GNADE User's Guide

GNADE, The GNat Ada Database Environment

Version 1.5.3

Document Revision $Revision: 1.45 $

Edited by

Michael Erdmann

GNADE User's Guide: GNADE, The GNat Ada Database Environment; Version 1.5.3; Document Revision $Revision: 1.45 $

Edited by Michael Erdmann

Revision History

Revision $Revision: 1.45 $ $Date: 2004/05/29 13:01:31 $ Revised by: $Author: merdmann $

Table of Contents

List of Tables

List of Examples

Preface

This document describes the GNADE project application and implementation wise. The document is intended as a living document for developers and users of the GNADE project.

I. Introduction

Chapter 1. Project Objectives

The objective of the GNADE project is to provide an open source environment of tools and libraries in order to integrate SQL into Ada 95. In order to achieve this ODBC and embedded SQL have be selected as platform.

ODBC provides the interface between application code and the underlying dbcs. This interface has been selected because most if the commonly used data bases are providing ODBC.

Embedded SQL (ESQL) provides the framework to integrate SQL queries into the Ada code. ESQL has been selected because there exists a huge amount of legacy code which could be reused. Even ESQL is standardized there are a lot of different implementations around. The ESQL translator in this project tries to merge several ESQL dialects into a single translator.

In long terms the project will provide means to integrate features which are not part of the ISO/92 ESQL specification via extensions of ESQL.

Chapter 2. Software License

The GNU Public License (GPL) applies with the following extension to all software components of this project.

As a special exception, if other files instantiate generics from GNADE Ada units, or you link GNADE Ada units or libraries with other files to produce an executable, these units or libraries do not by itself cause the resulting executable to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the executable file might be covered by the GNU Public License.

Chapter 3. Trademarks

Red Hat™ is a registered trademark of Red Hat, Inc..

Linux™ is a registered trademark of Linus Torvalds.

UNIX™ is a registered trademark of The Open Group.

Alpha™ is a registered trademark of the Digital Equipment Corporation.

Windows™is a registered trademark of the Microsoft Corporation.

Chapter 4. Supported Databases and OS platforms

The table below given an overview about the supported operating systems and databases. For the detailed versions for each product, consult the release notes of the relevant GNADE version.

Table 4-1. Supported Platforms

Linux Redhat 7.0	Postures	Automatically handled by the configure script
Linux Redhat 7.0	MySQL	Automatically handled by the configure script
Linux Redhat 7.0	MimerSQL	Automatically handled by the configure script
SuSe 7.0	Postgres	Automatically handled by the configure script
Windows NT	Postgres	The makefile in ./win32 do not support any automatic configuration and installation of the test database.
Windows NT	Mimer	The makefile in ./win32 do not support any automatic configuration and installation of the test database.
Windows 2000	MySQL	The makefile in ./win32 do not support any automatic configuration and installation of the test database.
Windows 98	Oracle 8i	This is not completly automatically installed.

Chapter 5. Getting started

The GNADE project distribution is currently distributed as development snapshot and installable packages. The development snapshot needs to be recompiled on the target system since it does not contains any binary files. The installable packages are only available for the platforms below:

- Linux

  The following RPM's are available this platform:
  —gnade-X.Y.Z-NNN.src.rpm

  This package contains the source code ready to compile for the linux platform.

  —gnade-X.Y.Z-NNN.i686.rpm

  This package contains precomplied binaries and all required libraries to build ODBC based applications.

- Windows 2000/98/XP

  setup.exe

  The setup for windows provides precompiled executables and libaries for windows and a source tree of the GNADE sources.

The development package contains the sources for all platforms so far supported. GNADE support the two major platforms Windows NT and Unix/Linux. The following sections are describing the installation steps for both platforms.

Chapter 6. Building GNADE on different platforms

Installation of the development snapshot on a Unix like systems

After you obtained the source code from the net you need to install and compile it. This chapter describes this first steps of installing the environment onto your system.

Unpacking the distribution

The source code is normally distributed as compressed tar file. To unpack the distribution execute the command:

```
gunzip -c gnade-src-x.y.z.tar.gz | tar xvf -
```

This will unpack the directory tree of the development environment.

Configuring the GNADE installation

The GNADE environment may be configured to a certain extend. The file make.conf.in contains some parameters which might be adopted to the needs of your system.

After unpacking the distribution change into the top level directory of the GNADE release. Before you run the configure script examine the contents of the file etc/config.site.

```
site_bindir="/usr/bin"
site_libdir="/usr/lib"
site_adadir="/usr/lib/ada"
```

Because system wide installation of GNADE depends on the type of your system please adopt the following parameters in make.conf.in before executing the configure script as shown below.

```
cd gnade-src-....
./configure [ --sampledb=<database> ] [ <native> ]
```

The configure script allowes you to by means of the --sampledb to create a test data base. Additionaly you may give a list of RDBMS producs for which you like to build native bindings. If nothing is given, only the ODBC interfaces relevant part of GNADE will be build. For the supported data base products please check the README file in the gnade directory. If you don't have one of the supported data bases on your system then omit the database. As a result the samples code will be compiled except the code for native bindings, but the sample data base will not be available.

Preparation of the test data base

In order to allow the installation of the test database, most of the commonly known dbcs's require a data base user to be installed. This normally required certain DBA privileges. There for this step is expected to be done manual as shown below (The name of the user, the name of the data base is specified in make.conf.in).

```
su <dbcs root>
gmake createuser
```

The user may be deleted by the command make removeuser.

In order to test the functionality of the data base you may create the test data base already at this point by the following commands:

```
gmake removedb
gmake createdb
```

This will create a database gnade which contains at least the table EMPLOYEES which may be checked manually.

> **Mimer SQL:** In case of Mimer SQL the user is created as root, but the make createdb command has to be executed as the same user which is used to run the test examples. If thus is not done, the examples will fail!

Compiling

To build the GNADE executable enter the command below:

```
gmake all
```

This will build all components of the GNADE project and the test data base is this has not been done previously.

Installing GNADE globally on the system

The development environment is self containing, which means as long as applications are developed in the directory where GNADE is installed and the make files are used, all components are taken from the GNADE lib directory. This method limits the use to one user. In order to make GNADE available to all users on your system you need to install the GNADE libraries. Installation is done as root by executing the directory the following command:

```
gmake install
```

This should install the libraries of the GNADE project in your system.

Integration with IDE Products

The GNADE environment provides limited support for the GPS product of ACT by integrating this document into the GPS help system. In order to integrate with GPS, the environment variable GPS_DOC_PATH has to include the installation directory of GNADE (e.g. /usr/local/gnade/doc).

Building the development snapshot on Windows NT/XP

Unpacking the distribution

The source code is normally distributed as ZIP file, which is easy to unpack by means of Windows utilities as e.g WinZip. From the DOS command line use:

```
unzip gnade-src-arch-version.zip
cd gnade-src-arch-version
```

This will unpack the directory tree of the development environment.

Configuring the release

The configuation is based upon the configuration file make.conf.win32 which is located in the root directory of the destribution. Normally you do not need to change anything in this configuration file. To be sure please review the contents of this file before you contiune. In oder to configure GNADE run the command below in the root directory of the GNADE distribution.

```
configure
```

Compiling the distribution

As for Unix the compilation process is based upon the execution of a Makefile. In order to compile the distribution perform the following commands:

```
gmake
```

Installation on your system

There is no automatic procedure doing so, but after compilation the result will be located in the following directories an can be moved to suitable places in your system.

—win32-bin

contains all executables of GNADE

—win32-lib

Contains all libaries

—win32-include

Contains all Ada specification and body files

—win32-libexec

All executables which are only used by gnade it self

—win32-doc

All documentation is stored here after production. For windows this directory is nt used.

—shared

All shared resources are stored here including the GNADE SDK project file. is nt used.

Chapter 7. Using the pre compiled packaged

Installation of RPM files on Linux Systems

First time installation of the RPM is done by means of the following command:

```
rpm -ivh gnade-X.Y.Z-build-arch.rpm
```

If a version of GNADE has been installed previously deinstall this version before installing the new version.

```
rpm -e gnade-X.Y.Z
```

The installation will provide a variable GNADE_BASEPATH which points to the installation directory of GNADE. The individual user on the system needs to add the GNADE project file to his ADA_PROJECT_PATH if he/she wants to use this project file.

```
ADA_PROJECT_PATH=$ADA_PROJECT_PATH:$GNADE_PATH/shared
```

Since the binaries release does not contain the gnade source tree, this package needs also to be installed if you want to do some changes to the GNADE software it self.This packages is simply installed by executing the command:

```
rpm -ivh gnade-X.Y.Z-build-src.rpm
```

Installation on Windows like systems

Download the windows setup file and execute the setup program in any directory. The installation program will prompt for the place where GNADE is to be installed. Select any place you feel is reasonable. The default is c:\gnade.

Chapter 8. GNADE Addon Packages

Each release of the GNADE project provides several add on packages. These individual packages are candiates for later integration into the GNADE source distribution.

