Paradise Version 0.1 Reference Manual [1] [2]

Computer Sciences Department
University of Wisconsin, Madison
paradise@cs.wisc.edu

January 26, 1994

[1]This work was partially supported by NASA Contracts #USRA-555-17 and #NAGW-3895 and by an IBM Research Initiation Grant

Abstract

This document describes Paradise [DKLPY94], a database system designed for handling GIS (Geographical Information Systems) type applications which is under development at the University of Wisconsin - Madison. Paradise uses SHORE [CDF+94] as its underlying persistent object manager. The Paradise front-end is implemented using Tk [Ous91], a publicly available X11 toolkit based on a lightweight interpretive command language Tcl [Ous90]. The design for the Paradise front-end is based on GEO [Vij92], a C++-based graphical user interface for geographical database systems [vOV91]. The current version of Paradise uses a client-server architecture and provides a subset of SQL as its query language. More information about Paradise can be obtained from the Paradise Mosaic home page: http://www.cs.wisc.edu/paradise/.

Contents

1 What is Paradise? 4

2 Installing Paradise 5

2.1 Requirements 5

2.2 Compiling Paradise 5

2.3 Configuring Paradise 6

2.4 Running the Client and Server on Separate Machines 7

3 Getting Started 8

3.1 Starting the Server 8

3.2 Paradise Client Usage Options 8

3.3 Using the Paradise Client in Batch Mode 9

3.4 Starting the Graphical Interface 9

3.4.1 Starting a New Paradise Session 10

3.4.2 Opening a Previously-Saved Paradise Session 10

4 Demonstration 11

5 The Paradise Map View 15

5.1 Map Window 15

5.1.1 Viewing Non-Spatial Attributes 16

5.1.2 Configuring Spatial Attributes 16

5.1.3 Zooming and Centering 17

5.2 Map View Menu 17

5.3 Map View Control Box 18

5.4 Status Box 19

6 Managing the Map Layers 20

6.1 Layer Management Dialog Box 20

6.1.1 Choosing and Reordering the Visible Layers 21

6.2 Presentation Configuration Dialog Box 22

6.3 Configure Appearance Dialog Box 23

7 Issuing Queries **26**

7.1 Query Composer 26

7.2 Query History 28

8 SQL Extensions **29**

8.1 Creating and Loading the Database, Extents, and Indices 29

8.2 Paradise Data Types 30

8.3 Constructor Functions 34

8.4 Updating the Database 35

8.5 Restrictions 36

9 Conclusions **37**

10 Acknowledgements **38**

List of Figures

1 New Session Information Dialog 10
2 File Select Box Dialog 11
3 Paradise Map View Window 15
4 Tuple Dialog Box 16
5 Paradise Layer Management Dialog 20
6 Paradise Presentation Configuration Dialog 22
7 Paradise Configure Appearance Dialog 23
8 Paradise Query Composer Dialog Box 26

1 What is Paradise?

Paradise is a database system aimed at handling GIS types of applications. Paradise provides a graphical user interface for querying and browsing the database and supports a subset of SQL for issuing ad-hoc queries. Paradise provides an extended-relational data model for modeling GIS applications. In addition to support for the base data types such as integer, real, and string, Paradise has built-in support for raster, polygon, polyline, point, circle, and video (mpeg images) data. Paradise uses SHORE [CDF+94] as its underlying persistent object manager.

The Paradise front end allows you to display objects with spatial attributes on a 2-D map. The map provides a layered display for overlapping spatial attributes. Each layer corresponds to one query. You can customize the ordering and display of the layers by selecting color, labelling, and other display attributes for each query.

The database can be queried either with a graphical interface or with ad-hoc queries. The graphical interface can be be used to issue implicit spatial queries by zooming, clicking, and sketching a rubber-banded box. For ad-hoc queries, Paradise provides a query composer. Menus in the query composer provide access to the database schema to assist in query composition. Query results can either be viewed on the 2-D map, if they have spatial attributes, or they can be displayed in a table so that you can browse the tuples in the result. The front end also provides context-sensitive help.

Paradise supports a subset of SQL for issuing ad-hoc queries. To SQL we have added the ability to invoke methods defined on the extended set of types. For example, you can calculate the area of a polygon by using the method polygon.area(). Paradise also supports the standard database operations including create and drop database, create and drop extent, create and drop indices, insert, and update.

The current version of Paradise employs a client-server architecture. The front-end ships queries in our SQL syntax to the Paradise server for execution. After executing the query, the server ships the result objects back to the client. The Paradise server is multi-threaded so that multiple clients can connect to the same server. In designing and implementing the Paradise sever, careful attention was paid to insure that the system could efficiently process queries (especially those involving spatial attributes) on large volumes of data.

2 Installing Paradise

The current version of Paradise includes Paradise source code and binary files and SHORE binary files. If you are not compiling Paradise, you need only refer to Sections 2.1 and 2.3. If you are compiling Paradise, you need to additionally refer to Section 2.2 for compilation instructions. Instructions for running the Paradise client and server on separate machines are provided in Section 2.4. The SPARC architecture is the only architecture supported in the current release of Paradise.

