description>
  <modification_date>SYSDATE</modification_date>
  <Core::AlgorithmImplementation>
    <algorithm_implementation_id>2</algorithm_implementation_id>
    <algorithm_id>2</algorithm_id>
    <cvs_revision>1.47</cvs_revision>
    <cvs_tag> </cvs_tag>
    <executable>GUS::PluginMgr::GusApplication</executable>
    <executable_md5>912ad90decf966dc67d4c217adca0020</executable_md5>
    <description>update for GUS 3.0</description>
    <modification_date>SYSDATE</modification_date>
    <Core::AlgorithmInvocation>
      <algorithm_invocation_id>2</algorithm_invocation_id>
      <algorithm_implementation_id>2</algorithm_implementation_id>
      <start_time>SYSDATE</start_time>
      <end_time>SYSDATE</end_time>
      <machine_id>0</machine_id>
      <cpus_used>1</cpus_used>
      <result>meta</result>
      <modification_date>SYSDATE</modification_date>
      <user_read>1</user_read>
      <user_write>1</user_write>
      <group_read>1</group_read>
      <group_write>1</group_write>
      <other_read>1</other_read>
      <other_write>0</other_write>
      <row_user_id>1</row_user_id>
```

```
        <row_group_id>1</row_group_id>
        <row_project_id>1</row_project_id>
        <row_alg_invocation_id>1</row_alg_invocation_id>
      </Core::AlgorithmInvocation>
      <user_read>1</user_read>
      <user_write>1</user_write>
      <group_read>1</group_read>
      <group_write>1</group_write>
      <other_read>1</other_read>
      <other_write>0</other_write>
      <row_user_id>1</row_user_id>
      <row_group_id>1</row_group_id>
    <row_project_id>1</row_project_id>
    <row_alg_invocation_id>1</row_alg_invocation_id>
  </Core::AlgorithmImplementation>
  <user_read>1</user_read>
  <user_write>1</user_write>
  <group_read>1</group_read>
  <group_write>1</group_write>
  <other_read>1</other_read>
  <other_write>0</other_write>
  <row_user_id>1</row_user_id>
  <row_group_id>1</row_group_id>
  <row_project_id>1</row_project_id>
  <row_alg_invocation_id>1</row_alg_invocation_id>
</Core::Algorithm>
```

9.10 Output from Registering SubmitRow

```
$ ga +create GUS::Common::Plugin::SubmitRow
Reading properties from /home/apps/GUS/dev/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/GUS/dev/.gus.properties
Thu Oct 30 15:20:18 2003        INFO    Plugin GUS::Common::Plugin::SubmitRow re
gistered with cvs revision '1.4' and cvs tag ' '
Thu Oct 30 15:20:18 2003        INFO    ...Just kidding: you didn't --commit

$ ga +create GUS::Common::Plugin::SubmitRow --commit
Reading properties from /home/apps/GUS/dev/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/GUS/dev/.gus.properties
Thu Oct 30 15:30:48 2003        INFO    Plugin GUS::Common::Plugin::SubmitRow re
gistered with cvs revision '1.4' and cvs tag ' '
```

9.11 Initial Population of DoTS::SequenceType

The statements used are in /home/apps/oracle/gusdev/scripts/populate.sh

```
#In this script the table names and the tablespace names were kept as the same case as that of
#the gusdb.org documentation on schemas.

#This part of the script is to populate data into the DoTS.SequenceType table.
#These values were created on 15th December and care was taken to have correct
#strand value, hierarchy,parent_sequence_type_id and comments

ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
```

```
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "1^^^DNA^^^null^^^null^^^1^^^1^^^DNA^^^unkown strandedness" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "2^^^DNA^^^ds^^^1^^^2^^^1^^^ds-DNA^^^first strand" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "3^^^DNA^^^ds^^^2^^^2^^^1^^^ds-DNA^^^second strand" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "4^^^DNA^^^ss^^^null^^^3^^^1^^^ss-DNA^^^single stranded DNA" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "5^^^RNA^^^null^^^null^^^4^^^1^^^RNA^^^unknown strandedness" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "6^^^RNA^^^ds^^^1^^^5^^^5^^^ds-RNA^^^first strand" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "7^^^RNA^^^ds^^^2^^^5^^^5^^^ds-RNA^^^second strand" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "8^^^RNA^^^mRNA^^^null^^^6^^^5^^^mRNA^^^mRNA" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "9^^^RNA^^^tRNA^^^null^^^7^^^5^^^tRNA^^^tRNA" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "10^^^RNA^^^rRNA^^^null^^^8^^^5^^^rRNA^^^rRNA" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "11^^^unknown^^^null^^^null^^^9^^^11^^^unknown^^^unkown" --commit
```