In order to install such packages the GNADE source distribution has to be installed and these packages have to be installed on top of this distribution.

In order to install add on packages change into the base directory of the GNADE project and follow the instructions below:

OCI - Oracle Call Interface

This package provides an API for Oracle.

```
cd dbi
gunzip -c oci-source-version.tar.gz | tar xvf -
cd oci-source-version
make
```

GSQL - Generic SQL Interface

This package provides a small application based upon the ODBC interface. The application allowes to create tables and the queries.

> **Note:** This is still an experimental application. The GUI implementation will be reworked as soon as possible by switching to glade.

To build the software you need GtkAda installed and execute the following commands in the installation directory of GNADE.

```
cd contrib
gunzip -c gsql-source-version.tar.gz | tar xvf -
cd gsql-source-version
gmake
```

Chapter 9. Integration with IDES from other vendors

Using Emacs

The GNADE project uses several file extension which are not commonly defined in the emacs default configuration. As a consequence syntax high ligthing is not activated. In order to activate syntax highlighting again the following lines have to be added to to .emacs file in your home directory:

```
(setq auto-mode-alist
    (append '(("\\.C$"  . c++-mode)
              ("\\.cc$" . c++-mode)
              ("\\.hh$" . c++-mode)
              ("\\.adq$" . ada-mode)
              ("\\.gpq$" . ada-mode)
              ("\\.gpb$" . ada-mode)
              ("\\.c$"  . c-mode)
              ("\\.h$"  . c-mode))
            auto-mode-alist))
```

Using GPS

GNADE integrates with GPS to a certain extend. For more details please refere to the corresponding chapter later in the book.

Chapter 10. Using the release with your database

The GNADE package provides a small test data base for the examples stored under ./samples. The Makefile assumes for each supported data base vendor X a ./samples/X directory where the example code for the native binding is stored.

The ODBC bases examples are using all the same data base stored under ./samples/sample_db. This data base contains the tables EMPLOYEES and DEPARTEMENTS.

If your data base is not supported

If your data base is not supported the test data base has to be installed manually or preferably the data base has to be included in the configuration process which is described in the following.

The DML commands to create the data base are contained in the gnade.postgres.sql file which can be used as a template for the new data base.

The following files have to be created for the new DBCS vendor X.

- Makefile.X

  This Makefile has the targets createuser, removeuser and createdb, removedb.

- gnade.X.sql

  This file contains the DML's for the creation of the data base.

- README.X

  This file contains a data base specific readme which is shown after createdb, removedb.

- removeuser.X, createuser.X

  These files do contains the command required to create a data base user which is allowed to create tables and able to read the gnade data base.

Installation of the ODBC Interface

In order to allow the test programs to connect to the data base via odbc the following entry has to be added either to /etc/odbc.ini or .odbc.ini on Unix systems.

During the process of configuration, templates for the ODBC wise installation of the data base are prepared under ./samples/sample_db as shown below.

Example 10-1. /etc/odbc.ini entry for the test data base

```
[DEMO_DB]
Description             = Demo Database for GNADE
Driver                  = PostgreSQL
Database                = gnade
Servername              = localhost
Port                    = 5432
ReadOnly                = No
RowVersioning           = No
ShowSystemTables        = No
ShowOidColumn           = No
FakeOidIndex            = No
ConnSettings            =
Trace               = Yes
TraceFile           = sql.log
```

Prepared Example Programs

All examples are located in the directories samples and contrib. In the directory samples you will find the following files and directories:

- Makefile

  This is the makefile which builds all examples

- esql

  Several examples demonstrating the features of the esql translator. All example programs are using the test database as it is generated while building the GNADE distribution.

  Table 10-1. Examples of embedded SQL

simple	This demonstrates a simple like SELECT. There are two version demonstrating the capability to process different comilation units.
nobel_co	This examples demonstrates the uage of nested cusrsors and enhanced connection management.
dynamic	A nice example for using dynamic sql with the esql translator.

attachment	This example demonstrates the usage of the GNADE SQL types VARCHAR and VARBINARY. This example allows to store/delete and retrieve the contents of a file from a test database.

- mysql

  Example for the MySQL native bindings

  This example creates it's own data base called "testdb" and issues a query. The query result is printed out.

- odbc

  An example how to use the ODBC interface directly.

  A simple example that executes a query on the gnade test data base.

- postgres

  Example for the Postgres native bindings

  A simple example that executes a query on the gnade test data base.

Chapter 11. Contact

The home page for the project is located at http://gnade.sourceforge.net.

All project activities are maintained at http://sourcefroge.net/projects/gnade

All technical communication regarding the GNADE project is done via a mailing list which is hosted at http://cert.uni-stuttgart.de/mailman/listinfo/gnade-dev (http://cert.uni-stuttgart.de/mailman/listinfo/gnade-dev).

The coordination of the development work is done by:

Michael Erdmann
<`Michael.Erdmann@snafu.de`>

Chapter 12. Authors

These are the authors and copyright holders of the GNADE software (in alphabetical order of their last name):

```
Denis Chalon            denis.chalon@itris.fr
Dimitry Anisimkov       anisimkov@yahoo.com
François Fabien         fr.fabien@infonie.fr
Juergen Pfeifer         juergen.pfeifer@gmx.net
Julio Cano              julius_bip@yahoo.com
Michael Erdmann         michael.erdmann@snafu.de
Stephen Leake           stephen_leake@acm.org
Sune Falck              sunef@hem.passagen.se
```

II. GNU Embedded SQL Translator for Ada 95

Chapter 13. Introduction to Embedded SQL

The GNU Embedded SQL Translator for Ada 95 reads a Ada 95 source file containing an Ada 95 package which contains embedded SQL Commands. A typical code fragment which is embedded into a normal Ada 95 source text is shown below:

Example 13-1. Example for Embedded SQL

```
          EXEC SQL AT DB01x
     SELECT LOCATION INTO :dep_location
        FROM departments
        WHERE DEPTNO = :depno ;

         if SQLCODE not in SQL_STANDARD.NOT_FOUND then
            Put_Line(
               "Employee : " & Trim(To_String(Name),Right)  & -- bug
               " working in dep. " & INT'Image(depno)       & --
               " located at " & Trim(To_String(dep_location),Right) );
         end if;
         ..............
```

Embedded SQL commands are always preceded by the string EXEC SQL. According to ISO/92 all text following until the semicolon forms the query which has to be send to the DBCS. The communication between the SQL query and the application code is done by means of so called host variables. A host variable contains either a parameter as input to a query or the result of a query after the query has been executed by the DBCS. A host variable in an SQL query is marked by a preceding colon (':'). Host variables are declared in a specially marked declare section, where the ISO/92 standard allows only a limited number of data types which may be used for host variables. These data types are defined in the package SQL_STANDARD.

In order to communicate to data bases, ESQL uses in each ESQL statement an optional data base identifier. This identifier is assigned by means of a connect statement to a data base as shown below. First of all is the connection identifier declared to be DB01.

```
       ........
   EXEC SQL DECLARE DB01 DATABASE   ;
       .........
   begin
      EXEC SQL CONNECT "gnade"
                     IDENTIFIED BY "xxxxxxx"
                     BY DB01
                     TO "DEMO_DB" ;
   end;
```

Later, during the initialization of the package, we connect as user "gnade" with the password "xxxxxx" to the database "DEMO_DB". The connection which will be used will be referred as DB01 in all ESQL statements. The name DEMO_DB refers to the data source name in the ODBC setup.

Chapter 14. Embedded SQL Syntax Specification

The GNU Ada 95 Embedded SQL

The ESQL translator is based on the ISO/92 standard for Embedded SQL, but a lot of issues have been left out there. In order to allow comfortable coding several extensions have been added, which have been derived from other popular ESQL dialects for Ada 95. These add ons are not specially marked, because i believe without these extensions it would not possible to implement an application.

Embedded SQL statement

Every embedded SQL Statement has the same general structure shown as below. For each query the programmer may specify the data bases where the query has to be applied. If the data base is not explicitly specified, the default data base connection is assumed.

```
Syntax:

     <embedded SQL statement> ::=
              <SQL prefix>
              statement or declaration
              [ <SQL terminator> ]

     <statement or declaration> ::=
              | <include clause>
              | <database clause>
              | <connect clause>
              | <declare clause>
              | <temporary table declaration>
              | <dynamic sql clause>
              | <query clause>
              | <fetch clause>
              | <embedded SQL declare section>
                <embedded exception declaration>
       ;

     <SQL prefix> ::=
                EXEC SQL [ <DB clause> ] [ <for clause>]

     <SQL terminator>  ::=
                END-EXEC
              | <semicolon>
              | <right paren>
              ;

     <DB clause> ::= AT <name>
```

```
<for clause> ::= FOR <expression> (not yet implemented)

<include clause> ::=
          <include_sqlca_clause>
        | <include_handles>
        ;

<declare clause> ::=
          <declare_db_clause>
        | <declare_table_clause>
        | <declare cursor>
        ;
```

Example 14-1. Embedded SQL Statements

```
EXEC SQL AT db01 select * from employees ;
```

This example sends a query to the db01.

