Documentation Library

Gamma/MySQL

Using Gamma™ with MySQL, Version 1.1

Cogent Real-Time Systems, Inc.

August 11, 2010

Gamma/MySQL: Using Gamma™ with MySQL, Version 1.1

Gamma binds a useful subset of MySQL functions, and gives object-oriented access to a MySQL database. The Gamma/MySQL bindings and this document are being made available as a beta release.

Published August 11, 2010
Cogent Real-Time Systems, Inc.
162 Guelph Street, Suite 253
Georgetown, Ontario
Canada, L7G 5X7

Toll Free: 1 (888) 628-2028
Tel: 1 (905) 702-7851
Fax: 1 (905) 702-7850

Information Email: info@cogent.ca
Tech Support Email: support@cogent.ca
Web Site: www.cogent.ca

Revision History

Revision 1.1-1 August 2004
Compatible with Cascade DataHub and Cascade Connect Version 5.0.
Revision 1.0-1 April 2003
Inital beta release.

Copyright, trademark, and software license information.

Copyright Notice

Trademark Notice

END-USER LICENSE AGREEMENT FOR COGENT BETA SOFTWARE

IMPORTANT - READ CAREFULLY: This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Cogent Real-Time Systems Inc. ("Cogent") of 162 Guelph Street, Suite 253, Georgetown, Ontario, L7G 5X7, Canada (Tel: 905-702-7851, Fax: 905-702-7850), from whom you acquired the Cogent beta software product(s) ("SOFTWARE PRODUCT" or "SOFTWARE"), either directly from Cogent or through one of Cogent's authorized resellers. The SOFTWARE PRODUCT includes computer software, any associated media, any printed materials, and any "online" or electronic documentation. By installing, copying or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree with the terms of this EULA, Cogent is unwilling to license the SOFTWARE PRODUCT to you. In such event, you may not use or copy the SOFTWARE PRODUCT, and you should promptly contact Cogent for instructions on return of the unused product(s) for a refund.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. **GRANT OF LICENSE:** This EULA grants you the following non-exclusive rights:

   - Evaluation Beta Software. You may use the SOFTWARE PRODUCT for evaluation purposes only, unless authorized and permitted, in writing by Cogent.

   Subject to the license expressly granted above, you obtain no right, title or interest in or to the SOFTWARE PRODUCT or related documentation, including but not limited to any copyright, patent, trade secret or other proprietary rights therein. All whole or partial copies of the SOFTWARE PRODUCT remain property of Cogent and will be considered part of the SOFTWARE PRODUCT for the purpose of this EULA.

   Unless expressly permitted under this EULA or otherwise by Cogent, you will not:

   i. use, reproduce, modify, adapt, translate or otherwise transmit the SOFTWARE PRODUCT or related components, in whole or in part;

   ii. rent, lease, license, transfer or otherwise provide access to the SOFTWARE PRODUCT or related components;

   iii. alter, remove or cover proprietary notices in or on the SOFTWARE PRODUCT, related documentation or storage media;

   iv. export the SOFTWARE PRODUCT from the country in which it was provided to you by Cogent or its authorized reseller;

   v. use a multi-processor version of the SOFTWARE PRODUCT in a network larger than that for which you have paid the corresponding multi-processor fees;

   vi. decompile, disassemble or otherwise attempt or assist others to reverse engineer the SOFTWARE PRODUCT;

   vii. circumvent, disable or otherwise render ineffective any demonstration time-outs, locks on functionality or any other restrictions on use in the SOFTWARE PRODUCT;

   viii. circumvent, disable or otherwise render ineffective any license verification mechanisms used by the SOFTWARE PRODUCT, or

ix. use the SOFTWARE PRODUCT in any application that is intended to create or could, in the event of malfunction or failure, cause personal injury or property damage.

2. **NO WARRANTY:** Cogent cannot warrant that the SOFTWARE PRODUCT will function in accordance with related documentation in every combination of hardware platform, software environment and SOFTWARE PRODUCT configuration. You acknowledge that software bugs are likely to be identified when the SOFTWARE PRODUCT is used in your particular application. You therefore accept the responsibility of satisfying yourself that the SOFTWARE PRODUCT is suitable for your intended use. This includes conducting exhaustive testing of your application prior to its initial release and prior to the release of any related hardware or software modifications or enhancements.
3. **LIMITATIONS:** The SOFTWARE PRODUCT, any related documentation and disks are provided "as is" without other warranties or conditions of any kind, including but not limited to implied warranties of merchantability, fitness for a particular purpose and non-infringement. You assume the entire risk as to the results and performance of the SOFTWARE PRODUCT. Nothing stated in this EULA will imply that the operation of the SOFTWARE PRODUCT will be uninterrupted or error free or that errors will be corrected. Other written or oral statements by Cogent, its representatives or others do not constitute warranties or conditions of Cogent.
4. **DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS:**

   Separation of Components. The SOFTWARE PRODUCT is licensed as a single product. Its component parts may not be separated for use on more than one computer.

   Termination. Without prejudice to any other rights, Cogent may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such an event, you must destroy all copies of the SOFTWARE PRODUCT and all of its component parts.

5. **UPGRADES:** If the SOFTWARE PRODUCT is an upgrade from another product, whether from Cogent or another supplier, you may use or transfer the SOFTWARE PRODUCT only in conjunction with that upgrade product, unless you destroy the upgraded product. If the SOFTWARE PRODUCT is an upgrade of a Cogent product, you now may use that upgraded product only in accordance with this EULA. If the SOFTWARE PRODUCT is an upgrade of a component of a package of software programs which you licensed as a single product, the SOFTWARE PRODUCT may be used and transferred only as part of that single product package and may not be separated for use on more than one computer.
6. **COPYRIGHT:** All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text and ’applets", incorporated into the SOFTWARE PRODUCT), any accompanying printed material, and any copies of the SOFTWARE PRODUCT, are owned by Cogent or its suppliers. You may not copy the printed materials accompanying the SOFTWARE PRODUCT. All rights not specifically granted under this EULA are reserved by Cogent.
7. **PRODUCT SUPPORT:** Cogent has no obligation under this EULA to provide maintenance, support or training.
8. **RESTRICTED RIGHTS:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (OCT 1988), FAR 12.212(a)(1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as appropriate. Manufacturer is Cogent Real-Time Systems Inc. 162 Guelph Street, Suite 253, Georgetown, Ontario, L7G 5X7, Canada.
9. **GOVERNING LAW:** This Software License Agreement is governed by the laws of the Province of Ontario, Canada. You irrevocably attorn to the jurisdiction of the courts of the Province of Ontario and agree to commence any litigation that may arise hereunder in the courts located in the Judicial District of Peel, Province of Ontario.

Table of Contents

Chapter 1. Introduction

1.1. System Requirements

Gamma/MySQL was compiled against MySQL Version 3.23, and should be compatible with MySQL version 3.*xx* and 4.*xx*. If you encounter difficulties, please contact Cogent.

Gamma/MySQL can run on any of the following operating systems:

- QNX 6.1.0 or later.
- Linux 2.4 or later, optionally with:

  The SRR IPC kernel module, which includes a synchronous message passing library modeled on the QNX 4 send/receive/reply message-passing API. This module installs automatically, but requires a C compiler for the installation. You can get more information and/or download this module at the Cogent Web Site.

  

  This module may not be necessary for some Gamma applications, but it is required for any use of timers, event handling, or inter-process communication.

1.2. Cogent Product Integration

Cogent products work together to support real-time data connectivity in Windows, Linux, and QNX. They can be dynamically integrated as a group of modules where each module connects to any other module(s) as needed. New modules can be added and existing modules reconfigured or modified, all during run-time. Data in any module of the system can be collected and redistributed to any other module via the Cascade DataHub and Cascade Connect. Communication with field devices is provided by one of several Cogent Device Drivers. Historical records of unlimited size can be maintained and queried with the Cascade Historian, and ASCII text files can be logged with the Cascade TextLogger.

Custom programs written in C or C++ can interface with the system, using the Cogent C API or the DataHub APIs for C++, Java, and .NET. In addition, Cogent's own dynamically-typed object-oriented programming language, Gamma, is fully compatible with all modules. User interfaces can be created in Gamma, which supports Photon in QNX and GTK in Linux.

