LOVD v. 2.0 structure and planned changes

Draft proposal 1.0; modified 2005-11-18.
Draft proposal 1.1; modified 2006-06-29.

Rewrite

LOVD requires a rewrite, to make the code more flexible and readable by other contributors. Some code dates as far back as 2002 and the quality, code structure and coding style does not meet up with the newer code from 2005. A rewrite also allows us to easily fit in the some hundred feature requests and bugfixes that we have listed.

Flexibility

The new structure of the LOVD gene databases will be dynamic, so that managers can select the columns that will be in use by the gene databases. Every column represents a XML entity to allow future importing and exporting of XML files into and from LOVD very easy.
Currently, the Custom Column functionality is the only way for users to modify the fixed list of LOVD columns, but they can't remove the LOVD columns they don't use. The plan is to expand the standard set of LOVD columns and allow others to edit or remove almost all of them. Also, if the users send us the columns they have added, we can incorporate these into the LOVD release as non-standard columns that users can activate for their genes.

Because the columns will represent a XML entity, we must either create our own LOVD XML standard which will expand over time as new entities (columns) are added, or adapt an existing (such as the OMG PML definition) and map our planned LOVD columns to this definition. Currently available is an XML export functionality developed by a VPAC (Victorian Partnership for Advanced Computing, www.vpac.org) student in Australia. To install the export utility, some code modification of the original LOVD v.1.1.0 release is necessary. Additional to this, the module is not compatible with the new LOVD structure and requires manual editing of a configuration file after custom columns have been added or removed.

Readability

The new LOVD will be written using a set coding style, documented within LOVD. This should make it much easier for any contributor to understand how LOVD works and how to add or edit code. Also, the table structure (current version attached to this document) should be available within the released archive files, so that contributors can have a good overview of the LOVD framework structure.

New features

The total list of new features incorporated is too long to mention, but the most important are:

- Store sequence variant data separate from patient data, to reduce data redundancy, increase flexibility and ease of use. This solves the problem having to create two entries for patients with compound heterozygous mutations.
- Include support for InnoDB transactional table types.
- Fully dynamic gene database tables (resulting in dynamic legends and dynamic submission form).
- Submitters can be edited and deleted by the managers.
- Use of Custom Links clarified.
- Search function of the public area will match the power of the search functionality in the curator area.
- Curator and manager accounts can be restricted access based on IP addresses (feature can be turned off).
- After a set amount of failed login attempts, accounts of submitters, curators and managers can be locked and have to be unlocked by an authorized user (feature can be turned off).
- "Forgot my password" option for submitters and curators/managers (will auto generate an alternative password for the user which is mailed to the known email address).
- Introduction of a LOVD event and error log.
- Submitters can track their own submissions and will be able to see and edit their full details (feature can be turned off).
- Some settings which previously were system wide will be available per gene (such as phenotype lists) and in some cases can also be changed by the curator.
- Some settings which previously were system wide will be available per user.
- Include support for modules (such as XML import/export and Mutalyzer nomenclature check).

Modules
Suggested is a module-based system allowing the extension of LOVD through a simplified manner. Unpacking module files into the module directory should be the only non web-based action. LOVD will detect the newly installed module and will allow it to be activated, after which the module is functional. The modules depend on LOVD libraries and will include own code for pages, forms and such, whereas for integration in existing LOVD pages or forms, LOVD needs to incorporate a few lines of code that will include short pieces of the modules' code where necessary.
This will imply that certain rules are followed considering file naming and directory structure, and that modules can not just be installed with full functionality on all LOVD installs. The LOVD install needs to support the module. This will mean modules have to be sent to the LOVD developer, who will incorporate code in the correct spot to incorporate the module's code. The module can then also be downloaded from the LOVD website. Whether or not the LOVD download will include any modules or that the modules are available as a separate download, is not yet clear.

Dropped features
One feature that is currently planned to be dropped from LOVD, is the half-done mutation check functionality. It was possible to import a LOVD generated cDNA reference sequence into the database, after which new submissions were compared to the known sequence data. This 'feature' was no longer maintained and did not work very well. A new project within our department, Mutalyzer, was pointed out as the replacement for this functionality. Following extensive development of Mutalyzer this year, it is ready to take over the mutation check functionality within LOVD. We plan to remove the existing code from LOVD and have the 1.1.0 to 2.0 upgrade delete all loaded sequence data from the database. Mutalyzer will not be available for download, but communication from LOVD to a Mutalyzer installation will be available as a LOVD module.

Upgrade
We offer curators to migrate their variant data to a LOVD v.2.0 install on our server in Leiden. The curators then only need to download the variant data from their LOVD v.1.1.0 installs, after which we will import it into LOVD 2.0.