All components of the ESQL statement which are not part of the esql grammar will be copied directly into the query which is to be sent to the dbcs.

SQL Query and FETCH clause

A query may be issued by either defining a cursor or a direct query where only one row is expected. The syntax for the later case is shown below.

```
<query> ::=
      'SELECT' <column list>
          'INTO' <host variable list>
          'WHERE'...... rest of query ......
      ;

<host variable list> ::=
        <variable> [ ['INDICATOR'] <variable>
      | <host variable list>
      | <empty>
      ;

<variable> :: = ':' <identifier> ;
```

The esql handles this statement as a normal SQL statement but removing the 'INTO' clause from the SQL string which is sent to the dbcs. The host variables listed in the <host variable list> are used to store the columns of the query result.

```
<fetch clause;> ::=
```

```
    'FETCH' <source;> 'INTO' <host variable list> ;

<source;> := {
              <empty>
           |  'FROM' <cursor>
           |   'USING' [ 'STATEMENT' ] <statementname>
     }
```

Either a cursor name or a statement name (see dynamic sql) may be given as a source for the fetch command. If the source if left empty, a unnamed cusrsor will be assumed.

> **ATTENTION:** The unnamed cursor has a global nature, which means there is only one unnamed cursor which is reallocated for each query. Any subsequent query with an unnamed cursor will destroy the previously allocated result sets.

Embedded SQL declare section

This section contains all definitions of host variables. Note, that not all data types are allowed for a variable in this section.

```
Syntax:
         <embedded SQL declare section> ::=
                <embedded SQL begin declare>
                  [ <embedded character set declaration>  ]
                  [ <host variable definition> ... ]
                <embedded SQL end declare>

         <embedded character set declaration>  ::=
              SQL NAMES ARE <character set specification>

         <embedded SQL begin declare>  ::=
              <SQL prefix>  BEGIN DECLARE SECTION [ <SQL terminator>  ]

         <embedded SQL end declare>  ::=
              <SQL prefix>  END DECLARE SECTION [ <SQL terminator>  ]

         <host variable definition>  ::=
                <Host Identifiers> ':' <Ada type specification>
                [ ':=' <Ada initial values> ';' ]

         <embedded variable name>  ::=
              ':' <host identifier>

         <host identifier>  ::=
                <Ada host identifier>

         <Host identifiers> ::=
```

```
      <host identifier>
    | <host identifier> ',' <host identifiers>

<Ada type specification>  ::=
      <Ada qualified type specification>
    | <Ada unqualified type specification>

<Ada qualified type specification>  ::=
    'SQL_STANDARD.' <Ada unqualified type specification>

<Ada unqualified type specification>  ::=
      CHAR
         [ CHARACTER SET [ IS ] <character set specification>  ]
         '(' 1..<length>')'
    | BIT '(' 1 .. <length> ')'
    | SMALLINT
    | INT
    | REAL
    | DOUBLE_PRECISION
    | SQLCODE_TYPE
    | SQLSTATE_TYPE
    | INDICATOR_TYPE
    | GNADE.<GNADE specific type>
```

Character set declaration is not supported. If the pedantic option (-pedantic) has been set, a warning will be issued and every thing will be skipped until the next semicolon.

Ada support host variable definitions in the scope of a subprogram. The current implementation of esql does not follow the scope of Ada. This will cause warning, that the type of a host variable is changed.

The translator will issue an Error if the -pedantic has been set if the type is not one of the ones listed above. If the -pedantic switch is not used only a warning is issued. The context clauses regarding the SQL_STANDARD and other packages has to be added to the source by the developer. The translator will add only those packages which are needed to interface with ODBC. The correctness of the identifier will not be checked by the translator except for lexical rules which are needed to parse the code. The Ada compiler has to verify the validity of the identifier.

> **Implementation Note:** The Character set modifier is not supported. It is simply discarded and a warning is issued, that the construct is not supported.

Embedded Exception Declaration

The clause may be used to define the handling of certain conditions after a query. The result of the query is evaluated and the action as defined in the action clause is executed.

```
Syntax:
         <embedded exception declaration>  ::=
```

```
     WHENEVER <condition>  <condition action>

<condition>  ::=
     SQLERROR | NOT FOUND | SQLWARNING

<condition action>  ::=
       CONTINUE
     | <go to>
     | RAISE <host_exception>
     | DO <target>
     | STOP

<go to>  ::=
     { GOTO | GO TO } <goto target>

<goto target>  ::=
      <host label identifier>
```

A defined condition is applies to the next SQL query. By using the switch `-noiso92` the code generator may be forced to apply whenever clause to all embedded SQL statements until the next whenever clause occurs.

Table 14-1. Exception Actions

GOTO	The translator inserts a goto statement to the label specified in the target.
RAISE	The translator inserts a raise statement with an exception as specified in the target. The exception information will contain the line number of the query in the input source file and the package name. Additionally the contents of the message in the SQLCA is added.
DO	The procedure named in the target specification is called.
CONTINUE	This clause will reset all previous actions for the given condition.

Handling of return codes

The following variables will be inserted automatically on package level.

```
package body XXX is

  SQLCODE  : SQL_STANDARD...;
  SQLSTATE : SQL_STANDARD...;
```

These variables will be updated after every query send to the data base. This variable may be used to check the result of a query. The elaboration of the WHENEVER clause is based on these variables as well. Please note, this method is not thread save.

SQL Communication Area

The GNU.DB.ESQL_SUPPORT package provides a so called SQL communication area type. This area contains informations about the result of the last query.

```
Syntax:

     <include_sqlca_clause> ::=
          INCLUDE SQLCA
```

This statement will insert a SQLCA in the Ada 95 code. If this is done in the declare section of a procedure as shown below, the SQLCA will be declared local to the procedure.

Example 14-2. Local SQLCA in procedures

```
procedure Print_Departement(
   departement  : in Integer ) is
   ---
   EXEC SQL BEGIN DECLARE SECTION END-EXEC
   Name                    : CHAR(1..15) := (others=>32);
   .............
   Salary                  : DOUBLE_PRECISION := 0.0;

   EXEC SQL END DECLARE SECTION END-EXEC

   EXEC SQL INCLUDE SQLCA ;    -- Make a private SQLCA

begin

   .....
   EXEC SQL AT DB01
      DECLARE emp_cursor CURSOR FOR
         SELECT EMPNO, FIRSTNAME, NAME, JOB, MANAGER, SALARY
         FROM employees
         WHERE deptno = :Depno  ;
   ....
end;
```

The application may access the contents by using the variable name SQLCA in the application code. This method is preferable in a multi thread environment, because it avoids interferences between threads through the global variables SQLCODE and SQLSTATE.

The SQLCA provides several fields containing usefull information about the most recently executed query as shown below:

```
type SQLCA_Type is record
```

```
   Message       : aliased String(1..255 );
   State         : aliased SQLSTATE_TYPE;
   SqlCode       : aliased SQLCODE_TYPE;
   Affected_Rows : aliased Integer := 0;
end record;
```

The parameter Affected_Rows contains the number of rows affected by the last query.

State and SqlCode do contain the result code of the last query. The SqlCode should not be used any more because the State information contains more information.

The field Message contains a string generated by the underlying dbcs containing information bout the most recent error.

Connection Handling

In order to connect to a data base, the data base identifier to be used has to be defined first. This identifier is a simple name which may be used in the AT clause of an embedded SQL statement and is declared by means of the "declare_db_clause". This clause will insert at the source where the clause is invoked a Ada statement declaring a connection object.

```
Syntax:

      <connect_clause> ::=
              CONNECT [ user ]
                 [ BY <Connection> ]
                 [ TO <db_name>  ]
                 [ AS <name> ]
                 [ IDENTIFIED BY  <password> ]

                 [ ON [COMMUNICATION|ATHORIZATION|OTHER] ERROR
                      [RAISE|GOTO|DO] <target> ]

     <declare_db_clause> ::=
              DECLARE <name> DATABASE
```

As shown in the example below, the declare_db_clause may be used in the argument list of a procedure.

Example 14-3. Using DB connections as procedure arguments

```
procedure Print_Employee(
   His_Number : Integer;
   EXEC SQL DECLARE DB01x DATABASE   ) is
   ---
   ..........
   ---
begin
   empno := INT(His_Number);
```

```
      EXEC SQL WHENEVER NOT FOUND DO Not_Found;

      EXEC SQL AT DB01x
         SELECT NAME, DEPTNO INTO :name, :depno
            FROM employees
            WHERE EMPNO = :empno ;

      .....................
   end Print_Employee;
```

This construct allows to write library packages using data base connections as arguments.

The ’ON’ clause is used to define the handling of errors which may occure during connection. Please note, that the execution of a procedure is straigth forward, which means after the procedure returns the execution continues after the connect statement!

> **Implementation Note:** The data base connection variable inserted by this statement has the name GNADE_DB_<db_name> and is of the type ESQL_Support.CONNECTION_Handle. Such a name should never be used in the application code.