1.3. Where can I get help?

If you are having problems with a Cogent product, first check the Troubleshooting Guide. If you can't find the answer there, you can contact Cogent Real-Time Systems, Inc. for technical support for any product you have purchased.

- Email: <`support@cogent.ca`>
- Phone: 1-888-628-2028
- Fax: (905) 702-7850

Chapter 2. Tutorials

2.1. Tutorial 1: Interactive Session

Gamma code can be run in two ways: in interactive mode, and as an executable program. This first tutorial demonstrates an interactive session and introduces some of the basic Gamma/MySQL methods and functions needed to get started.

1. To begin, we start Gamma at a shell prompt and then load the Gamma/MySQL support library.

```
    [sh]$ gamma
    This software is free for non-commercial use, and no valid commercial license
is installed.  For more information, please contact info@cogent.ca.

Gamma(TM) Advanced Programming Language

Copyright (C) Cogent Real-Time Systems Inc., 1995-2005.  All rights reserved.
Gamma Version 5.2 Build 5 at Feb 11 2005 10:21:37
    Gamma> require ("MySQLSupport");
    "/usr/cogent/require/MySQLSupport.g"
```

   This `require` statement loads the `MySQLSupport.g` library, which contains all the Gamma/MySQL methods and functions.

2. Next we create an instance of the `MYSQL` class, using the Gamma function `new`.

```
    Gamma> mysql = new MYSQL();
    {MYSQL (client_flag . 0) (connector_fd) (db) (field_alloc .
    ... rest of class definition}
```

   The `MYSQL` class supports the connection to the database, and most of the methods and functions documented in this manual relate to this class.

3. The next step is to actually make the connection, using the `MYSQL.connect` method.

```
    Gamma> mysql.connect ("localhost", "test", "");
    {MYSQL (client_flag . 8325) (connector_fd) (db) (field_alloc .
    ...rest of class definition}
```

   The `MYSQL.connect` method has three parameters: *`host`*, *`user`*, and *`password`*. We have chosen `localhost` for the *`host`*, `test` for the *`user`*, and an empty string for the *`password`* so that this example can be run on any host by any user.

4. Once the connection is made, we can create a new database, select it, and then create a table for it.

```
    Gamma> mysql.query(string("CREATE DATABASE ", "test_interactive"));
    0
    Gamma> mysql.select_db ("test_interactive");
    0
    Gamma> try mysql.create_table("friends"); catch;
    nil
```

   Creating the database was done with the `MYSQL.query` method, which can be used to pass SQL commands to MySQL. However, there is no similar SQL command for tables available in the version of MySQL we are currently running, so we use the `.create_table` method for tables. We surround this statement with a Gamma `try/catch` statement, which lets us ignore an error and continue if the table already exists. This is essentially the same as using the `IF NOT EXISTS` command option. (We could have also used the `.create_db` method in a similar way to create the database.)

5. Now that a table is created, we can add columns to it, using Gamma's `MYSQL.add_column` method.

```
    Gamma> mysql.add_column ("friends", "name", "VARCHAR(20)", nil, nil);
    nil
    Gamma> mysql.add_column ("friends", "phone", "VARCHAR(20)", nil, nil);
```

```
nil
Gamma> mysql.add_column ("friends", "email", "VARCHAR(20)", nil, nil);
nil
```

6. The next step is to populate the table with data. Gamma uses an object-oriented approach to the data in a MySQL database, which is achieved by first creating a class named `Friend` from the `friends` table, using the `MYSQL.class_from_table` method.

```
Gamma> mysql.class_from_table (#Friend, nil, "friends");
(defclass Friend nil [(__columns . #0=[id name phone email]) (__primary_key . id)
(__table . #1="friends")][email id name phone])
```

   The class variables: `__columns`, `__primary_key`, and `__table` contain the essential information about the table. The instance variables are named after the column names; in this case: `email`, `id`, `name`, and `phone`, with the `id` variable being added automatically, and incremented as necessary. Each instance of this class corresponds to one row of the table. For example, to put a new row into the table, we simply make an instance of the class, and assign the instance variables (except `id`), as shown in the next step.

7. Once the `Friend` class has been created, we can create instances and assign values to them. Each instance corresponds to a row in the `friends` database.

```
Gamma> newrow = new(Friend);
{Friend (email) (id) (name) (phone)}
Gamma> newrow.name = "Kisbet Grimes";
"Kisbet Grimes"
Gamma> newrow.phone = "414-595-3389";
"414-595-3389"
Gamma> newrow.email = "kisbetg@netmail.com";
"kisbetg@netmail.com"
Gamma> mysql.insert(newrow);
nil
Gamma> newrow = new(Friend);
{Friend (email) (id) (name) (phone)}
Gamma> newrow.name = "Merle Royer";
"Merle Royer"
Gamma> newrow.phone = "605-783-2045";
"605-783-2045"
Gamma> newrow.email = "mark55@interisp.com";
"mark55@interisp.com"
Gamma> mysql.insert(newrow);
nil
```

   The Gamma function `new` is used to create instances of the class. As each instance is created, the system automatically assigns the `id` number. The Gamma method `MYSQL.insert` is used to insert the row into the table.

8. With values in the database, perhaps now we want to make a query. Gamma/MySQL provides several ways of doing this, one of which is the `MYSQL.query_and_store` method.

```
Gamma> all_friends = mysql.query_and_store("SELECT * FROM friends");
{MYSQL_RES (current_field . 0) (eof . 1) (field_count . 4)
(fields . [{MYSQL_FIELD (decimals . 0) (def) (flags . 49667)
                        (length . 11) (max_length . 1) (name . "id")
                        (table . "friends") (type . 3)}
           {MYSQL_FIELD (decimals . 0) (def) (flags . 1) (length . 20)
                        (max_length . 13) (name . "name") (table . "friends")
                        (type . 253)} {MYSQL_FIELD (decimals . 0) (def)
                        (flags . 1) (length . 20) (max_length . 12) (name . "phone")
                        (table . "friends") (type . 253)}
           {MYSQL_FIELD (decimals . 0) (def) (flags . 1) (length . 20)
                        (max_length . 19) (name . "email") (table . "friends")
                        (type . 253)}])}
```

   The results of the query are returned as an instance of the `MYSQL_RES` class. The `fields` instance variable of that class holds the data in an array of `MYSQL_FIELD` instances, one instance per row of data.

9. To see the query results, we use the Gamma `mysql_pretty_print` function.

```
Gamma> mysql_pretty_print(stdout, all_friends);
id name          phone        email
-- ------------- ------------ ------------------
 1 Kisbet Grimes 414-595-3389 kisbetg@netmail.com
 2 Merle Royer   605-783-2045 mark55@interisp.com
nil
```

10. At the end of our tutorial session, we can drop (remove) our test database. This makes it possible to run the tutorial again later.

```
Gamma> mysql.query(string("DROP DATABASE ", "test_interactive"));
0
Gamma>
```

    Dropping the database can be done with the `MYSQL.query` method, as shown above. If the `DROP DATABASE` command is not available for some reason, you can use the the `MYSQL.drop_db` method.