```
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "12^^^RNA^^^predicted_mRNA^^^null^^^10^^^8^^^predicted mRNA^^^mRNA sequence predicted by an
    algorithm from genomic DNA" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "13^^^virtual^^^null^^^11^^^13^^^null^^^virtual^^^virtual nucleic acid sequence" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "14^^^DNA^^^GSS^^^ds^^^12^^^1^^^GSS^^^Genome Survey Sequence" --commit
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
   --attrlist
   sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,name,description
   --valuelist
   "15^^^DNA^^^oligonucleotide^^^ss^^^13^^^1^^^oligonucleotide^^^synthetic single stranded
    oligonucleotide" --commit
```

9.12 Initial Population of SRes::ExternalDatabase and SRes::ExternalDatabaseRelease

```
#In this script the table names and the tablespace names were kept as the same case as that of
#the gusdb.org documentation on schemas.

#INSERT DATA INTO THE Sres::externaldatabase and Sres::Externaldatabaserelease
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "1^^^GO Function^^^go function" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "2^^^GO Component^^^go component" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "3^^^GO Process^^^go process" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "4^^^EMBL^^^embl" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "5^^^DDBJ^^^ddbj" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "6^^^NBRF PIR^^^nbrf pir" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "7^^^PRF^^^prf" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "8^^^PDB^^^pdb" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
```

```
   --attrlist external_database_id,name,lowercase_name
   --valuelist "9^^^Patents^^^patents" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "10^^^GenInfo Backbone Id^^^geninfo backbone id" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "11^^^General Database Identifier^^^general database identifier" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "12^^^NCBI Reference Sequence^^^ncbi reference sequence" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "13^^^Local Sequence Identifier^^^local sequence identifier" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "14^^^GenBank^^^genbank" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabase
   --attrlist external_database_id,name,lowercase_name
   --valuelist "15^^^Taxonomy^^^taxonomy" --commit
# Inserting corresponding data into externaldatabaserelease
# ========================================================
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "1^^^1^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "2^^^2^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "3^^^3^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "4^^^4^^^26-NOV-03^^^77" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "5^^^5^^^01-SEP-03^^^55" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "6^^^6^^^24-NOV-03^^^78.3" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "7^^^7^^^01-NOV-03^^^94" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "8^^^8^^^09-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "9^^^9^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "10^^^10^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
```

```
   --valuelist "11^^^11^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "12^^^12^^^04-NOV-03^^^7" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "13^^^13^^^15-DEC-03^^^unknown" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "14^^^14^^^15-OCT-03^^^138" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::ExternalDatabaseRelease
   --attrlist external_database_release_id,external_database_id,release_date,version
   --valuelist "15^^^15^^^15-DEC-03^^^unknown" --commit
```

9.13 Inserting Initial Rows into SRes::GORelationshipType

The statements used are in /home/apps/oracle/gusdev/scripts/populate.sh

```
#ADDING DATA TO GORelationshiptype table
ga GUS::Common::Plugin::SubmitRow --tablename SRes::GORelationshipType
   --attrlist go_relationship_type_id,name --valuelist "1^^^isa" --commit
ga GUS::Common::Plugin::SubmitRow --tablename SRes::GORelationshipType
   --attrlist go_relationship_type_id,name --valuelist "2^^^partof" --commit
```

9.14 VBI's GUS Directory Structure

```
/home/apps/GUS/
               dev
               pd
           oracle/
                  scripts
```

10 Appendix 2

10.1 Plugins Documentation

All Plugins will print out a long help message when they are invoked using the form:

```
$ ga <plugin_class_name> --help
```

The help text emitted contains Plugin specific options and GA options. The ga options common to all Plugins are:

```
--commit
    Actualy commit changes to the database
    default: 0

--debug
    output extra debugging information to verify correct operation
    default: 0

--sqlVerbose
    Use this flag to switch on a log of SQL statements executed by Perl
    object layer.
    default: 0