There are situation where you might want to build a general purpose package which contains ESQL statements. In such a situation you may either pass the data base as an argument of each procedure/function of this package as shown below:

```
procedure ...(
   EXEC SQL DECLARE DB DATABASE ) is
begin
   EXEC SQL AT DB ........ ;
end X;
```

If this is to expensive you may implement a common procedure which sets the database globaly.

```
procedure Initialize(
   EXEC SQL DECLARE DB DATABASE ) is
begin
   EXEC SQL AT MYDB
      DATABASE IS DB ;
end Initialize;
```

> **Implementation Note:** Using this approach special you have to ensure that always the correct database is active before calling any operation from your utility package. This approach should be used if you are dealing with only one data base.

Cursor Handling

A cursor is a declares a SQL query with its input and result parameters. The result set is created when the cursor is opened. The syntax for declaring, opening and closing a cursor is shown below.

```
'DECLARE' <name> [ 'REOPENABLE' | 'LOCAL' ] 'CURSOR'
'FOR' <sql query>

'OPEN' <name>
'CLOSE' <name> ['FINAL']
```

LOCAL cursors are only defined within the scope of the block where the nearest DECLARE section is. If the scope is left, the cursor and the associated result set are deleted.

Example 14-4. Local Cursors

As shown in the example below, the cursor emp_cursor will only be valid in the scope of the procedure Print_Departement:

```
  procedure Print_Departement( ..........
          ... departement  : in Integer ) is

  EXEC SQL BEGIN DECLARE SECTION ;
  ....
  Depno                   : INT := INT( Departement );
  ....
  EXEC SQL END DECLARE SECTION ;
begin

  EXEC SQL AT DB01
     DECLARE emp_cursor LOCAL CURSOR FOR
        SELECT EMPNO, FIRSTNAME, NAME, JOB, MANAGER, SALARY
        FROM employees
        WHERE deptno = :Depno ORDER BY EMPNO, NAME;
        .......
```

Normally it is not possible to open the same cursor twice. The type REOPENABLE has been introduced, in order to allow the recursive opening of cursors. This feature may also be emulated by means of recursive procedures with local cursors.

If the cursor type is omitted the cursor and its associated result set to exist only once.

Mixing ODBC and embedded SQLcode

In order to allow mixed use of ODBC and ESQL constructs to access the ODBC handles has been added to the translator. The construct below allows to access either the statement handle or the connection handle of the specified data base name.

```
<include_handle> ::=
```

```
'INCLUDE'
     { 'STATEMENT' 'HANDLE' [ <cursor>] |
       'CONNECTION' 'HANDLE' }
     'OF'  [ <dbname>]
```

In case of the statement handle, the name of the cursor may be specified. If no cursor is given, the statement handle of the last query will be returned.

Example 14-5. Accessing ODBC handles

```
H  : SQLHSTMT;
C  : SQLHDBC ;
...
EXEC SQL AT DB01x
   SELECT LOCATION INTO :dep_location
      FROM DEPARTMENTS
      WHERE DEPTNO = :depno ;
......

-- get the ODBC handles
H := EXEC SQL INCLUDE STATEMENT HANDLE OF DB01x ;
C := EXEC SQL INCLUDE CONNECTION HANDLE OF DB01x ;
```

Dynamic SQL

Currently only the syntax for dynamic SQL is supported. The idea of dynamic SQL is that the application can generate a query by generating a string. This query is executed by the data base and the application may access the result set. This can be achieved by either using the ODBC bindings directly or by using the dynamic SQL constructs as they are provided by the embedded SQL translator.

The name of a statement (<statement_name>) is defined in a DECLARE clause. Each dynamic SQL command is identified by such a name.

As for ODBC, the esql translator provides a prepare and an execute method. With the prepare clause the query is sent to the underlying data base system, but no result set is yet created. This very much comparable with declaring a cursor. After the query has been prepared, the query is executed by means of the execute clause.

```
<dynamic sql clause > ::=
       <prepare clause>
     | <execute clause>
     | <close statement clause>
     ;
```

The prepare clause takes as input the statement name and the query string, which is simply a Ada 95 string variable. Any parameters in the query are marked by means of a '?' character. The host variables of the parameters are listed in the USING clause the the prepare statement.

```
<prepare clause> ::=
      'PREPARE' <statement_name>
         'FROM' { <name> | <string> }
         [ 'USING' <hostvars> ]
      ;
```

The execute clause takes the name of the statement as input for execution. If the USING section in the prepare clause was not included, the parameters of the statement may be assigned latest at this point via the USING clause in this statement.

```
<execute_clause> ::=
      'EXECUTE' <statement_name>
      [ 'USING' <hostvars> ]
      ;
```

The close statement clause is used to close the cursor associated with the statement it sef.

```
<close statement clause > ::=
      'CLOSE' 'STATEMENT' <statement_name>
      ;
```

The result set of the execute is accessed via the FETCH clause as for normal cursors as shown in the following example.

Example 14-6. Using dynamic SQL

```
   EXEC SQL END DECLARE SECTION END-EXEC

   EXEC SQL DECLARE test_sql  STATEMENT ;

   S   : constant String := "SELECT NAME FROM employees WHERE EMPNO = ?";

begin

   EXEC SQL CONNECT $DBUSER
            IDENTIFIED BY $DBPASSWD
            BY DB01
            TO $DBSOURCE ;

   EXEC SQL AT DB01
      PREPARE test_sql
      FROM S
      USING :EMPNO ;

   EMPNO := 5;
```

```
EXEC SQL AT DB01
   EXECUTE test_sql
   USING :NAME :NAME_IND ;

loop
   EXEC SQL AT DB01
      FETCH USING STATEMENT test_sql
      INTO  :name :name_ind  ;

      exit when SQLCODE in SQL_STANDARD.NOT_FOUND;

      Put_Line( "Result " & To_String( name ) );
end loop;

CLOSE STATEMENT test_sql;
```

GNADE Specific Datatypes

The GNADE ESQL translator supports implementation defined data types as e.g. VARCHAR in order to simplify the implementation of Ada 95 applications. The specifications of these types is done in the SQL_STANDARD.GNADE package.

```
<GNADE impl. specific types> ::=
     'VARCHAR ( ' <max> ')',
   | 'VARBINARY ( ' <max> ')'
   ;
```

The type VARCHAR is used to handle strings with variable length. The descriminant in the VHARCHAR type specifies the maximal size of a string.

The application programmer may use the operations Is_Null and Length to figure out if the variable contains data and the length of the data.

An application example is shown below. Additional examples may found in the samples/esql directory.

Example 14-7. Using VARCHAR

```
with Ada.Strings;                use Ada.Strings;
with sql_standard;               use sql_standard;
with gnu.db.esql_Support;        use gnu.db.esql_support;
use  gnu.db;

procedure Test is

      val         :   String := "FIRSTNAME";
```

```
      -- declare host and program variables
      EXEC SQL BEGIN DECLARE SECTION;

      ENAME        :   GNADE.VARCHAR(50);
      EMPNO        :   sql_standard.int;

      SQLCODE      :   sql_standard.sqlcode_type; -- for ANSI mode
      SQLSTATE     :   sql_standard.sqlstate_type; -- ANSI mode

      tt           :   GNADE.VARCHAR ( 50 );

      EXEC SQL END DECLARE SECTION;

      SQL_ERROR    :   exception;
      SQL_WARNING :    exception;

begin
      EXEC SQL CONNECT $DBUSER
               IDENTIFIED BY $DBPASSWD
               TO $DBSOURCE ;

      To_VARCHAR( "Michael", tt );


      EXEC SQL
            SELECT  empno, name
            INTO :EMPNO, :ENAME
            FROM employees
            WHERE FIRSTNAME = :tt
      ;

      Put_Line( "empno : " & Integer'Image(Integer(empno)) );
      Put_Line( "found name : " & To_String( ename ) );

end Test;
```

Chapter 15. The ESQL Translator

Compilation Process

A ESQL module is either a package or a file containing only a single compilation unit (procedure). The file containing the Ada 95 code is read in by the translator which translates all ESQL statements into Ada 95 statements.

The name of the output file is generated by replacing the extension of the file name with ".adb". Any extension may be used, but by convention the extension ".adq" is used.

If you are using make, add the following lines to your makefile and process works automatically.

```
.SUFFIXES: .adb .adq

ESQL=esql

.adq.adb:
        $(ESQL) $(ESQLFLAGS) $*.adq
```

The resulting adb file has to be compiled as it is well known using the GNAT.

> **Implementation Note:** The generated code is based on a support package, which is used to interface with ODBC. All object names generated by the translator begin with the string GNADE_. It is strongly recommended to avoid such names in the application code in order to avoid an conflicts.

Invocation of the GNU ESQL Translator (gesql)

gesql [-pedantic] [-debugcode] [-iso92] [-nosqlstate] [-limit number] [-schema file] [-debug] [-v] [-s] [-h, --help] [-gnatnosref] [-Dname=value] `file`...

The command translates embedded SQL statement into Ada 95 for the give input file(s) and writes out for each input file an Ada 95 output file by replacing the extension of the input file by ".adb".