These are the fundamentals of using Gamma/MySQL. The next tutorial (Section 2.2, *Tutorial 2: `create_pets.g`*) is an executable program that follows a similar sequence, and introduces more Gamma/MySQL methods and functions. As a review and for your convenience, all the input for the tutorial you have just completed is as follows:

```
require ("MySQLSupport");
mysql = new MYSQL();
mysql.connect ("localhost", "test", "");
mysql.query(string("CREATE DATABASE ",  "test_interactive"));
mysql.select_db ("test_interactive");
try mysql.create_table("friends"); catch;
mysql.add_column ("friends", "name", "VARCHAR(20)", nil, nil);
mysql.add_column ("friends", "phone", "VARCHAR(20)", nil, nil);
mysql.add_column ("friends", "email", "VARCHAR(20)", nil, nil);
mysql.class_from_table (#Friend, nil, "friends");
newrow = new(Friend);
newrow.name = "Kisbet Grimes";
newrow.phone = "414-595-3389";
newrow.email = "kisbetg@netmail.com";
mysql.insert(newrow);
newrow = new(Friend);
newrow.name = "Merle Royer";
newrow.phone = "605-783-2045";
newrow.email = "mark55@interisp.com";
mysql.insert(newrow);
all_friends = mysql.query_and_store("SELECT * FROM friends");
mysql_pretty_print(stdout, all_friends);
mysql.query(string("DROP DATABASE ", "test_interactive"));
```

2.2. Tutorial 2: `create_pets.g`

This tutorial demonstrates some of the Gamma-specific MySQL methods. It uses the `petdata.txt` data file (shown below) to recreate a `pets` table similar to the one in the MySQL documentation tutorial.

```
/* Get the Gamma library functions and methods for MySQL. */
require ("MySQLSupport");

/* Set up variables.  To make the example most universal, we define
the host, user, and password as seen below.  These make the database
and tables available to all users, and let them be changed by
anyone. */
host := "localhost";
user := "test";
password := "";
database := "test_pets";
```

```
function main ()
{

  /* Make a connection to the database by creating a new instance
     of the MYSQL class, and localize all variables.*/
  local mysql = new MYSQL();
  local allpets, query, requery, removed_row, newrow, inserted, deleted;
  local query_data, mapped_array, field_array;

  /* Test the connection. */
  if (!mysql.connect (host, user, password))
    error (string ("Could not connect to host: ", host,
     " as user: ", user));

  /* Remove any previously-created database with the name we want to use
     and then create and select the database.*/
  mysql.query(string("DROP DATABASE IF EXISTS ", database));
  mysql.query(string("CREATE DATABASE ", database));
  mysql.select_db (database);

  /* Remove any previously-created table called "pets" using the MYSQL.drop_table
     method, in a try/catch statement.  This is essentially the same as issuing
     the command: mysql.query("DROP TABLE IF EXISTS pets");*/
  try mysql.drop_table ("pets"); catch;

  /* Create the "pets" table, and add columns to it. */
  mysql.create_table("pets");
  mysql.add_column ("pets", "name", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "owner", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "species", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "sex", "CHAR(1)", nil, nil);
  mysql.add_column ("pets", "birth", "DATE", nil, t);
  mysql.add_column ("pets", "death", "DATE", nil, t);

  /* Load the data from the "petdata.txt" file. */
  mysql.query("LOAD DATA LOCAL INFILE 'petdata.txt' INTO TABLE pets");

  /* Test the MYSQL.query_and_store method, and print the whole table. */
  allpets = mysql.query_and_store("SELECT * FROM pets");
  princ("QUERY  The .query_and_store method returns: \n");
  mysql_pretty_print(stdout, allpets);

  /* Test the MYSQL.class_from_table method. */
  mysql.class_from_table (#mypetclass, nil, "pets");
  pretty_princ ("\nCLASS  The .class_from_table method ",
  "returns the mypetclass:\n", mypetclass, "\n");

  /* Test the MYSQL.query_to_temp_class method. */
  query = mysql.query_to_temp_class (mypetclass, "select * from pets where sex = 'm'");
  pretty_princ ("\nQUERY  The .query_to_temp_class method applied ",
  "to males in mypetclass returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is: ", query[0].name, "\n");

  /* Test the MYSQL.query_to_class method. */
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("\nQUERY  The .query_to_class method applied ",
  "to females in mypetclass returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is: ", query[0].name, "\n");

  /* Create a new row. */
  newrow = new(mypetclass);
  newrow.name = "Petunia";
  newrow.owner = "Warner Bros.";
  newrow.species = "pig";
  newrow.sex = "f";
  newrow.birth = "1925-12-25";
  newrow.death = "\N";
```

```
  /* Test the MYSQL.insert method. */
  inserted = mysql.insert(newrow);
  pretty_princ ("\nINSERT  After inserting this new row:\n", newrow, "\n");
  pretty_princ ("The .insert method returns:\n", inserted, "\n");
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("And now the .query_to_class method ",
  "applied to females in mypetclass returns:\n", query, "\n");

  /* Test the MYSQL.delete method. */
  remove_row = query[0];
  deleted = mysql.delete(remove_row);
  pretty_princ ("\nDELETE  After deleting this row:\n", remove_row, "\n");
  pretty_princ ("The .delete method returns:\n", deleted, "\n");
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("And a query_to_class on females ",
  "in mypetclass now returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is now: ", query[0].name, "\n");

  /* Test the MYSQL.requery method. */
  requery = mysql.requery(query[1]);
  pretty_princ ("\nREQUERY  The .requery method applied to the ",
  "second class in the above query returns:\n", requery, "\n");

  /* Test the MYSQL.query_and_store method. */
  query = mysql.query_and_store ("select name, birth from pets");
  pretty_princ("\nQUERY  The .query_and_store method applied ",
  "to the 'name' and 'birth' fields in mypetclass ",
        "\nreturns this MYSQL_RES istance:\n", query, "\n");
  query_data = query.Data();
  pretty_princ("Its data, recovered using ",
        "the .Data() method, is:\n", query_data, "\n");

  princ("\nTest complete.\n");
}
```

Input file: `petsdata.txt`

```
 Fluffy Harold cat f 1993-02-04 \N
 Claws Gwen cat m 1994-03-17 \N
 Buffy Harold dog \N 1989-05-13 \N
 Fang Benny dog m 1990-08-27 \N
 Bowser Diane dog m 1995-08-31 2002-07-29
 Chirpy Gwen bird f 1998-09-11 \N
 Whistler Gwen bird \N 1997-12-09 \N
 Slim Benny snake m 1996-04-29 \N
```

There is a tab at the beginning of each line, to allow for the auto-insertion of `id` field values.

Output

```
QUERY  The .query_and_store method returns:
id name     owner  species sex birth      death
-- -------- ------ ------- --- ---------- ----------
 1 Fluffy   Harold cat     f   1993-02-04 nil
 2 Claws    Gwen   cat     m   1994-03-17 nil
 3 Buffy    Harold dog         1989-05-13 nil
 4 Fang     Benny  dog     m   1990-08-27 nil
 5 Bowser   Diane  dog     m   1995-08-31 2002-07-29
 6 Chirpy   Gwen   bird    f   1998-09-11 nil
 7 Whistler Gwen   bird        1997-12-09 nil
 8 Slim     Benny  snake   m   1996-04-29 nil

CLASS  The .class_from_table method returns the mypetclass:
(defclass mypetclass nil
[(__columns . #0=[id name owner species sex birth death])
 (__primary_key . id) (__table . #1=pets)]
[birth death id name owner sex species])
```

```
QUERY  The .query_to_temp_class method applied to males in mypetclass returns:
[{TempQuery (birth . 1994-03-17) (death) (id . 2) (name . Claws)
     (owner . Gwen) (sex . m) (species . cat)}
 {TempQuery (birth . 1990-08-27) (death) (id . 4) (name . Fang)
     (owner . Benny) (sex . m) (species . dog)}
 {TempQuery (birth . 1995-08-31) (death . 2002-07-29) (id . 5)
     (name . Bowser) (owner . Diane) (sex . m) (species . dog)}
 {TempQuery (birth . 1996-04-29) (death) (id . 8) (name . Slim)
     (owner . Benny) (sex . m) (species . snake)}]
The name of the first pet is: Claws

QUERY  The .query_to_class method applied to females in mypetclass returns:
[{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
      (owner . Harold) (sex . f) (species . cat)}
 {mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}]
The name of the first pet is: Fluffy

INSERT  After inserting this new row:
{mypetclass (birth . 1925-12-25) (death . N) (id . 9) (name . Petunia)
     (owner . Warner Bros.) (sex . f) (species . pig)}
The .insert method returns:
nil
And now the .query_to_class method applied to females in mypetclass returns:
[{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
      (owner . Harold) (sex . f) (species . cat)}
 {mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}
 {mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
      (name . Petunia) (owner . Warner Bros.) (sex . f)
      (species . pig)}]

DELETE  After deleting this row:
{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
     (owner . Harold) (sex . f) (species . cat)}
The .delete method returns:
nil
And a query_to_class on females in mypetclass now returns:
[{mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}
 {mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
      (name . Petunia) (owner . Warner Bros.) (sex . f)
      (species . pig)}]
The name of the first pet is now: Chirpy

REQUERY  The .requery method applied to the second class in the above query returns:
{mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
     (name . Petunia) (owner . Warner Bros.) (sex . f)
     (species . pig)}

QUERY  The .query_and_store method applied to the 'name' and 'birth' fields in mypetclass
returns this MYSQL_RES istance:
{MYSQL_RES (current_field . 0) (eof . 1) (field_count . 2)
    (fields . [{MYSQL_FIELD (decimals . 0) (def) (flags . 1)
       (length . 20) (max_length . 8) (name . name)
       (table . pets) (type . 253)}
        {MYSQL_FIELD (decimals . 0) (def) (flags . 0) (length . 10)
       (max_length . 10) (name . birth) (table . pets) (type . 10)}])}
Its data, recovered using the .Data() method, is:
[[Claws 1994-03-17] [Buffy 1989-05-13] [Fang 1990-08-27]
 [Bowser 1995-08-31] [Chirpy 1998-09-11] [Whistler 1997-12-09]
 [Slim 1996-04-29] [Petunia 1925-12-25]]

Test complete.
```