--verbose
    Use this flag to enable output of logVerbose messages from the
    plugin.
    default: 0

--veryVerbose
    Use this flag to enable output of logVeryVerbose and logVerbose
    messages from the plugin.
    default: 0

--user *string*
    Set the user name in new or changed rows with this GUS user name
    (from Core.UserInfo table) [default is value in gus config file]

--group *string*
    Set the group name in new or changed rows with this GUS group name
    (from Core.GroupInfo table) [default is value in gus config file]

--project *string*
    set the project name in new or changed rows with this GUS project
    name (from Core.Project table) [default is value in gus config file]

--comment *string*
    Set Core.AlgorithmInvocation.comment with this comment

--algoinvo *integer*
```

```
        Use this algorithm invocation id in the event that a new algorithm
        invocation id cannot be generated automatically
        default: 1

    --gusconfigfile *file*
        The gus config file to use [note: the default is your
        $GUS_CONFIG_FILE]
        default: /home/apps/gus/.gus.properties
        format: GUS config file format
        mustExist: 1

    --help
        Request long help

    --helpHTML
        Request help in HTML
```

10.1.1 SubmitRow

```
NAME
    GUS::Common::Plugin::SubmitRow - Update a row with values supplied as
    command line arguments.

SYNOPSIS
    ga GUS::Common::Plugin::SubmitRow --help

    ga GUS::Common::Plugin::SubmitRow --helpHTML

    ga GUS::Common::Plugin::SubmitRow --tablename tablename --attrlist
    string-list --valuelist string [--refresh ] [--commit ] [--debug ]
    [--sqlVerbose ] [--verbose ] [--veryVerbose ] [--user string] [--group
    string] [--project string] [--comment string] [--algoinvo integer]
    [--gusconfigfile file]

DESCRIPTION
    This plugin either inserts or updates a row in the database, depending
    upon whether you provide a primary key as one of the attributes. If
    you do, that row will be read, updated with the attribute values you
    supply, and written. Otherwise, a new row will be inserted.

TABLES AFFECTED
    Core::AlgorithmInvocation
        The plugin manager (ga) inserts a row into this table describing the
        plugin and the parameter values used

TABLES DEPENDED ON
    Core::Algorithm
        The algorithm (ie, this plugin) responsible for the update

    Core::AlgorithmImplementation
        The specific implementation of it
```

```
    Core::AlgorithmParamKey
        The keys for the plugin's command line parameters

    Core::AlgorithmParamKeyType
        The data types of the parameters

RESTARTING
    This plugin has no restart facility.

ARGUMENTS IN DETAIL
    --tablename *tablename* (Required)
        Name of table or view to submit to

    --attrlist *string-list* (comma delimited) (Required)
        List of attributes to update

    --valuelist *string* (Required)
        List of values to update (^^^ delimited)

    --refresh
        True to update the row with new modification data and
        algorithmInvocation regardless of whether it has changed from the
        database

        default: 0

NOTES
    This plugin replaces "GUS::Common::Plugin::UpdateGusFromCla"
```

10.1.2 LoadTaxon

```
Plug_in to populate the Taxon, GeneticCode, and TaxonName tables using files downloaded from NCBI
Usage:
  --gencode=s
        hint:     file from ncbi with genetic_code_id,abbreviation,ame,code,starts
                  e.g./usr/local/db/local/taxonomy/gencode.dmp
        type:     string

  --names=s
        hint:     file from ncbi with tax_id,name,name_class,etc
                  e.g. /usr/local/db/local/taxonomy/names.dmp
        type:     string

  --nodes=s
        hint:     file from ncbi with tax_id,parent_id,rank,
                  etc e.g. /usr/local/db/local/taxonomy/nodes.dmp
        type:     string

  --restart=i
        hint:     tax_id of parent for restart, 0 if no re
        type:     int
```

10.1.3 LoadGoOntology

```
Loads GO Terms into GUS and creates relationships among them
Usage:
  --component_db_id=i
        hint:     External database Id in GUS of the cellular component GO branch
        type:     int

  --component_ext_db_rel=i
        hint:     External database release id in GUS of the cellular component GO branch
        type:     int

  --create_release!
        hint:     Set this to automatically create an external database release id for this GO Term vers
        type:     boolean

  --file_path=s
        hint:     location of .ontology files to read
        type:     string
        REQUIRED!