Table 15-1. Options

-pedantic	The translator will complain about non ISO/92 constructs, even if they are supported. Default is off.
-debugcode	If this switch is set, debug code is inserted after each query. Default is off.
-iso92	If set, a whenever clause is always active till the next whenever clause. The default is off.

-nosqlstate	If set, the SQLSTATE and SQLCODE variable is not inserted automatically any more. This switch might be used to minimize the porting effort for PRO*Ada™ code.
-limit number	Set the maximum number of error before the translator terminates.
-debug	If this switch is set, the esql translator outputs debugging information. This output should be sent in with bug reports. Default is off.
-s	No copyright messages are printed at all.
-v	Verbose mode
-schema file	If the embedded SQL code contains declare table clauses, the table declaration is mapped into a SQL create table command. This switch is valid for all files compiled after wards.
-connectpkg name	This switch includes the named package into the expanded Ada 95 source code. This might be used if you like to extend the connection type.
-gnatnosref	Suppress the insertion of the GNAT pragma Source_Reference. This switch is usefull if you are using some kind of preprocessor on embedded SQL files which are already including this pragma (e.g gnatprep -s).
-Dname=value	Substitute all occurences of $name in the source file by the given value. This might saves a gnatprep run.

Chapter 16. Building Applications using ESQL

Using ESQL with make

The program listing below shows a minimal Makefile which compile the source file nobel.gpq with embessed SQL code into a program with the name nobel_co (see samples/standalone).

Example 16-1. A minimal Makefile

```
ESQLFLAGS = $(DEBUG) -v -s -pedantic $(FLAGS)

# use always the local esql translator
ESQL=gesql

DBUSER=gnade
DBPASSWD=gnade
DBSOURCE=gnade

DBAUTH=  -DDBUSER=\"$(DBUSER)\"   \
  -DDBPASSWD=\"$(DBPASSWD)\"   \
  -DDBSOURCE=\"$(DBSOURCE)\"
##
## New compiler rule
##
.SUFFIXES: .adb .gpq

.gpq.adb:
 $(ESQL) $(ESQLFLAGS) $(DBAUTH) $*.gpq

all :: nobel_co

##
## nobel_co
##
nobel.adb: nobel.gpq

nobel_co: nobel_co.adb nobel.adb nobel.ads
 gnatmake nobel_co -o nobel_co ‘gnade-config --libs‘

###
clean:
 rm -rf $(PROGRAMS) b~*.* *.o *.ali core *~ *~*~ bb.out
 rm -rf nobel.adb
```

Additional command Line Interface Options

An application build with the ESQL translator will support the following command line options.

application [--esql-warnings] [--esql-errors]

Table 16-1. Options

--esql-warnings	If this flag is set, warnigns generated by the esql support package will be printed into standard out. This option is only valid if the Warning procedure has not been overloaded.
--esql-errors	Errors issued by the esql support package will be printed into standard out. This option id only valid if the Error procedure has not been overloaded.

Redirecting ESQL Support output

Since the ESQL support package creates some diagnostic printouts it may be interesting to redirect these messages towards any destination. This might be done by overloading the type Connection_Type of the GNU.DB.ESQL_Support.ODBC package and providing implementations of the Error, Warning and Connect procedures.

Example 16-2. Redirecting ESQL diagnostics

```
package Dynamic_Connect is

   type My_Connect_Type is new ODBC.Connection_Type with record
         .... your extensions .....
      end record;

   procedure Error(
      C : in My_Connect_Type;
      T : in String );

   procedure Warning(
      C : in My_Connect_Type;
      T : in String );

   function Connect(
      Source     : in String;
      UserName   : in String;
      Password   : in String  ) return ODBC.Connection_Handle;

end Dynamic_Connect;
```

The implementation might be found in samples/esql.

III. ODBC bindings for Ada 95

Chapter 17. Introduction to ODBC

The ODBC interface provides an interface between applications and an underlying data base in such a way, that the application code does not depend on the underlying data base.

The ODBC interface consists of a so called driver manager and the ODBC driver it self. The driver manager (DM) is a library that on one site offers the specified ODBC API to applications. The DM therefore is what you essentially link to your application. But in large parts the DM routines are only stubs. At run time the DM decides which database to access and based on the type of the database which vendors database ODBC driver to load. So basically most DM implementations require that the OS supports dynamic linking and that the database vendors provide the database site of the ODBC drivers as dynamic loadable entity (aka DLL or shared libraries). But the DM does more than just to provide these stubs and the dynamic linking of the corresponding implementations. As ODBC evolves over time, the DM is also responsible to handle the situation that with a new version of ODBC new API entries are defined, but they are not available in a database driver because this driver was developed when an earlier version of ODBC was the rule (for example we now have ODBC 3.52 and the MySQL ODBC driver is written for ODBC 2.5x). So an application might link against an ODBC 3.52 DM and use all the new and hot ODBC entries, although the database used doesn't have them in its ODBC driver. The DM usually reacts in one of two ways:

- it raises an error indicating an unsupported call.
- it emulates the new call by translating it to a previous (maybe deprecated) call or series of calls. Funny enough this happens quite often and the way how to emulate a new call by existing ones is in most cases exactly described in the ODBC spec.

The mechanism how to select the right driver is system dependent, but the principal idea is that you have some kind of repository where you associate logical names with configuration information telling the DM the specifics which driver to load. On Win32 this repository can be the registry or so called DSN-files, on UNIX this is mostly an ODBC.INI file containing the information in some structured fashion. The application opens the database by specifying such a logical name and its the task of the DM to consult the repository and to dynamically load the right database driver. In this way, a carefully written application can not only be written in a database independent fashion (using the ODBC API), but also the resulting binary can be dynamically configured to use different databases. This is what makes ODBC so successful on Win32 and will make it more and more important also on UNICes. You can write very generic data aware code ranging from applications like MS Access that can operate on any database that supports ODBC, to GUI widgets like data grids that you can incorporate into your GUI application and that binds "magically" to nearly any database you want.

The database ODBC driver is typically a sharable object that implements the ODBC interface on the database site and is loaded by the DM. In theory - although quite uncommon - you may link such a driver directly to your application. This will work if your application makes only ODBC calls that are implemented by the ODBC version used when writing the database driver. Your application then is written in a database independent fashion, but the binary is bound to a specific database.

Chapter 18. Using the Ada 95 ODBC Bindings

General remarks

The ODBC binding for Ada 95 presented in this project is a thin binding to the ODBC interface following the naming conventions of ODBC which means most of the commonly available code examples may be applied to Ada 95 only with minor changes due to the fact, that C and Ada 95 are completely different languages.

Therefore we will not describe the ODBC API here in detail. Please read the original documentation from Microsoft or any other source you can find. We will discuss here only the binding specific aspects.

A minimal odbc example

A code fragments of minimal ODBC program are shown below. The code fragment consists of three basic sections, the initialization code, the connections to the data base and the query it self (the source code is found in the samples/odbc directory).

Example 18-1. Preparing data of the ODBC driver

```
SQLAllocHandle (SQL_HANDLE_ENV, SQL_NULL_HANDLE, EnvironmentHandle);
SQLSetEnvAttr  (EnvironmentHandle,  Environment_Attribute_ODBC_Version’
               (Attribute => SQL_ATTR_ODBC_VERSION,
                Value     => SQL_OV_ODBC3));
SQLAllocHandle (SQL_HANDLE_DBC, EnvironmentHandle, ConnectionHandle);
```

This section connects to the data base. In this case named by the name "gnade" with the password "gnade".

Example 18-2. Connecting to the data base via ODBC

```
SQLConnect (ConnectionHandle => ConnectionHandle,
           ServerName       => "DEMO_DB",
           UserName         => "gnade",
           Authentication   => "gnade");
```

After the connection has been established, the query has to be done. Let us assume a query like:

```
SELECT name, firstname
FROM employees
WHERE manager = :name;
```

Assuming this query, the query will be sent to the dbcs by means of the SQLPrepare method. This will not create any result set, but it binds the command to the previously allocated statement handle.

Example 18-3. Preparing the Query via ODBC

```
declare
   ......
   Name, Firstname : aliased Name_String;
   Len_Firstname, Len_Name : aliased SQLINTEGER;
begin
   SQLAllocHandle (SQL_HANDLE_STMT, ConnectionHandle, StatementHandle);
   SQLPrepare (StatementHandle,
               "SELECT " & QuoteIdentifier ("name") & ", " &
               QuoteIdentifier ("firstname") &
               " FROM " & QuoteIdentifier ("employees") & " " &
               "WHERE " & QuoteIdentifier ("manager") & " = ? " &
               "ORDER BY " & QuoteIdentifier ("name") & "," &
               QuoteIdentifier ("firstname"));
```

Example 18-4. Using host variable with ODBC

The host variable :name is substituted by a '?' sign in the query and the Ada 95 variable "Search_Manager".