2.3. Tutorial 3: `create_foodsinc.g`

This example creates a database, enters and deletes date, and then emits a unique query to the database and maps the results to instances of a temporary class.

```
#!/usr/cogent/bin/gamma
/*
 * Create a new table in a mysql database
 */

require ("MySQLSupport");

host := "localhost";
user := "test";
password := "";
database := "test_foodsinc";

/* Derive a class from MYSQL just to prove that we can
   do it.  There is no real need in this case, but if we
   want to specialize the connection somehow, we can. */

class Demo MYSQL
{
}

method Demo.make_row (klass, name, address, phone)
{
  local row = new klass();

  row.name = name;
  row.address = address;
  row.phone = phone;
  .insert (row);
  row;
}

function main ()
{
  local  mysql = new Demo();
  local  cust, sup, mistakes, common;

  if (!mysql.connect (host, user, password))
    error (string ("Could not connect to host: ", host,
     " as user: ", user));

  if (mysql.select_db (database) == -1)
    mysql.query(string("CREATE DATABASE ", database));

  if (mysql.select_db (database) == -1)
    error (string ("Could not select or create database: ", database));

  /* Drop the test tables.  Ignore errors. */
  try mysql.drop_table ("customers"); catch;
  try mysql.drop_table ("suppliers"); catch;

  /* Create new customers and suppliers tables. */
  mysql.create_table ("customers");
  mysql.add_column ("customers", "name", "TEXT", nil, nil);
  mysql.add_column ("customers", "address", "TEXT", nil, t);
  mysql.add_column ("customers", "phone", "TEXT", nil, t);

  mysql.create_table ("suppliers");
  mysql.add_column ("suppliers", "name", "TEXT", nil, nil);
  mysql.add_column ("suppliers", "address", "TEXT", nil, t);
  mysql.add_column ("suppliers", "phone", "TEXT", nil, t);

  /* Create Gamma classes for Customer and Supplier, using the
     MySQL tables "customers" and "suppliers" as templates for
     the classes. */
```

```
mysql.class_from_table (#Customer, nil, "customers");
mysql.class_from_table (#Supplier, nil, "suppliers");

/* Create several customers.  Hang onto one of them for later
   manipulation. */
cust = mysql.make_row (Customer, "Baker Bob's Pastries",
  "New York", "111-555-1212");
mysql.make_row (Customer, "Willy Wonka's Chocolate Factory",
  "Somewhere in Kansas", "222-555-1212");
mysql.make_row (Customer, "This is a mistake",
  "Nowhere", "000-000-0000");

/* Create several suppliers. */
mysql.make_row (Supplier, "Sugar Hill \"North\" Farms",
  "Kingston, Jamaica", "333-555-1212");
sup = mysql.make_row (Supplier, "Mack and Molly Milk Farms",
 "Dublin, Ireland", "777-555-1212");
mysql.make_row (Supplier, "Baker Bob's Pastries",
  "New York", "111-555-1212");
mysql.make_row (Supplier, "Oops, another mistake",
  "Anywhere", "999-555-1212");

/* Change the customer phone number for Baker Bob */
cust.phone = "444-555-1212";
mysql.update(cust);

/* Change the city for Mack and Molly */
sup.address = "Clonmacnoise, Ireland";
mysql.update(sup);

/* Remove all customers whose address is Nowhere */
mistakes = mysql.query_to_class
  (Customer, "select * from customers where address = \"Nowhere\"");
with cust in mistakes do
  {
    princ ("Deleting ", cust.name, "\n");
    mysql.delete (cust);
  }

/* Remove all suppliers whose name includes the word "mistake" */
mistakes = mysql.query_to_class
  (Supplier, "select * from suppliers where name like \"%mistake%\"");
with sup in mistakes do
  {
    princ ("Deleting ", sup.name, "\n");
    mysql.delete (sup);
  }

/* Find all customers who are also suppliers, using the name as
   the comparison field.  Print the results.  Yes, we could have
   retrieved both phone numbers in the original select, but that
   would not be as demonstrative.

   Note that column names are allowed to contain spaces and
   characters that are not valid Gamma identifier characters.  To
   work with these column names, you must escape the invalid
   characters within the instance variable identifier.  Look at
   "supplier id" for an example.  Life is easier if you avoid
   this.
   */

common = mysql.query_to_temp_class
  (nil, "select c.id as cid, s.id as \"supplier id\", c.name, c.address,
         c.phone from customers c, suppliers s where c.name = s.name");

with match in common do
  {
    princ (match.name, " is both a customer (id ", match.cid,
```

```
      ") and a supplier (id ", match.supplier\ id, ")\n");
      cust = mysql.query_to_class
 (Customer, string ("select * from customers where id = ",
      match.cid));
      sup = mysql.query_to_class
 (Supplier, string ("select * from suppliers where id = ",
      match.supplier\ id));
      princ ("  Purchasing phone number is ", cust[0].phone, "\n");
      princ ("  Sales phone number is ", sup[0].phone, "\n");
    }

}
```

Output

```
    Deleting This is a mistake
    Deleting Oops, another mistake
    Baker Bob’s Pastries is both a customer (id 1) and a supplier (id 3)
      Purchasing phone number is 444-555-1212
      Sales phone number is 111-555-1212
```

2.4. Example function: `table_list.g`

This example lists all the tables in a MySQL database.

```
require ("MySQLSupport");

host := "localhost";
user := "test";
password := "";
database := "test_gamma";

function main ()
{
  local  mysql = new MYSQL();
  local  result;

  if (!mysql.connect (host, user, password))
    error (string ("Could not connect to host: ", host,
     " as user: ", user));
  else if (mysql.select_db (database) == -1)
    error (string ("Could not select database: ", database));

  // We can use the built-in method
  princ("\nThe tables in ", database, "\nUsing the built-in method list_tables():\n");
  result = mysql.list_tables ("%");
  with row in result.Data() do
    princ (row[0], "\n");

  // ... or we can issue the query manually
  princ("Using the query_and_store method for a manual query:\n");
  result = mysql.query_and_store("show tables");
  with row in result.Data() do
    princ (row[0], "\n");
  princ ("\n");
}
```

Output

```
    The tables in test_gamma
    Using the built-in method list_tables():
    customers
    suppliers
    Using the query_and_store method for a manual query:
    customers
    suppliers
```

2.5. Example function: `query_table.g`

This example prints all of the rows of the database tables.

```
require ("MySQLSupport");

host := "localhost";
user := "test";
password := "";
database := "test_gamma";
tables := list("customers", "suppliers");

function print_query(tbl)
{
  local result = mysql.query_and_store(string("select * from ", tbl));

  princ("Table name: ", tbl, "\n");
  mysql_pretty_print (stdout, result);
  princ("\n");
}

function main ()
{
  local  mysql = new MYSQL();

  if (!mysql.connect (host, user, password))
    error (string ("Could not connect to host: ", host,
     " as user: ", user));
  else if (mysql.select_db (database) == -1)
    error (string ("Could not select database: ", database));

  princ("\n");

  with tab in tables do
    print_query(tab);

}
```

Output

```
   Table name: customers
   id name                           address             phone
   -- ------------------------------ ------------------- ------------
    1 Baker Bob’s Pastries            New York            444-555-1212
    2 Willy Wonka’s Chocolate Factory Somewhere in Kansas 222-555-1212

   Table name: suppliers
   id name                      address               phone
   -- ------------------------- --------------------- ------------
    1 Sugar Hill "North" Farms  Kingston, Jamaica     333-555-1212
    2 Mack and Molly Milk Farms Clonmacnoise, Ireland 777-555-1212
    3 Baker Bob’s Pastries      New York              111-555-1212
```

Chapter 3. Classes

MYSQL

MYSQL — the connection to the database.