Upgrading existing 1.1.0 installs to 2.0 requires intervention from the install's LOVD database manager. Since the file structure of LOVD v.2.0 is different from the 1.1.0 installations, the LOVD database manager should first move all 1.1.0 files, unpack the 2.0 files in place, fill in the new config.ini file, upload the 2.0 files and start an upgrade script which will upgrade the database contents of the 1.1.0 installation to 2.0. Because of the tremendous amount of work that needs to go into this upgrade script, it is very likely that LOVD v.2.0 will first be available without it.
It will be necessary to confront the database manager with discrepancies in the data that does not fit within the 2.0 version. The database manager will usually be provided a default action. The upgrade script will have to process, amongst others, these changes: alter all standard LOVD tables to the updated ones, migrate any installed custom columns to the new system, reconfigure any installed custom links, fill in new settings, reconfigure current settings, process submitter data, and split all old LOVD sequence variant data into variant data and patient data and migrate them to the new gene and patient database table structure.

The upgrade script should be checked on copies of as many known LOVD installations as possible, to make sure no LOVD v.1.1.0 user will get into trouble after upgrading. The manual and quick reference guide also have to be updated.

Table structure (draft version – LOVD v.2.0-pre-07)

```
/*******************************************************************************
 *
 * LOVD MySQL TABLES
 *
 * Modified    : 2006-06-08
 * Version     : 2.0-pre-07
 *
 *************/

  'CREATE TABLE ' . TABLE_USERS . ' (
   userid TINYINT(3) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   institution VARCHAR(75) NOT NULL,
   email VARCHAR(75) NOT NULL,
   username VARCHAR(20) NOT NULL,
   password CHAR(32) NOT NULL,
   password_autogen CHAR(32) NOT NULL,
   phpsessid CHAR(32) NOT NULL,
   current_db VARCHAR(12) NOT NULL,
   curate_dbs VARCHAR(255) NOT NULL,
   refresh_list TINYINT(1) UNSIGNED NOT NULL,
   list_size SMALLINT(4) UNSIGNED NOT NULL,
   level TINYINT(1) UNSIGNED NOT NULL,
   allowed_ip VARCHAR(255) NOT NULL,
   login_attempts TINYINT(1) UNSIGNED NOT NULL,
   last_login DATETIME,
   deleted TINYINT(1) NOT NULL,
   created_by TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
   created_date DATETIME NOT NULL,
   edited_by TINYINT(3) UNSIGNED ZEROFILL,
   edited_date DATETIME)'

  'CREATE TABLE ' . TABLE_SUBS . ' (
   submitterid SMALLINT(5) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT PRIMARY KEY,
   firstname VARCHAR(50) NOT NULL,
   lastname VARCHAR(50) NOT NULL,
   institute VARCHAR(75) NOT NULL,
   department VARCHAR(75) NOT NULL,
   address TEXT NOT NULL,
   city VARCHAR(50) NOT NULL,
   country VARCHAR(50) NOT NULL,
   email VARCHAR(200) NOT NULL,
   telephone VARCHAR(50) NOT NULL,
   username VARCHAR(20) NOT NULL,
   password CHAR(32) NOT NULL,
   password_autogen CHAR(32) NOT NULL,
   submits SMALLINT(5) NOT NULL,
   login_attempts TINYINT(1) UNSIGNED NOT NULL,
   last_login DATETIME,
   deleted TINYINT(1) NOT NULL,
   created_by TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
   created_date DATETIME NOT NULL,
   edited_by TINYINT(3) UNSIGNED ZEROFILL,
   edited_date DATETIME)'

  'CREATE TABLE ' . TABLE_CONFIG . ' (
   system_title VARCHAR(255) NOT NULL,
   location_name VARCHAR(255) NOT NULL,
   email_address VARCHAR(75) NOT NULL,
   send_signature TINYINT(1) UNSIGNED NOT NULL,
   send_stats TINYINT(1) UNSIGNED NOT NULL,
   send_listings TINYINT(1) UNSIGNED NOT NULL,
   use_cookies TINYINT(1) UNSIGNED NOT NULL,
   my_submissions TINYINT(1) UNSIGNED NOT NULL,
   lock_curators TINYINT(1) UNSIGNED NOT NULL,
   lock_submitters TINYINT(1) UNSIGNED NOT NULL,