The columns name and first name of the query are bound the the Ada 95 host variable Name and Firstname.

```
MB.SQLBindParameter (StatementHandle, 1, SQL_PARAM_INPUT,
                    SQL_C_SLONG, SQL_INTEGER, 0,
                    0, Search_Manager'Access,
                    0, Len'Access);

SB.SQLBindCol (StatementHandle, 1, SQL_C_CHAR,
              Name'Access, Name'Length, Len_Name'Access);
SB.SQLBindCol (StatementHandle, 2, SQL_C_CHAR,
              Firstname'Access, Firstname'Length,
              Len_Firstname'Access);
```

Example 18-5. Creating the result set for a query

Finally the result set is created by executing the query at the data base.

```
SQLExecute (StatementHandle);
```

Example 18-6. Fetching data of the result set via ODBC

The following section reads in one result tuple after the other by means of the SQLFetch method. The result is stored in the host variable which have been specified in the SQLBindCol methods in the previous steps.

```
      declare
         EndFlag         : Boolean := False;
      begin
         loop
           exit when EndFlag;
           SQLFetch (StatementHandle);
           SQLFixNTS (String (Name), Len_Name);
           SQLFixNTS (String (Firstname), Len_Firstname);
           Put (String (Name (1 .. Integer (Len_Name))));
           Put (", ");
           Put (String (Firstname (1 .. Integer (Len_Firstname))));
           New_Line;

         end loop;
      exception
         when No_Data => EndFlag := True;
      end;
   end;
```

After the result set has been processed, the we disconnect from the data base and return all held resources to the odbc driver.

```
   SQLCommit (ConnectionHandle);
   SQLDisconnect (ConnectionHandle);

   SQLFreeHandle (SQL_HANDLE_DBC, ConnectionHandle);
   SQLFreeHandle (SQL_HANDLE_ENV, EnvironmentHandle);
```

Implemented ODBC methods

The methods exported by the odbc packages do follow the same naming conventions as the ODBC standard. The methods listed below are implemented in this release.

```
SQLAllocHandle
SQLBindCol
SQLBindParameter
SQLCancel
SQLCloseCursor
SQLColumns
SQLConnect
SQLCopyDesc
SQLDescribeCol
SQLDisconnect
```

```
SQLEndTran
SQL_Error_Message
SQLExecDirect
SQLExecute
SQLFetch
SQLFetchScroll
SQLFreeHandle
SQLFreeStmt
SQLGetConnectAttr
SQLGetCursorName
SQLGetData
SQLGetDiagField
SQLGetDiagRec
SQLGetEnvAttr
SQLGetFunctions
SQLGetInfo
SQLGetStmtAttr
SQLGetTypeInfo
SQL_LEN_BINARY_ATTR
SQL_LEN_DATA_AT_EXEC
SQLNativeSql
SQLNumParams
SQLNumResultCols
SQLParamData
SQLPrepare
SQLPutData
SQLRowCount
SQLSetEnvAttr
SQLSpecialColumns
SQLStatistics
SQLTables
```

Chapter 19. Building ODBC based programs

The root package of the ODBC binding is GNU.DB.SQLCLI. We've chosen the name SQLCLI to indicate that our main focus is to implement at least the Command Level Interface (CLI) of SQL/92. ODBC is an enhanced implementation of CLI.

Depending on your platform you must add the path to the package sources and the compiled files to you ADA_INCLUDE_PATH and ADA_OBJECTS_PATH. If you're using a platform that supports shared libraries, the libadaodbc.so file should be in a directory searched by your dynamic linker automatically or you must add the directory containing this file to your LD_LIBRARY_PATH.

The ODBC binding references the calls offered by an ODBC driver manager. The GNADE project doesn't implement its own driver manager, but it relies on the one you are using on your system. Please consult your system documentation to find the name of the library that implements the driver manager.

On Linux we suggest to use the unixODBC driver manager (http://www.unixodbc.org). If you use this one, you have to add "-largs -lodbc" to your gnatmake arguments if you want to compile an ODBC program.

Chapter 20. Ada95 aspects of the ODBC binding

The ODBC API typically maintains a set of resources on behalf of the calling application, such as an ODBC Environment, Connections, Statements etc. All those resources have attributes that can be set or get by an application. These attributes have different data types.

As a rather low level API ODBC is oriented to wards low level languages like C. For the above mentioned access to the attributes of various resources the API implements calls in such a way that you have to specify a pointer to a chunk of memory and a parameter containing the length of this area in bytes and then the API fills the area of memory with data or reads data from the area. It's up to the caller to make sure that the so described memory area contains valid data of a type expected by the call. A "C" language prototype of a typical call of this category looks like this:

```
SQLRETURN SQLGetConnectAttr(
   SQLHDBC ConnectionHandle,
   SQLINTEGER Attribute,
   SQLPOINTER Value,
   SQLINTEGER BufferLength,
   SQLINTEGER *StringLength);

SQLRETURN SQLSetConnectAttr(
   SQLHDBC ConnectionHandle,
   SQLINTEGER Attribute,
   SQLPOINTER Value,
   SQLINTEGER StringLength);
```

The parameter "Attribute" is actually an enumeration. An integer number denotes the attribute you're interested in. Different attributes have different data types and there is no rule for the mapping of attributes to their type. You have to read the documentation!

We think this is not the level of type safety we should provide to Ada95 clients of this API. We therefore implemented the following scheme to deal with this mapping problem. We will not describe the internals of this scheme here, but how to use it in your application.

The core of the mapping mechanism is the generic package GNU.DB.SQLCLI.Dispatch which you never will instantiate directly. Lets for example take the connection attributes of the ODBC API to demonstrate the use. You'll find the connection attribute handling in the package GNU.DB.SQLCLI.Connection_Attribute. What you find there is an enumeration type named SQL_CONNECTION_ATTRIBUTE. This type represents the plain SQLINTEGER parameter of the above mentioned C API call. In this package you'll find these instantiations:

```
   package Connection_Attributes is
     new GNU.DB.SQLCLI.Generic_Attr (Context   => SQLHDBC,
                                T               => SQL_CONNECTION_ATTRIBUTE,
                                Base            => SQLINTEGER,
                                Get             => Get_Connect_Attr,
                                Set             => Set_Connect_Attr,
                                Default_Context => Null_Handle);
```

```
subtype Connection_Attribute is
   Connection_Attributes.Attribute_Value_Pair;

package Dispatch is new GNU.DB.SQLCLI (Connection_Attribute);
```

The generic package GNU.DB.SQLCLI.Generic_Attr defines an abstract tagged type Attribute_Value_Pair. This type has a single component: "Attribute", which is of the enumeration type to be mapped (formal parameter T in the above instantiation). There exist derived types from this abstract type for the various data types that are possible as attributes (bitmap, boolean, boolean_string, context, enumerated, integer, pointer, string, unsigned). All these derived types add one additional component to the abstract base type: "Value" whose type is selected according to the needs of the attribute to be mapped.

The dispatch package has the instantiation of the generic as parameter and does set up internally all mappings necessary to return a correctly typed Attribute_Value_Pair'Class for an attribute enumeration value. The C API calls now translate into these Ada95 calls:

```
function SQLGetConnectAttr
  (ConnectionHandle : SQLHDBC;
   Attribute        : SQL_CONNECTION_ATTRIBUTE;
   MaxLength        : SQLSMALLINT := SQL_MAX_OPTION_STRING_LENGTH)
  return Connection_Attribute'Class;

procedure SQLSetConnectAttr
  (ConnectionHandle : in SQLHDBC;
   AttrRec          : in Connection_Attribute'Class);
```

If you look into the package GNU.DB.SQLCLI.Connection_Attribute you for example find there this definition

```
type ACCESS_MODE is (SQL_MODE_READ_WRITE,
                     SQL_MODE_READ_ONLY);
for ACCESS_MODE'Size use SQLINTEGER'Size;

SQL_MODE_DEFAULT : constant ACCESS_MODE := SQL_MODE_READ_WRITE;

package Dsp_Access_Mode is new
  Dispatch.A_Enumerated (SQL_ATTR_ACCESS_MODE,
                         ACCESS_MODE,
                         SQLINTEGER,
                         "ACCESS_MODE");
subtype Connection_Attribute_Mode is Dsp_Access_Mode.Info;
```

From this you can see that the connection attribute SQL_ATTR_ACCESS_MODE is mapped to an enumerated type ACCESS_MODE. So a call to set the access mode looks like this:

```
SQLSetConnectAttr (connHandle,
                   Connection_Attribute_Mode'(
                      Attribute => SQL_ATTR_ACCESS_MODE,
                      Value     => SQL_MODE_READ_ONLY)
                  );
```

and a call to get the attribute may look like this:

```
declare
   attr : Connection_Attribute_Mode;
begin
   attr := Connection_Attribute_Mode(
              SQLGetConnectAttr (connHandle, SQL_ATTR_ACCESS_MODE)
           );
end;
```

Note that the type conversion is required to do the dynamic type check of the function return which returns a Connection_Attribute'Class value.