Synopsis

```
class MYSQL
{
    client_flag;
    connector_fd;
    db;
    field_alloc;
    field_count;
    fields;
    free_me;
    host;
    host_info;
    info;
    options;
    packet_length;
    passwd;
    port;
    protocol_version;
    reconnect;
    server_capabilities;
    server_language;
    server_status;
    server_version;
    thread_id;
    unix_socket;
    user;
}
```

Methods

These methods are either wrapped MySQL functions, or they are unique to Gamma. The methods unique to Gamma are linked to the reference page in this manual. The methods that correspond to a MySQL function are linked to that reference page in the MySQL documenatation in the HTML version of this book.

add_column (*tablename*, *column*, *type*, *default_val*, *allow_null*)
change_user (*user*, *password*, *db*)
character_set_name ()
class_from_table (*symclassname*, *superclass*, *tablename*)
classes_from_tables (*superclass*, *verbose*?=nil)
close ()
connect (*host*, *user*, *password*)
create_db (*db*)
create_table (*tablename*)
delete (*row*)
drop_db (*db*)
drop_table (*tablename*)
dump_debug_info ()
errno ()
error ()
field_count ()
get_host_info ()
get_proto_info ()
get_server_info ()

info ()
init ()
insert (*row*)
insert_id ()
kill (*pid*)
list_dbs (*wild*)
list_fields (*table*, *wild*)
list_processes (*wild*)
list_tables (*wild*)
ping ()
query (*q*)
query_and_store (*query_string*)
query_to_class (*klass*, *query_string*)
query_to_temp_class (*superclass*, *query*)
real_connect (*host*, *user*, *passwd*, *port*, *unix_socket*, *clientflag*)
real_escape_string (*to*, *from*, *length*)
real_query (*q*, *length*)
requery (*row*)
select_db (*db*)
shutdown ()
stat ()
store_result ()
thread_id ()
update (*row*)
use_result ()

Related functions

These functions are either wrapped MySQL functions, or they are unique to Gamma. The functions unique to Gamma are linked to the reference page in this manual. The functions that correspond to a MySQL function are linked to that reference page in the MySQL documenatation in the HTML version of this book.

my_init ()
mysql_class_from_fields (*symclassname*, *superclass*, *fields*)
mysql_debug (*debug*)
mysql_eof (*res*)
mysql_escape_string (*to*, *from*, *from_length*)
mysql_fetch_field (*result*)
mysql_fetch_field_direct (*res*, *fieldnr*)
mysql_fetch_fields (res)
mysql_field_seek (*result*, *offset*)
mysql_field_tell (*res*)
mysql_free_result (*MYSQL_RES*)
mysql_get_client_info ()
mysql_map_class (*class*, *fields*, *data*)
mysql_num_fields (*res*)
mysql_pretty_print (*file*, *res*)
mysql_row_seek (*result*, *MYSQL_ROWS*)
mysql_row_tell (*res*)
mysql_thread_safe ()
mysql_value_string (*value*)

MYSQL_FIELD

MYSQL_FIELD — information about a field.

Synopsis

```
class MYSQL_FIELD
{
    decimals;
    def;
    flags;
    length;
    max_length;
    name;
    table;
    type;
}
```

MYSQL_OPTIONS

MYSQL_OPTIONS — available options.

Synopsis

```
class MYSQL_OPTIONS
{
    charset_dir;
    charset_name;
    client_flag;
    compress;
    connect_timeout;
    db;
    host;
    init_command;
    my_cnf_file;
    my_cnf_group;
    named_pipe;
    password;
    port;
    ssl_ca;
    ssl_capath;
    ssl_cert;
    ssl_key;
    unix_socket;
    use_ssl;
    user;
}
```

MYSQL_RES

MYSQL_RES — the result of a query.

Synopsis

```
class MYSQL_RES
{
    current_field;
    eof;
    field_count;
    fields;
}
```

Methods

This method is unique to Gamma and is linked to the reference page in this manual.

Data ()

MYSQL_ROWS

MYSQL_ROWS — rows of data.

Synopsis

```
class MYSQL_ROWS
{
}
```

Chapter 4. Methods and Functions Unique to Gamma

MYSQL.add_column

MYSQL.add_column — adds a new column to a table.

Syntax

```
MYSQL.add_column (tablename, column, type, default_val, allow_null)
```

Arguments

tablename

A string naming the table.

column

A string naming the column to add.

type

A string naming the type of the column. Any legal MySQL type can be named here.

default_value

If non-nil, specifies the default value.

allow_null

nil to disallow NULL in this column, or non-nil to allow NULL.

Returns

On success, nil, or an error on failure.

Description

This method of the MYSQL class adds a new column to an existing table. If the default value is non-nil, then it will be used as the default value for the column. If *allow_null* is non-nil, then NULL will be allowed in this column.

This method is defined in the Gamma library /usr/cogent/require/MySQLSupport.g. It is made available to a Gamma program with the statement: require("MySQLSupport");.

Example

This example is taken from Section 2.2, *Tutorial 2: create_pets.g*

```
...

  /* Create the "pets" table, and add columns to it. */
  mysql.create_table("pets");
  mysql.add_column ("pets", "name", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "owner", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "species", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "sex", "CHAR(1)", nil, nil);
  mysql.add_column ("pets", "birth", "DATE", nil, t);
  mysql.add_column ("pets", "death", "DATE", nil, t);
```

...

MYSQL.classes_from_tables

`MYSQL.classes_from_tables` — creates an array of Gamma classes from all tables.

Syntax

```
MYSQL.classes_from_tables (superclass, verbose?=nil)
```

Arguments

`superclass`

The superclass for the newly created class, or `nil`.

`verbose`

If non-`nil`, will cause each table name to be printed as it is processed.

Returns

An array of the classes created, or `nil`.

Description

This method of the `MYSQL` class queries the database represented by `MYSQL` for all of the table names, and creates a class corresponding to each table, with each table's column names mapped as instance variables for the corresponding class. It is possible for some of the entries in the return array to be `nil` if the creation of a particular class corresponding to a table fails.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

MYSQL.class_from_table

`MYSQL.class_from_table` — creates a Gamma class from a table.

Syntax

`MYSQL.class_from_table (`*`symclassname`*`, `*`superclass`*`, `*`tablename`*`)`

Arguments

`symclassname`

A symbol, representing the name of the class to create.

`superclass`

The superclass for the newly created class, or `nil`.

`tablename`

The name of the MySQL table, as a string, to use as the template for this class.

Returns

On success, the new class. Otherwise, `nil`.

Description

This method of the `MYSQL` class creates a new Gamma class whose instance variables correspond to the column names in the table *`tablename`*. If the class named *`symclassname`* already exists, or the table *`tablename`* does not exist in the MySQL database, then this function returns `nil`.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* *`create_pets.g`*

```
...

  /* Test the MYSQL.class_from_table method. */
  mysql.class_from_table (#mypetclass, nil, "pets");
  pretty_princ ("\nCLASS  The .class_from_table method ",
  "returns the mypetclass:\n", mypetclass, "\n");

...
```

The output for this section of the code is as follows:

```
CLASS  The .class_from_table method returns the mypetclass:
(defclass mypetclass nil
[(__columns . #0=[id name owner species sex birth death])
 (__primary_key . id) (__table . #1=pets)]
[birth death id name owner sex species])
```

MYSQL.create_table

`MYSQL.create_table` — creates a new, single-column table.