  --flat_file=s
        hint:     read data from this flat_file.  If blank, read data from all .ontology files in filepa
        type:     string

  --function_db_id=i
        hint:     External database Id in GUS of the molecular function GO branch
        type:     int

  --function_ext_db_rel=i
        hint:     External database release id in GUS of the molecular function GO branch
        type:     int

  --id_file=s
        hint:     read and append successfully processed ID here (necessary for crash-recovery)
        type:     string
        REQUIRED!

  --loadAgain!
        hint:     set this to reload a version of the GO Ontology that has been previously loaded
        type:     boolean

  --process_db_id=i
        hint:     External database Id in GUS of the biological process GO branch
        type:     int

  --process_ext_db_rel=i
        hint:     External database release id in GUS of the biological process GO branch
        type:     int
```

10.1.4 GBParser

```
Module used to parse GenBank records into the DB
Usage:
```

```
  --db_rel_id=i   (Required)
        hint:     GUS ExternalDatabaseRelease id for the GenBank release being loaded.
        type:     int

  --div=s
        hint:     OPTIONAL: If given, entries will be filtered on DIV. (Ex: div="PRI")
        type:     string

  --failTolerance=i
        hint:     number of entries that can fail before aborting
        type:     int
        default:  100

  --file=s        (Required)
        hint:     GenBank formatted file to read
        type:     string

  --gbRel=s       (Required)
        hint:     GenBank release
        type:     string

  --start=i
        hint:     Entry number in specified file to start at.
        type:     int

  --testnumber=i
        hint:     number of iterations for testing
        type:     int

  --updateAll!
        hint:     Update ALL entries, regardless of release date
        type:     boolean
```

10.1.5 InsertNewExternalSequences

```
Insert new ExternalSequences (NA or AA) from a FASTA file
  --external_database_releas (Required)
        hint:     ExternalDatabase release id for these sequences
        type:     int

  --log_frequency=i
        hint:     frequency to print log
        type:     int
        default:  10

  --no_check!
        hint:     if true, does NOT check to see if external_database_release_id,
                  source_id is already in db...
        type:     boolean

  --no_sequence!
        hint:     if on command line, will not set sequence
        type:     boolean
```

```
--regex_chromosome=s
      hint:     regular expression to pick the chromosome from the defline
      type:     string

--regex_contained_seqs=s
      hint:     regular expression to pick the number of contained sequences from the defline
      type:     string

--regex_desc=s
      hint:     regular expression to pick the description of the sequence from the defline
      type:     string

--regex_mol_wgt=s
      hint:     regular expression to pick the molecular weight of the sequence from the defline
      type:     string

--regex_name=s
      hint:     regular expression to pick the name of the sequence from the defline
      type:     string

--regex_secondary_id=s
      hint:     regular expression to pick the secondary id of the sequence from the defline
      type:     string

--regex_source_id=s
      hint:     regular expression to pick the source_id of the sequence from the defline
      type:     string

--sequence_type_id=i
      hint:     sequence type id for these sequences
      type:     int

--sequencefile=s (Required)
      hint:     name of file containing the sequences
      type:     string

--startAt=i
      hint:     ignores entries in fasta file prior to this number...
      type:     int

--table_name=s (Required)
      hint:     Table name to insert sequences into, in schema::table format
      type:     string
      patterns: DoTS::ExternalNASequence, DoTS::VirtualSequence, DoTS::ExternalAASequence,
                DoTS::MotifAASequence

--taxon_id=i
      hint:     taxon id for these sequences
      type:     int

--testnumber=i
      hint:     number of iterations for testing
      type:     int
```

```
  --update!
        hint:     if true, checks to see if row is updated...
        type:     boolean

  --update_longest!
        hint:     if true, checks to see if sequence is longer than that currently loaded,
                  if so, updates the entry...
        type:     boolean

  --writeFile=s
        hint:     writes a sequence file of the new entries
        type:     string

you must provide --external_database_release_id, --table_name and valid fasta sequencefile
on the command line
```

11 Troubleshooting

11.1 Running the SubmitRow Plugin