You'll find this technique in these packages:

- GNU.DB.SQLCLI.Info
- GNU.DB.SQLCLI.Connection_Attribute
- GNU.DB.SQLCLI.Statement_Attribute
- GNU.DB.SQLCLI.Environment_Attribute

Due to the dynamic type checking implemented for the attribute handling, all calls dealing with attributes will cost some more cycles than a direct call to the plain C API. All other ODBC calls are a very thin layer around the C API. As attribute set/get calls are rare compared to queries etc. this is acceptable. But it explains while a - in theory - thin binding is compiled into a rather huge library. This is because all the type mapping information is compiled into the library.

IV. Native Bindings

Chapter 21. Introduction to native bindings

The GNADE project supplies bindings to client libraries of commonly known data bases systems. The intention of bindings is often to provide an API which reflects special features of a data base. There for there is currently no unified interface for all data base bindings available.

Chapter 22. MySQL bindings

The following example is stored under ./samples/mysql. It requires the installed client libraries of the MySQL product.

The MYSQL API

An instance of the MySQL.Object type represents the data base on application level. All operations on the data base are performed on this data type.

Each issued query is identified by a query id which issued to refer to the query.

Every program has to connect and authorize at the data base. This is done by the Methods User, Password and Connect as shown in the example below.

The data base is selected by the primitive Select_DB.

Example 22-1. MySQL native binding - Connecting to the database.

```
with GNU.DB.MySQL;           use GNU.DB.MySQL;
with GNU.DB;                 use GNU.DB ;


   dBase      : MySQL.Object;
   qId        : MySQL.Query_ID;

begin
   Initialize( dBase );

   User(     dBase, "gnade" );
   Password( dBase, "" );
   Connect(  dBase, "localhost" );

   Select_DB( dBase, "testdb" );
```

A query is send by the method Query to the data base. The query string is a normal SQL query or DML command. The result set of a query is described by a so called query identifier with the type Query_ID. The result set is generated at the time where the Query method is executed.

Example 22-2. MySQL native binding - Executing a query

```
   .....
   qID := Query( dBase, "select * from Test where id='Otto'");
   Put_Line( "Nbr of Rows:" & Integer'Image(Nbr_of_Rows(dBase, qID)) );
   ....
```

The result set may be reed out by means of the Next method as it is shown below.

Example 22-3. MySQL native binding - Accessing the result set

```
........
while true loop
   declare
      Insert_Time : Time;
   begin
      Nbr_Tuples := Nbr_Tuples + 1;
      Put_Line( "'" & To_String( String_Field( dBase, qId, "id" ) ) & "'" );
      Insert_Time := Date_Field( dBase, qID, 2);
      Next( dBase, qID );
   exception
      when Field_Parse_Error =>
         Put_Line("Field parse error");
         Next( dBase, qID );
      when Others =>
         raise;
   end;
end loop;
.........
```

After the result set has been processed the query context has to be returned to MySQL via the Drop_Query method.

If the application intends to disconnect completely, the data base instance should be Finalized as shown below.

Example 22-4. MySQL native binding - Dropping the query

```
.....
Drop_Query( dBase, qID );

Finalize( dBase );
.....
```

NOTE: All data types are allocated a part of the result set. The only known deviation are blobs. The function Get_Blob_Field returns a dynamically allocated by array which need the be deallocated by the application explicitly by means for the Free procedure. If this is not done your application will loose memory.

Building programs with MySQL

The MySQL API stored in the library adamysql uses the client library of MySQL, which means the following linker options have to be passed to gnatmake:

```
gnatmake .... -largs .. L/usr/lib/mysql -ladamysql -lmysqlclient
```

A sample makefile is stored under ./samples/mysql.

Chapter 23. Postgres bindings

V. ADBC - Ada Database Connectivity

Chapter 24. Introduction

The intention of the ADBC packages is to provide an data base API which allowes to implement application independent of the used data base system.

This interface has to be understood as an experimental interface which is under permanent discussion in the Ada 95 community. The discussion will be done via comp.lang.ada and the gnade development list. The GNADE package will contain only the stable versions of the ADBC interface. All intermediate versions will be released as seperate packages.

The interface does not rely on any other components of the GNADE project and can be used stand alone.

Chapter 25. Basic Concepts

This chpater gives an overview of the concepts of the ADBC interface.

The ADBC interface is based upon the following major components:

i. Hostvariables
ii. Statements
iii. Resultsets

Hostvariables

Host variables are used to transfer data between the data base and the application. In a query the hostvariables are indicated by a name preceeded by an ':' in a query. The API allowes to associcate Ada 95 host variables with these names. An Ada 95 hostvariable provides the value and the additional attribute Is_Null to the application programmer, which means the variable is has a value or is not set.

> **Driver dependencies:** Not all database drivers do provide the null indication for a colon in the result set. As a consequence the driver may not be able to support this. The driver should always return true and a defined value.

Statements

Statements are SQL command string where the hostvariables are marked by a ':' character. In the example belowm the variable myname may be used is an input parameter for the query

```
select * from employees where name = :myname
```

Resultset

Chapter 26. Using the ADO Interface

A Sample Application

For the communication with the underlying database the application uses a so called connection object. The connection object uses a so called driver handle in order to communication with the database specific driver. The code fragment below shows how to declare a connection to a MySQL data base.

```
procedure Client is
   .............
   X  : Driver.Handle := Driver.MySQL.Create;
   C  : Connection.Object(X);
   .............
```

In order to connect to the data base, the Connect method has to be invoked as shown below, where the parameters do depend on the actual data base used.

```
begin
  ...
  Connect(C, "gnade", Password => "", Database => "gnade" );
```

After the connection has been established statements for execution may be prepared. In order to do so, the method prepare has to be invoked as shown below.

```
   -- prepare a query
   S := Prepare( C,
        "select empno, name, firstname from EMPLOYEES " &
        "where empno > :emp and name = :id ;");

   Bind(S, "id", V );
   Bind(S, "emp", E );
```

This method analysis the given statements for so called hostvariables and invokes the underlying data base in order to evaluate an execution plan.

The following Bind commands connect the hostvariables V and E with the host variable names "id" and "emp". The variables V,E are input parameters for the query.

In order to execute the query, the following code fragment is used:

```
   -- execute a query
   Value( V, "Bundy" );
   Value( E, 50 );
   R := Execute(C, S );
```

This fragment sets the values of the two hostviariables V and E and executes the statement denoted by S on the connection C. The result of this execution is the result set denoted by the variable R.

A result set denotes all all records which are generated by the execution of an SQL query statement.

```
-- retrieve the result
declare
   package D_Empname  is
      new String_Domain( Size => Name_Type'Length, Result => R, Name => "name");
   package D_Firstname  is
      new String_Domain( Size => Name_Type'Length, Result => R, Name => "firstna
   package D_Empno is
      new Integer_Domain( Number_Type => Empno_Type, Result => R, Name => "empno
begin
   while not End_Of_Result(R) loop
      Fetch( R );

      Put_Line(Integer'Image(D_Empno.Value) & " " &
         D_Empname.Value &
         D_Firstname.Value );
   end loop;
end ;
Deallocate( R );
```

Building a Sample Application

Things to know

VI. OCI - Oracle Call Interface

Chapter 27. Introduction

Chapter 28. Basic Concepts

Hostvariables

Statements

```
select * from employees where name = :myname
```

Resultset

Chapter 29. Using the OCI packages

A Sample Application

```
select * from employees where name = :myname
```

Building a Sample Application

The Ada95 Interface to Oracle® RDBMS~Y is depend only from Oracle Call Interface library (OCI). The applications using the OCI should be linked with a static or dynamic OCI libraries provided by Oracle corporation. The descriptions of linking for the different UNIX platforms is in the "Oracle Administrator's Reference" chapters "Oracle Call Interface" and "Oracle Precompiler and Oracle Call Interface Linking and Makefiles" There are written that for the UNIX platforms you should do linking like the $ORACLE_HOME/rdbms/demo/demo_rdbms.mk does.

Things to know

VII. GNADE command line tools

Chapter 30. Introduction

The GNADE project provides some elementary command line tools on top of the ODBC interface.

Chapter 31. lsodbc - Display data source information

lsodbc [-u login] [-l password] [-l] *source*...

Display all available information about all or a certain odbc source.

Table 31-1. Options

-u login [-l password]	The login name used for the database associated with the source name. This makes only sense if the contents of a certain data source has to be checked, since the login/passwd is only sent once and it will be applied to all given data sources, which is proably not correct.
-l	Show longest format of the available information.

Chapter 32. sql - execute an sql command

sql [-u login] [-l password] [-q] [-h] `source` sql query

Execute an sql command on the given data source. The result of the query will be printed into stdout. If the -q switch is not given a copyright notice and a header line will be printed out.

Table 32-1. Options

-u login [-l password]	The login name used for the database associated with the source name. This makes only sense if the contents of a certain data source has to be checked, since the login/passwd is only sent once and it will be applied to all given data sources, which is proably not correct.
-q	Run quiet by not printing out the copyright and other notices.
-h	Print out a help message.