Syntax

```
MYSQL.create_table (tablename)
```

Arguments

`tablename`

A string to use as the table name.

Returns

On success, `nil`, or an error on failure.

Description

This method of the `MYSQL` class creates a new table with a single column, named "`id`" as an auto-incrementing integer primary key. To add more columns to the table, use the `MYSQL.add_column` method.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2: `create_pets.g`*

```
...

  /* Create the "pets" table, and add columns to it. */
  mysql.create_table("pets");
  mysql.add_column ("pets", "name", "VARCHAR(20)", nil, nil);
  mysql.add_column ("pets", "owner", "VARCHAR(20)", nil, nil);

...
```

MYSQL.delete

`MYSQL.delete` — deletes a row from a table.

Syntax

```
MYSQL.delete (row)
```

Arguments

row

An instance of a class corresponding to a table.

Returns

On success, `nil`, or an error on failure.

Description

This method of the `MYSQL` class attempts to delete a row from the table corresponding to the class of the *row* argument. If the primary key of the row is set, then the deletion is relative to that primary key. If the primary key is not set then this function constructs a delete command to the database specifying as much information as possible based on the values of the instance variables in the row.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* `create_pets.g`

```
...

  /* Test the MYSQL.delete method. */
  remove_row = query[0];
  deleted = mysql.delete(remove_row);
  pretty_princ ("\nDELETE  After deleting this row:\n", remove_row, "\n");
  pretty_princ ("The .delete method returns:\n", deleted, "\n");
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("And a query_to_class on females ",
  "in mypetclass now returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is now: ", query[0].name, "\n");

...
```

The output for this section of the code is as follows:

```
DELETE  After deleting this row:
{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
     (owner . Harold) (sex . f) (species . cat)}
The .delete method returns:
nil
And a query_to_class on females in mypetclass now returns:
[{mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}
 {mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
      (name . Petunia) (owner . Warner Bros.) (sex . f)
      (species . pig)}]
The name of the first pet is now: Chirpy
```

MYSQL.drop_table

`MYSQL.drop_table` — deletes a table from the database.

Syntax

```
MYSQL.drop_table (tablename)
```

Arguments

`tablename`

A string naming the table.

Returns

On success, `nil`, or an error on failure.

Description

This method of the `MYSQL` class causes a table to be deleted from the database. All rows in the table will also be deleted. This method is equivalent to `MYSQL.query("DROP TABLE tablename")`.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2: `create_pets.g`*

```
...

  /* Remove any previously-created table called "pets"  This is essentially
     the same as issuing the command: mysql.query("DROP TABLE IF EXISTS pets");*/
  try mysql.drop_table ("pets"); catch;

  /* Create the "pets" table, and add columns to it. */
  mysql.create_table("pets");

...
```

The try/catch statement allows the processor to continue even if there is an error.

MYSQL.insert

`MYSQL.insert` — inserts a new row into a table.

Syntax

```
MYSQL.insert (row)
```

Arguments

row

An instance of a class corresponding to a table.

Returns

On success, `nil`, or an error on failure.

Description

This method of the `MYSQL` class inserts a new row into the table corresponding to the class of the *row* argument. The value in the primary key of the row is ignored, on the assumption that the primary key is auto-incrementing. The new primary key value will be stored into the *row* instance.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* `create_pets.g`

```
...

  /* Test the MYSQL.insert method. */
  inserted = mysql.insert(newrow);
  pretty_princ ("\nINSERT  After inserting this new row:\n", newrow, "\n");
  pretty_princ ("The .insert method returns:\n", inserted, "\n");
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("And now the .query_to_class method ",
  "applied to females in mypetclass returns:\n", query, "\n");

...
```

The output for this section of the code is as follows:

```
INSERT  After inserting this new row:
{mypetclass (birth . 1925-12-25) (death . N) (id . 9) (name . Petunia)
     (owner . Warner Bros.) (sex . f) (species . pig)}
The .insert method returns:
nil
And now the .query_to_class method applied to females in mypetclass returns:
[{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
      (owner . Harold) (sex . f) (species . cat)}
 {mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}
 {mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
      (name . Petunia) (owner . Warner Bros.) (sex . f)
      (species . pig)}]
```

MYSQL.query_and_store

`MYSQL.query_and_store` — queries a database and stores the results.

Syntax

`MYSQL.query_and_store (`*`query_string`*`)`

Arguments

`query_string`

A string containing a well-formed SQL query.

Returns

An instance of `MYSQL_RES`.

Description

This method of the `MYSQL` class emits a query to the database, and collects the results into an instance of `MYSQL_RES`. This function will throw an error if the query fails.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* `create_pets.g`

```
...

  /* Test the MYSQL.query_and_store method. */
  query = mysql.query_and_store ("select name, birth from pets");
  pretty_princ("\nQUERY  The .query_and_store method applied ",
  "to the 'name' and 'birth' fields in mypetclass ",
        "\nreturns this MYSQL_RES istance:\n", query, "\n");
  query_data = query.Data();
  pretty_princ("Its data, recovered using ",
        "the .Data() method, is:\n", query_data, "\n");

...
```

The output for this section of the code is as follows:

```
QUERY  The .query_and_store method applied to the 'name' and 'birth' fields in mypetclass
returns this MYSQL_RES istance:
{MYSQL_RES (current_field . 0) (eof . 1) (field_count . 2)
    (fields . [{MYSQL_FIELD (decimals . 0) (def) (flags . 1)
       (length . 20) (max_length . 8) (name . name)
       (table . pets) (type . 253)}
        {MYSQL_FIELD (decimals . 0) (def) (flags . 0) (length . 10)
       (max_length . 10) (name . birth) (table . pets) (type . 10)}])}
Its data, recovered using the .Data() method, is:
[[Claws 1994-03-17] [Buffy 1989-05-13] [Fang 1990-08-27]
 [Bowser 1995-08-31] [Chirpy 1998-09-11] [Whistler 1997-12-09]
 [Slim 1996-04-29] [Petunia 1925-12-25]]
```

MYSQL.query_to_class

`MYSQL.query_to_class` — maps query results to class instances.

Syntax

```
MYSQL.query_to_class (klass, query_string)
```

Arguments

`klass`

A class, usually created by a call to `mysql_class_from_table`.

`query_string`

A string containing a well-formed SQL query.

Returns

An array of instances of *`klass`*.

Description

This method of the `MYSQL` class emits a query to the database, and maps the results to instances of the class *`klass`*, with each row in the query result being used to create a single instance. The query should return 0 or more rows whose columns correspond to the instance variables of *`klass`*. This function will throw an error if the class is not defined.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* *`create_pets.g`*

```
...

  /* Test the MYSQL.query_to_class method. */
  query = mysql.query_to_class (mypetclass, "select * from pets where sex = 'f'");
  pretty_princ ("\nQUERY  The .query_to_class method applied ",
  "to females in mypetclass returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is: ", query[0].name, "\n");

...
```

The output for this section of the code is as follows:

```
QUERY  The .query_to_class method applied to females in mypetclass returns:
[{mypetclass (birth . 1993-02-04) (death) (id . 1) (name . Fluffy)
      (owner . Harold) (sex . f) (species . cat)}
 {mypetclass (birth . 1998-09-11) (death) (id . 6) (name . Chirpy)
      (owner . Gwen) (sex . f) (species . bird)}]
The name of the first pet is: Fluffy
```

MYSQL.query_to_temp_class

`MYSQL.query_to_temp_class` — maps query results to instance variables of a temporary class.

Syntax

```
MYSQL.query_to_temp_class (superclass, query_string)
```

Arguments

`superclass`

The superclass for the newly created class, or `nil`.

`query_string`

A string containing a well-formed SQL query.

Returns

An array of instances of a temporary class.

Description

This method of the `MYSQL` class emits a query *`query_string`*, to the database represented by `MYSQL`, collects the results, and constructs a temporary Gamma class whose instance variables correspond to the columns of the resulting query. This function is useful when the query is unique, and when the class that describes the rows does not need to persist once this query has been handled. The resulting class may be named the same as previous temporary classes, and it is possible that instances of more than one temporary class can be in use at the same time. It is not possible to accurately determine the class of an instance resulting from this function by name. To determine the class of an instance, use the Gamma function `class_of`.