   lock_uninstall TINYINT(1) UNSIGNED NOT NULL,
   technique_list TEXT NOT NULL,
   technique_url VARCHAR(255) NOT NULL)'

  'CREATE TABLE ' . TABLE_STATUS . ' (
   lock_update TINYINT(1) UNSIGNED NOT NULL,
   version VARCHAR(15) NOT NULL,
   signature CHAR(32) NOT NULL,
   update_check DATETIME,
   update_version VARCHAR(15),
   update_level TINYINT(1) UNSIGNED,
   update_description VARCHAR(255),
   date_install DATE NOT NULL,
   date_updated DATE,
   upgrade_progress TINYINT(2) NOT NULL)'
```

```
/*******************************************************************************
 *
 * LOVD MySQL TABLES (Continued)
 *
 *************/

  'CREATE TABLE ' . TABLE_COLS . ' (
   colid VARCHAR(100) NOT NULL PRIMARY KEY,
   head_column VARCHAR(20) NOT NULL,
   description_form VARCHAR(255) NOT NULL,
   description_legend_short VARCHAR(255) NOT NULL,
   description_legend_full TEXT NOT NULL,
   mysql_type VARCHAR(20) NOT NULL,
   form_type VARCHAR(100) NOT NULL,
   select_options TEXT NOT NULL,
   preg_pattern VARCHAR(100) NOT NULL,
   public TINYINT(1) UNSIGNED NOT NULL,
   public_form TINYINT(1) UNSIGNED NOT NULL,
   created_by TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
   created_date DATETIME NOT NULL,
   edited_by TINYINT(3) UNSIGNED ZEROFILL,
   edited_date DATETIME)'

  'CREATE TABLE ' . TABLE_LINKS . ' (
   linkid TINYINT(3) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT PRIMARY KEY,
   linkname VARCHAR(50) NOT NULL,
   pattern_text VARCHAR(50) NOT NULL,
   replace_text TEXT NOT NULL,
   link_description TEXT NOT NULL,
   active TINYINT(1) UNSIGNED NOT NULL)'

  'CREATE TABLE ' . TABLE_COLS2LINKS . ' (
   colid VARCHAR(100) NOT NULL,
   linkid TINYINT(3) UNSIGNED NOT NULL,
   UNIQUE(colid, linkid))'

  'CREATE TABLE ' . TABLE_LOGS . ' (
   logname VARCHAR(10) NOT NULL,
   date DATETIME NOT NULL,
   mtime MEDIUMINT(6) UNSIGNED ZEROFILL NOT NULL,
   event VARCHAR(12) NOT NULL,
   log TEXT NOT NULL,
   UNIQUE (logname, date, mtime))'

  'CREATE TABLE ' . TABLE_MODULES . ' (
   moduleid VARCHAR(10) NOT NULL UNIQUE,
   version VARCHAR(15) NOT NULL,
   active TINYINT(1) NOT NULL,
   date_install DATE NOT NULL,
   date_updated DATE)'

  'CREATE TABLE ' . TABLE_DBS . ' (
   symbol VARCHAR(12) NOT NULL UNIQUE,
   gene VARCHAR(75) NOT NULL,
   chrom_location VARCHAR(20) NOT NULL,
   url_homepage VARCHAR(150) NOT NULL,
   url_external TEXT NOT NULL,
   allow_download TINYINT(1) UNSIGNED NOT NULL,
   id_entrez INT(10) UNSIGNED NOT NULL,
   id_omim_gene INT(10) UNSIGNED NOT NULL,
   id_omim_disease TEXT NOT NULL,
   id_hgmd VARCHAR(12) NOT NULL,
   id_gdb INT(10) UNSIGNED NOT NULL,
   note_index TEXT NOT NULL,
   note_listing TEXT NOT NULL,
   ref_exon_numbering VARCHAR(255) NOT NULL,
   genbank TINYINT(1) NOT NULL,
   genbank_url VARCHAR(25) NOT NULL,
   refseq VARCHAR(1) NOT NULL,
   refseq_url VARCHAR(255) NOT NULL,
   disease_list TEXT NOT NULL,
   disease_url VARCHAR(255) NOT NULL,
   created_by TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
   created_date DATETIME NOT NULL,
   edited_by TINYINT(3) UNSIGNED ZEROFILL,
   edited_date DATETIME)'

  'CREATE TABLE ' . TABLE_PATIENTS . ' (
   patientid MEDIUMINT(7) UNSIGNED ZEROFILL NOT NULL AUTO_INCREMENT PRIMARY KEY,
   status TINYINT(1) UNSIGNED NOT NULL,
   created_by TINYINT(3) UNSIGNED ZEROFILL NOT NULL,
   created_date DATETIME NOT NULL,
   edited_by TINYINT(3) UNSIGNED ZEROFILL,
   edited_date DATETIME)'
```

```
/*****************************************************************************
 *
 * LOVD MySQL TABLES (Continued)
 *
 *************/

   'CREATE TABLE ' . TABLE_PAT2VAR . ' (
    patientid MEDIUMINT(7) UNSIGNED NOT NULL,
    allele ENUM("1", "2") NOT NULL,
    symbol VARCHAR(12) NOT NULL,
    variantid MEDIUMINT(7) UNSIGNED NOT NULL,
    UNIQUE(patientid, allele, symbol, variantid))'
```