```
ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType \
--attrlist sequence_type_id,nucleotide_type,sub_type,strand,hierarchy,parent_sequence_type_id,\
name,description,modification_date,user_read,user_write,group_read,group_write,other_read,\
other_write,row_user_id,row_group_id,row_project_id,row_alg_invocation_id \
--valuelist 2000^^^'TNA'^^^null^^^null^^^1^^^1^^^'TNA'^^^'TNA, unkown standedness'^^^SYSDATE\
^^^1^^^1^^^1^^^1^^^1^^^0^^^6^^^3^^^2^^^1
Reading properties from /home/apps/gus/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/gus/.gus.properties
Wed Nov  5 11:19:33 2003        ALGINVID        4
Wed Nov  5 11:19:33 2003        COMMIT  commit off
Can't locate object method "new" via package "GUS::Model::Dots::SequenceType" at
/home/apps/gus/gushome/lib/perl/GUS/Common/Plugin/SubmitRow.pm line 104.
```

```
$ ga GUS::Common::Plugin::SubmitRow --tablename DoTS::SequenceType
     --attrlist SEQUENCE_TYPE_ID,NUCLEOTIDE_TYPE,SUB_TYPE,STRAND,HIERARCHY,\
PARENT_SEQUENCE_TYPE_ID,NAME,DESCRIPTION,MODIFICATION_DATE,USER_READ,USER_WRITE,\
GROUP_READ,GROUP_WRITE,OTHER_READ,OTHER_WRITE,ROW_USER_ID,ROW_GROUP_ID,ROW_PROJECT_ID,\
ROW_ALG_INVOCATION_ID
     --valuelist 2000^^^'TNA'^^^null^^^null^^^1^^^1^^^'TNA'^^^'TNA,\
unkown standedness'^^^SYSDATE^^^1^^^1^^^1^^^1^^^1^^^0^^^6^^^3^^^2^^^1
Reading properties from /home/apps/gus/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/gus/.gus.properties
Wed Nov  5 12:29:00 2003        ALGINVID        4
Wed Nov  5 12:29:00 2003        COMMIT  commit off
DBD::Oracle::st execute failed: ORA-00957: duplicate column name (DBD ERROR: OCIStmtExecute)
[for Statement "INSERT INTO DoTS.SequenceType (  ROW_USER_ID, USER_READ, group_read, user_read,
sequence_type_id, other_write, modification_date, GROUP_WRITE, OTHER_READ, USER_WRITE,
MODIFICATION_DATE, row_group_id, ROW_PROJECT_ID, PARENT_SEQUENCE_TYPE_ID, user_write, NAME,
group_write, other_read, HIERARCHY, ROW_GROUP_ID, OTHER_WRITE, row_user_id, NUCLEOTIDE_TYPE,
row_project_id, row_alg_invocation_id, DESCRIPTION, GROUP_READ, SEQUENCE_TYPE_ID,
ROW_ALG_INVOCATION_ID )
VALUES   (  ?, ?, ?, ?, ?, ?, SYSDATE, ?, ?, ?, SYSDATE, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ? ) " with ParamValues: :p5=1, :p23='4', :p20='1', :p12='1', :p8='1', :p14='TNA',
:p24='TNA, unkown standedness', :p26='2000', :p15=1, :p19='0', :p4=1, :p21='TNA', :p18='3',
:p10='1', :p27='1', :p13=1, :p2='1', :p16=1, :p6=0, :p3=1, :p25='1', :p1='6', :p7='1', :p17='1',
:p22='1', :p9='1', :p11='2'] at /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiDbHandle.pm
line 144.
```

```
 SQL ERROR!! involving
    INSERT INTO DoTS.SequenceType (  ROW_USER_ID, USER_READ, group_read, user_read,
    sequence_type_id, other_write, modification_date, GROUP_WRITE, OTHER_READ, USER_WRITE,
    MODIFICATION_DATE, row_group_id, ROW_PROJECT_ID, PARENT_SEQUENCE_TYPE_ID, user_write,
    NAME, group_write, other_read, HIERARCHY, ROW_GROUP_ID, OTHER_WRITE, row_user_id,
    NUCLEOTIDE_TYPE, row_project_id, row_alg_invocation_id, DESCRIPTION, GROUP_READ,
    SEQUENCE_TYPE_ID, ROW_ALG_INVOCATION_ID )
    VALUES   (  ?, ?, ?, ?, ?, ?, SYSDATE, ?, ?, ?, SYSDATE, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
    ?, ?, ?, ?, ?, ?, ?, ? )
 Values: 6, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 1, 1, TNA, 1, 1, 1, 3, 0, 1, TNA, 1, 4, TNA,
         unkown standedness, 1, 2000, 1 at
         /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiDbHandle.pm line 184
        GUS::ObjRelP::DbiDbHandle::death('GUS::ObjRelP::DbiDbHandle=HASH(0x85dc65c)',
```