The return value is an array of instances, one instance per row produced by the query. These instances are the only instances of their class. It is possible to determine whether two instances were generated from the same query by comparing their classes:

```
class_of(instance1) == class_of(instance2)
```

The *`superclass`* argument specifies whether the temporary class will be derived from an existing class. This can be useful to automatically attach methods to the temporary class through inheritance.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* *`create_pets.g`*

```
...

  /* Test the MYSQL.query_to_temp_class method. */
  query = mysql.query_to_temp_class (mypetclass, "select * from pets where sex = 'm'");
  pretty_princ ("\nQUERY  The .query_to_temp_class method applied ",
  "to males in mypetclass returns:\n", query, "\n");
  pretty_princ ("The name of the first pet is: ", query[0].name, "\n");

...
```

The output for this section of the code is as follows:

```
QUERY  The .query_to_temp_class method applied to males in mypetclass returns:
[{TempQuery (birth . 1994-03-17) (death) (id . 2) (name . Claws)
```

```
    (owner . Gwen) (sex . m) (species . cat)}
 {TempQuery (birth . 1990-08-27) (death) (id . 4) (name . Fang)
    (owner . Benny) (sex . m) (species . dog)}
 {TempQuery (birth . 1995-08-31) (death . 2002-07-29) (id . 5)
    (name . Bowser) (owner . Diane) (sex . m) (species . dog)}
 {TempQuery (birth . 1996-04-29) (death) (id . 8) (name . Slim)
    (owner . Benny) (sex . m) (species . snake)}]
The name of the first pet is: Claws
```

MYSQL.requery

`MYSQL.requery` — updates a row by querying the database.

Syntax

```
MYSQL.requery (row)
```

Arguments

row

An instance of a class corresponding to a table.

Returns

The same row. This function will throw an error on failure.

Description

This method of the MYSQL class issues a query to the MySQL server in an attempt to update the information in the *row* instance with the current data in the database. Normally the primary key for this row should be known in order for the query to succeed. If the primary key is not known, then this function will emit the most precise query that it can with the data available in the row instance. If the resulting query generates any number of rows other than 1, then the function throws an error.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

Example

This example is taken from Section 2.2, *Tutorial 2:* `create_pets.g`

```
...

  /* Test the MYSQL.requery method. */
  requery = mysql.requery(query[1]);
  pretty_princ ("\nREQUERY  The .requery method applied to the ",
  "second class in the above query returns:\n", requery, "\n");

...
```

The output for this section of the code is as follows:

```
REQUERY  The .requery method applied to the second class in the above query returns:
{mypetclass (birth . 1925-12-25) (death . 0000-00-00) (id . 9)
     (name . Petunia) (owner . Warner Bros.) (sex . f)
     (species . pig)}
```

MYSQL.update

`MYSQL.update` — updates the database with the values in a row.

Syntax

```
MYSQL.update (row)
```

Arguments

row

An instance of a class corresponding to a table.

Returns

A `MYSQL_RES` instance. This function will throw an error on failure.

Description

This method of the `MYSQL` class attempts to update the database with the values that are stored in the instance variables of the row. The primary key must be known in order for this function to succeed.

This method is defined in the Gamma library `/usr/cogent/require/MySQLSupport.g`. It is made available to a Gamma program with the statement: `require("MySQLSupport");`.

MYSQL_RES.Data

`MYSQL_RES.Data` — replaces the `MYSQL_RES.data` field.

Syntax

```
MYSQL_RES.Data()
```

Arguments

none

Returns

A two-dimentional array of [*row*][*field_value*].

Description

This method for the MYSQL_RES class replaces the field `MYSQL_RES.data` that would exist in the C API. Gamma needs to construct a two-dimensional array of data where the array is:

```
data[row][field_value]
```

That is, the first dimension of the array is the set of rows, and the second dimension of the array is the field values for each row.

This is a built-in Gamma method that is similar to but does not exactly does not match the MySQL method `MYSQL_RES.Data`.

Example

This example is taken from Section 2.2, *Tutorial 2:* `create_pets.g`

```
...

  /* Test the MYSQL.query_and_store method. */
  query = mysql.query_and_store ("select name, birth from pets");
  pretty_princ("\nQUERY  The .query_and_store method applied ",
  "to the 'name' and 'birth' fields in mypetclass ",
        "\nreturns this MYSQL_RES istance:\n", query, "\n");
  query_data = query.Data();
  pretty_princ("Its data, recovered using ",
        "the .Data() method, is:\n", query_data, "\n");

...
```

The output for this section of the code is as follows:

```
QUERY  The .query_and_store method applied to the 'name' and 'birth' fields in mypetclass
returns this MYSQL_RES istance:
{MYSQL_RES (current_field . 0) (eof . 1) (field_count . 2)
    (fields . [{MYSQL_FIELD (decimals . 0) (def) (flags . 1)
       (length . 20) (max_length . 8) (name . name)
       (table . pets) (type . 253)}
        {MYSQL_FIELD (decimals . 0) (def) (flags . 0) (length . 10)
       (max_length . 10) (name . birth) (table . pets) (type . 10)}])}
Its data, recovered using the .Data() method, is:
[[Claws 1994-03-17] [Buffy 1989-05-13] [Fang 1990-08-27]
 [Bowser 1995-08-31] [Chirpy 1998-09-11] [Whistler 1997-12-09]
 [Slim 1996-04-29] [Petunia 1925-12-25]]
```

mysql_class_from_fields

`mysql_class_from_fields` — creates a class to hold data of future SQL queries.

Syntax

```
mysql_class_from_fields (symclassname, superclass, fields)
```

Arguments

`symclassname`

A symbol, representing the name of the class to create.

`superclass`

The superclass for the newly created class, or `nil`.

`fields`

A one-dimensional array of strings specifying the names of the instance variables of the newly-created class. These names should correspond to the column names of future SQL queries.

Returns

A class named *`symclassname`*.

Description

This function creates a new class named *`symclassname`* as a derived class of *`superclass`*, using the field names in the *`fields`* array as the names of the new class's instance variables. This function's intended purpose is to create a class that can be instantiated to hold the data of future SQL queries.

mysql_pretty_print

`mysql_pretty_print` — prints a table of rows from a query.

Syntax

```
mysql_pretty_print (file, res)
```

Arguments

`file`

An open file pointer, such as `stdout`.

`res`

A `MYSQL_RES` instance.

Returns

`nil`.

Description

This function prints a table of rows resulting from a query to the file pointer *`file`*. The file pointer is the result of a call to the Gamma function `open` or `open_string`, or is one of `stdout` or `stderr`. The *`res`* argument is the result of a MySQL method like `MYSQL.query`.

Example

This example is taken from Section 2.2, *Tutorial 2: `create_pets.g`*

```
...

  /* Test the MYSQL.query_and_store method, and print the whole table. */
  allpets = mysql.query_and_store("SELECT * FROM pets");
  princ("QUERY  The .query_and_store method returns: \n");
  mysql_pretty_print(stdout, allpets);

...
```

The output for this section of the code is as follows:

```
QUERY  The .query_and_store method returns:
id name     owner  species sex birth      death
-- -------- ------ ------- --- ---------- ----------
 1 Fluffy   Harold cat     f   1993-02-04 nil
 2 Claws    Gwen   cat     m   1994-03-17 nil
 3 Buffy    Harold dog         1989-05-13 nil
 4 Fang     Benny  dog     m   1990-08-27 nil
 5 Bowser   Diane  dog     m   1995-08-31 2002-07-29
 6 Chirpy   Gwen   bird    f   1998-09-11 nil
 7 Whistler Gwen   bird        1997-12-09 nil
 8 Slim     Benny  snake   m   1996-04-29 nil
```

mysql_map_class

mysql_map_class — maps fields and data onto an existing Gamma class.

Syntax

```
mysql_map_class(class, fields, data)
```

Arguments

class

An existing Gamma class.

fields

A one-dimensional array of field names that will be used to create instance variables in the class.

data

A two-dimensional array of data as returned by MYSQL_RES.Data.