Chapter 33. exp - export a table from a data base

exp [-u login] [-l password] [-q] [-h] source table [-f filename]

Export the named table located in the given data source into a file. The named file is intended for the use by the imp (import) tool.

Table 33-1. Options

-u login [-l password]	The login name used for the database associated with the source name. This makes only sense if the contents of a certain data source has to be checked, since the login/passwd is only sent once and it will be applied to all given data sources, which is proably not correct.
-f filename	Specifies the name of the export file. If not given the name tablename.exp will be used. This option is only allowed after a table name has been given.
-q	Run quiet by not printing out the copyright and other notices.
-h	Print out a help message.

Chapter 34. imp - import data into a table from export files

imp [-u login] [-l password] [-q] [-h] source table *file*...

Import one or more export file into a table which is located in the data source specified by the source parameter.

Table 34-1. Options

-u login [-l password]	The login name used for the database associated with the source name. This makes only sense if the contents of a certain data source has to be checked, since the login/passwd is only sent once and it will be applied to all given data sources, which is proably not correct.
-q	Run quiet by not printing out the copyright and other notices.
-h	Print out a help message.

VIII. GNADE and the GPS from ACT

Chapter 35. Introduction

The GNADE project components integrate with GPS of ACT by means of help files, usage of project files and some external tools.

After installation of the GNADE package by means of RPM (Linux) or Setup.exe program (Windows), GNADE provides a top level project file which contains most of the defintiontion needed to use GNADE. The location of this project file called gnade.gpr is defined by the environment variable GNADE_BASEPATH. This variable needs to be included in the ADA_PROJECT_PATH allowing the GNAT and GPS to access this project file.

Chapter 36. GPS Project Files

After installation of the GNADE components a typical project file for a project using embedded SQL is shown below:

```
with "gnade";
project Nobel is

   for Languages use ("Ada");
   for Source_Dirs use (".");
   for Object_Dir use ".";
   for Main use ("nobel_co.adb" );

   for Exec_Dir use ".";

   package Linker is
      for Default_Switches ("ada")
      use ("-g", LibOpt, "-lgnadeaux", "-lgnadeodbc",
           "-L" & gnade.DMLOC,
           "-l" & gnade.DMLIB );
   end Linker;

   package Binder is
      for Default_Switches ("ada") use ("-static");
   end Binder;

   package Compiler is
      for Default_Switches ("ada") use ("-g", GNADE.IncludeOPT );
   end Compiler;

   package Builder is
      for Default_Switches ("ada")
      use ("-s", "-m", "-g", "-gnatQ", GNADE.IncludeOPT);
   end Builder;

   package Ide is
      for Vcs_Kind use "CVS";
   end Ide;

end Nobel;
```

Most of the common defintions are taken from the common GNADE project files. If your are developing a pure ODBC Option it will be sufficient to execute "Build" "Make" "Project" from the GPS Menu.

If you are developing SW involving the esql translator a preprocessing run on the embedded SQL source is needed before compiling the source. Unfortunatly the GPS does not allow to execute preprocessor runs. Therefore you have to rely on a Makefile doing the job. This can be done by means of "Build" "Make" "Custom Make".

Appendix A. Frequently asked questions

This section contains the FAQ's of the GNADE project.

Q: How to setup an application development environment

Simply copy the contents of the directory samples/standalone to the place where are going to develop you application and modify the makefile accordingly.

Q: How can i use GPS with GNADE

Pls. refere to the section in the reference.

Q: How to handle strings in where clauses

I like to use strings in the WHERE clause of a query, but nothing seems to work.

In such a situation a length indicator is needed. This is done by adding the INDICATOR keyword as shown below.

Example A-1. Using a string in the WHERE clause

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC
   firstname : CHAR(1..80);
   ..
EXEC SQL END DECLARE SECTION END-EXEC

move( name, firstname );
namelength := INDICATOR_TYPE(name'Legnth);

SELECT
   number,
   ...............
   contact_postcode, contact_country
INTO
   :stu_number,
   ...............
   :stu_contact_postcode, :stu_contact_country
 FROM STUDENT
   WHERE name_first = :firstname INDICATOR :namelength
```

Since GNADE version 1.1.9 the data type VARCHAR has been introduced which already includes the length indicator.

Q: How to handle connection failures

Intercept the DATABASE_ERROR exception as shown below.

Example A-2. Intercpetion connection errors

```
begin

   EXEC SQL CONNECT $DBUSER
            IDENTIFIED BY $DBPASSWD
            BY DB01
            TO $DBSOURCE ;    -- Hallo Test
   ...............
exception
   when GNU.DB.SQLCLI.DATABASE_ERROR =>
      Put_Line("Connection Error");
      .....................
when Others =>
   raise;
```

In addition GNADE esql provide the ON clause in the CONNECT statement which allows to intercept communication and authorization errors.

Appendix B. The GNU.DB Packages

GNU.DB.ESQL_Support

This package contains procedure and functions common to all data base interfaces used by the gesql. Most of the functions located in this package are dedicated to the mapping between ISO/92 and Ada 95 data types.

String related type conversion

```
with SQL_STANDARD;             use SQL_STANDARD;
with GNU.DB.ESQL_SUPPORT;     use GNU.DB.ESQL_SUPPORT;

function To_String(
   Item      : in SQL_STANDARD.CHAR ) return String;

procedure To_String(
   Item      : in SQL_STANDARD.CHAR;
   Target    : out String );

procedure Move(
   S : in String ;
   C : out Sql_Standard.Char );
```

These function is used to convert between ISO/92 Strings and the Ada String type.

SQL Communication Area

This package contains the definition of the SQL communication area, which is updated after each issued sql query.

```
type SQLCA_Type is record
      Message   : aliased String(1..255 );
      State     : aliased SQLSTATE_TYPE;
      SqlCode   : aliased SQLCODE_TYPE;
   end record;
```

The field Message contains a string which is generated by the underlying dbcs in case of errors as informational string. State and SQLCODE contain the result of the last query.

The state variable is a string of 4 characters. The first 2 characters denote the class of the state. The constants SUCCESS_CLASS, WARNING_CLASS and NOT-FOUND_CLASS may be used to distinguish the different error classes as shown below:

```
   if SQLCA.State(1..2) = NOTFOUND_CLASS then
    .....
```

```
end if;
```

Exceptions

This package defines some implementation defined exceptions.

```
Out_Of_Resources     : exception ;
No_Reopenable_Cursor : exception ;
```

The Out_Of_Resources exception is raised by the ESQL_Support module in case where no more internal resources are available. Normally there is no recovery possible and the application should terminate cleanly.

The exception No_Reopenable_Cursor is raised, if a cursor is opened which is not declared as reopen able or local.

ODBC related packages

The packages supporting the ODBC interface ar listed below:

```
GNU.DB.SQLCLI.Bind
GNU.DB.sqlcli-connection_attribute-debug.ads
GNU.DB.Sqlcli.Connection_attribute
GNU.DB.Sqlcli.Desc
GNU.DB.Sqlcli.Diag
GNU.DB.Sqlcli.Dispatch
GNU.DB.Sqlcli.Environment_attribute-debug
GNU.DB.Sqlcli.Environment_attribute
GNU.DB.Sqlcli.Generic_attr-bitmap_attribute
GNU.DB.Sqlcli.Generic_attr-boolean_attribute
GNU.DB.Sqlcli.Generic_attr-boolean_string_attribute
GNU.DB.Sqlcli.Generic_attr-context_attribute
GNU.DB.Sqlcli.Generic_attr-enumerated_attribute
GNU.DB.Sqlcli.Generic_attr-integer_attribute
GNU.DB.Sqlcli.Generic_attr-pointer_attribute
GNU.DB.Sqlcli.Generic_attr-string_attribute
GNU.DB.Sqlcli.Generic_attr-unsigned_attribute
GNU.DB.Sqlcli.Generic_attr
GNU.DB.Sqlcli.Info-debug
GNU.DB.Sqlcli.Info
GNU.DB.Sqlcli.Statement_attribute-debug
GNU.DB.Sqlcli.Statement_attribute
GNU.DB.Sqlcli
```

Appendix C. Porting legacy code

This section describes the migration steps for migrating from legacy code to GNADE embedded SQL. Because only a limited number of ports have been performed this section will evolve over the time.

Migrating from Oracle to GNADE

The Oracle product seems to have a lot of extension compared to ISO/92. Migrating from Oracle to GNADE using ODBC has to be done manually.

Host variables

All host variables have to be moved into the DECLARE section and the types of these variables has to be reworked as it is required by ISO/92.

Query Results

The default SQLCA with the name ORACLE does not exist. Due to the fact, that the contents of the GNADE SQLCA is different this code has to be reworked manually.

Others

The ESQL translator of Oracle supports non ISO/92 WHENEVER clauses which are supported by the GNU ESQL translator as well.

Due to the fact, that ODBC requires different parameters for the CONNECT clause this has to be reworked as well.

Appendix D. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Appendix E. GNU Public License (GPL) Version 2

```
        GNU GENERAL PUBLIC LICENSE
           Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

         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.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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

```
  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.

  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
```