```
        '\x{a} SQL ERROR!! involving\x{a} \x{a}    INSERT INTO DoTS.SequenceType (...')
        called at /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiDbHandle.pm line 147
        GUS::ObjRelP::DbiDbHandle::sqlExec('GUS::ObjRelP::DbiDbHandle=HASH(0x85dc65c)',
        'GUS::ObjRelP::DbiDbHandle::st=HASH(0x888faec)','ARRAY(0x888fabc)','\x{a}
        INSERT INTO DoTS.SequenceType (  ROW_USER_ID, USER_READ,...') called at
        /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiRow.pm line 674
        GUS::ObjRelP::DbiRow::quote_and_insert('GUS::Model::DoTS::SequenceType=HASH(0x886e450)'
        ,'HASH(0x886e4b0)') called at /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiRow.pm
        line 621 GUS::ObjRelP::DbiRow::insert('GUS::Model::DoTS::SequenceType=HASH(0x886e450)')
        called at /home/apps/gus/gushome/lib/perl/GUS/Model/GusRow.pm line 1677
        GUS::Model::GusRow::submit('GUS::Model::DoTS::SequenceType=HASH(0x886e450)')
        called at /home/apps/gus/gushome/lib/perl/GUS/Common/Plugin/SubmitRow.pm line 108
        GUS::Common::Plugin::SubmitRow::run('GUS::Common::Plugin::SubmitRow=HASH(0x8597a28)',
       'HASH(0x8806144)') called at
        /home/apps/gus/gushome/lib/perl/GUS/PluginMgr/GusApplication.pm line 430
        eval {...} called at /home/apps/gus/gushome/lib/perl/GUS/PluginMgr/GusApplication.pm
         line 427
        GUS::PluginMgr::GusApplication::doMajorMode_Run('GUS::PluginMgr::GusApplication=
        HASH(0x813361c)','GUS::Common::Plugin::SubmitRow') called at
        /home/apps/gus/gushome/lib/perl/GUS/PluginMgr/GusApplication.pm line 283
        GUS::PluginMgr::GusApplication::doMajorMode('GUS::PluginMgr::GusApplication=
        HASH(0x813361c)','GUS::Common::Plugin::SubmitRow') called at
        /home/apps/gus/gushome/lib/perl/GUS/PluginMgr/GusApplication.pm line 192
        GUS::PluginMgr::GusApplication::parseAndRun('GUS::PluginMgr::GusApplication=
        HASH(0x813361c)','ARRAY(0x813aee8)') called at /home/apps/gus/gushome/bin/ga line 11

gus@tyler-lab:/home/apps/gus/gushome/config$ ga +meta --commit
Reading properties from /home/apps/gus/gushome/config/GUS-PluginMgr.prop
Reading properties from /home/apps/gus/.gus.properties
DBI subclasses 'GUS::ObjRelP::DbiDbHandle::db' and ::st are not setup,
RootClass
 ignored at /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiDatabase.pm
line 152

DBI subclasses 'GUS::ObjRelP::DbiDbHandle::db' and ::st are not setup,
RootClass
 ignored at /home/apps/gus/gushome/lib/perl/GUS/ObjRelP/DbiDatabase.pm
line 152

bash-2.05b$ pwd
/home/apps/gus_source/GUS
bash-2.05b$ cd Model/
bash-2.05b$ ls
bin  config  data  lib  schema  src
bash-2.05b$ cd lib/perl/
bash-2.05b$ ls
App  Core  DoTS  GusRow.pm  RAD3  SRes  TESS  generated
bash-2.05b$ more generated
Tue Oct 28 15:59:53 EST 2003

bash-2.05b$ pwd
```

```
/home/apps/gus_source/GUS/Model/src/java/org/gusdb/model
bash-2.05b$ more generated
Tue Oct 28 16:03:04 EST 2003
```