Returns

A one-dimensional array of instances of class *class*, where the values of the instance variables named in *fields* have the values provided in *data*. The instances in the return array correspond to the rows in data, and are arranged in the same order as the first dimension of the *data* array.

Description

This function maps the *fields* and *data* onto an existing Gamma class, and then constructs a set of instances of that class, one per row in the data. The names of the class instance variables will be the elements of the *fields* array.

This is a built-in Gamma function that is similar to but does not exactly match the MySQL function mysql_map_class.

mysql_value_string

`mysql_value_string` — converts a Gamma value into a string for use in an SQL statement.

Syntax

```
mysql_value_string (value)
```

Arguments

value

Any Gamma expression.

Returns

A string representation of the Gamma value suitable for use in an SQL statement.

Description

This function converts the supplied *value* into a string that is suitable for inclusion in an SQL statement. It will convert a number to its string representation, and will convert a string into that string surrounded by double quotes, with embedded double quotes escaped using backslash characters. Any other data type will be converted to its string representation.

Example

```
Gamma> require("MySQLSupport");
"/usr/cogent/require/MySQLSupport.g"
Gamma> mysql_value_string("Hi there.");
"\"Hi there.\""
Gamma>  mysql_value_string(27);
"27"
Gamma> mysql_value_string(27.2);
"27.19999999999999289"
Gamma> mysql_value_string(date());
"\"Fri Apr 18 11:11:20 2003\""
Gamma> mysql_value_string(#3 + 4);
"(+ 3 4)"
Gamma> mysql_value_string(2003-04-17);
"1982"
Gamma> mysql_value_string(#2003-04-17);
"(- (- 2003 4) 17)"
```

Chapter 5. Functions that Access Global Variables

The following functions let you get and set the value of global variables.

`max_allowed_packet` functions

`max_allowed_packet functions` — get or set the maximum data packet size.

Syntax

```
max_allowed_packet_get ()
max_allowed_packet_set (value)
```

Arguments

`value`

A number.

Returns

The `_get` function returns the current value; the `_set` function returns the newly-set value.

Description

These functions get or set the size of the biggest data packet that the server will allow. This is 16M in MySQL 3.23. See A.2.8 Packet too large Error in the MySQL documentation for more information.

`port` functions

`port functions` — get or set the port number.

Syntax

```
mysql_port_get ()
mysql_port_set (value)
mysql_unix_port_get ()
mysql_unix_port_set (value)
```

Arguments

`value`

A number.

Returns

The `_get` function returns the current value, the `_set` function returns the newly-set value.

Description

These functions get or set the TCP/IP port number to use for the connection to the database. See 4.1.4 Running Multiple MySQL Servers on the Same Machine in the MySQL documentation for related information.

`net_buffer_length` functions

`net_buffer_length functions` — get or set the net buffer length.

Syntax

```
net_buffer_length_get ()
net_buffer_length_set (value)
```

Arguments

`value`

A number.

Returns

The `_get` function returns the current value, the `_set` function returns the newly-set value.

Description

These functions get or set the maximum length of a row for multi-row-insert statements. Should be set to less than 16M. See 4.8.5 mysqldump, Dumping Table Structure and Data in the MySQL documentation, particularly the discussion of the `-O net_buffer_length` option, for more information.

Chapter 6. See MySQL Documentation

6.1. Gamma Functions for MySQL C API Functions

These MySQL functions have been wrapped as functions in Gamma. There is a one-to-one correspondence between the Gamma implementation and MySQL's C API implementation.

```
mysql_debug
mysql_eof
mysql_escape_string
mysql_fetch_field
mysql_fetch_field_direct
mysql_fetch_fields
mysql_field_seek
mysql_field_tell
mysql_free_result
mysql_get_client_info
mysql_init
mysql_num_fields
mysql_row_seek
mysql_row_tell
mysql_thread_safe
```

6.2. Gamma Methods for MySQL C API Functions

These MySQL functions have become methods of the `MYSQL` class in Gamma. Each method corresponds exactly to the same function named `mysql_*` in the MySQL C API and the MySQL documentation.

```
MYSQL.change_user / mysql_change_user
MYSQL.character_set_name / mysql_character_set_name
MYSQL.close / mysql_close
MYSQL.connect / mysql_connect
MYSQL.create_db / mysql_create_db
MYSQL.drop_db / mysql_drop_db
MYSQL.dump_debug_info / mysql_dump_debug_info
MYSQL.errno / mysql_errno
MYSQL.error / mysql_error
MYSQL.field_count / mysql_field_count
MYSQL.get_host_info / mysql_get_host_info
MYSQL.get_proto_info / mysql_get_proto_info
MYSQL.get_server_info / mysql_get_server_info
MYSQL.info / mysql_info
MYSQL.init / mysql_init
MYSQL.insert_id / mysql_insert_id
MYSQL.kill / mysql_kill
MYSQL.list_dbs / mysql_list_dbs
MYSQL.list_fields / mysql_list_fields
MYSQL.list_processes / mysql_list_processes
MYSQL.list_tables / mysql_list_tables
MYSQL.ping / mysql_ping
MYSQL.query / mysql_query
MYSQL.real_connect / mysql_real_connect
MYSQL.real_escape_string / mysql_real_escape_string
MYSQL.real_query / mysql_real_query
MYSQL.select_db / mysql_select_db
MYSQL.shutdown / mysql_shutdown
MYSQL.stat / mysql_stat
MYSQL.store_result / mysql_store_result
MYSQL.thread_id / mysql_thread_id
MYSQL.use_result / mysql_use_result
```

6.3. Other Functions and Methods

The following functions are not documented per se in the MySQL Online Documentation, but some of them are mentioned in context.

`get_tty_password` (*`opt_message`*)

See 4.3.1 GRANT and REVOKE Syntax in the MySQL documentation.

`make_scrambled_password` (*`to`*, *`password`*)

See 4.3.1 GRANT and REVOKE Syntax in the MySQL documentation.

`my_net_init` (*`net`*, *`vio`*)

Not mentioned in the MySQL documentation.

`my_net_read` (*`net`*)

Not mentioned in the MySQL documentation.

`my_net_write` (*`net`*, *`packet`*, *`len`*)

Not mentioned in the MySQL documentation.

`my_thread_init` ()

Not mentioned in the MySQL documentation.

`scramble` (*`to`*, *`message`*, *`password`*, *`old_ver`*)

Not mentioned in the MySQL documentation.

`MYSQL.read_query_result` ()

See 8.1.3.221 mysql_use_result() in the MySQL documentation for a discussion of related topics.

`MYSQL.refresh` (*`refresh_options`*)

See 4.2.7 Privileges Provided by MySQL in the MySQL documentation.

`MYSQL.send_query` (*`q`*, *`length`*)

Not mentioned in the MySQL documentation.

Appendix A. GNU General Public License

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program",

below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE

PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type “show w”. This is free software, and you are welcome to redistribute it under certain conditions; type “show c” for details.

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program “Gnomovision” (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B. GNU Lesser General Public License

GNU Lesser General Public License

Version 2.1, February 1999

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method:

1. we copyright the library, and
2. we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author’s reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Section 0

This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

Section 1

You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.

b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

   (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

Section 4

You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

Section 5

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

Section 6

As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work

during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

Section 7

You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

Section 8

You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who

have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 9

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

Section 10

Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

Section 11

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 12

If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 13

The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

Section 14

If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 15

BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 16

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the library’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library ’Frob’ (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That’s all there is to it!

Index

N

O

P

R

S

T

U

Colophon

This book was produced by Cogent Real-Time Systems, Inc. from a single-source group of SGML files. Gnu Emacs was used to edit the SGML files. The DocBook DTD and related DSSSL stylesheets were used to transform the SGML source into HTML, PDF, and QNX Helpviewer output formats. This processing was accomplished with the help of OpenJade, JadeTeX, Tex, and various scripts and makefiles. Details of the process are described in our book: Preparing Cogent Documentation, which is published on-line at
`http://developers.cogentrts.com/cogent/prepdoc/book1.html`.

Text written by Andrew Thomas and Bob McIlvride.