2.1 Requirements

To install Paradise, you will need the items listed below. If you will not be compiling Paradise, you will only need the first two items.

- Sparcstation running SunOS (we run release 4.1.3), with **at least 32MB of main memory**.
- Up to 100MB of disk space for the compiled Paradise code plus disk space for the database volume and log records.
- GNU gcc 2.6.0 or later including the GNU assembler, linker, and preprocessor. You might also be able to use gcc 2.5.8.
- Perl version 4.0 (we use version 4.0.1.8). The files generated using Perl are included with the distribution, so Perl is not needed for the first compilation and installation of Paradise. However, if you use 'make clean' to remove generated files, you will need to use Perl to recompile Paradise.
- Sun's /bin/make or GNU Make (we use GNU Make version 3.71 and /bin/make for SunOS 4.1.3).
- GNU Flex (we use version 2.4.7) and Yacc.
- Imake (we use Revision 1.65 of 7/25/91) from the X Consortium. Imake can be found in your X11 installation as X11/bin/imake.
- The normal Unix utilities like /bin/csh, /bin/sh, /bin/sed, /bin/mv, etc.

2.2 Compiling Paradise

To compile Paradise, you first need to edit the Project.tmpl file, located in the config directory, to set the paths for certain utilities. Then in the main paradise directory issue the following commands to compile and install Paradise. Note that to build Paradise, **gcc, imake, make**, and the normal Unix utilities must be in your path.

```
startup
tools/makemake .
tools/depend
make
make install
```

The **startup** command determines what type of machine you have and sets up a few necessary links depending on the type of machine it is running on. Currently only the SPARC architecture is supported.

The **tools/makemake** command generates the makefiles for Paradise using imake.

The **tools/depend** command generates dependencies. Making dependencies causes some other targets to be made in some directories.

The **make** command completes the actual compilation.

The **make install** command copies all Paradise executables into the directory paradise/installed/bin. This is the directory from which you will run Paradise. In addition, the Tcl library and all Paradise Tcl scripts are placed in the subdirectory of installed named ptk.

The **make clean** command can be used to remove all generated files.

2.3 Configuring Paradise

Before you run Paradise, it must be configured. To do this, you edit a configuration file to specify the log and database storage spaces and set a few environment variables.

1. First, change directory to the directory paradise/installed/bin and copy the file named .pdsrc.tmpl to the file .pdsrc; then edit the file .pdsrc as follows:
   (a) Modify the "shoreserver.pds.server.sm_logdir" variable to specify the appropriate log directory. The database log will be stored in the directory you specify. The log directory must be created before you start Paradise. The location of the log directory should be specified with a full pathname.
   (b) Modify the "shoreserver.pds.server.sm_device" variable to specify the appropriate data storage device. The database will be stored in the device you specify. The device can be a UNIX file or a raw disk. The directory containing the device must be created before you start Paradise. If the database is to be stored in a UNIX file, Paradise will create the file if it does not already exist. The location of the data storage device should be specified with a full pathname.
2. There are four other options in the .pdsrc file. You may edit these options if you are not happy with the defaults.

   Log Size The log size is controlled by the variable "shoreserver.pds.server.sm_logsize" and is specified in Kbytes. The default is 20000 Kbytes.

   diskrw Process The "shoreserver.pds.server.sm_diskrw" variable specifies the location of the diskrw process. The default is the same directory as the .pdsrc file. The location of the diskrw process may be specified with either a full pathname or a relative pathname from the directory where the server executable is located.

   Storage Manager Buffer Pool The SHORE storage manager buffer pool size is controlled by the "*.sm_bufpoolsize" variable and is specified in Kbytes. The default is 16384 Kbytes.

   Device Quota The quota for the database storage device is controlled by the ".pds.server.sm_dev_quota" variable and is specified in Kbytes. The default is 100000 Kbytes.

3. Edit the pds_init file so that the rm command for the log points to your log directory.
4. Set the PDROOT environment variable to point to the paradise/installed directory. The PDROOT environment variable must be set whenever you run Paradise; therefore, you may want add this change to the appropriate shell initialization file.

2.4 Running the Client and Server on Separate Machines

The Paradise client and server can be run on separate machines. To do this, you need to copy or move the executables for the either the client or server to the new machine.

To move the client to a different machine, first copy or move the whole directory paradise/installed/ptk to the new machine. Then copy or move the following files from the directory paradise/installed/bin to the new machine:

- pdfe
- pdwish

If you move the client, you will need to set the PDROOT environment variable on the client machine to point to the ptk directory on that machine.

To move the server to a different machine, copy or move the following files from the directory paradise/installed/bin to the new machine:

- diskrw
- server
- .pdsrc
- pds_init
- pds_restart

If you move the server, you will need to edit the pds_init file to ensure that the rm command for the log points to the appropriate log directory and you will need to edit the .pdsrc file to point to the proper log directory and data storage device.

Finally, on the client machine, set the environment variable PDHOST to the address of the machine on which the server will be running. For example you might set PDHOST to the machine address machine_name.cs.wisc.edu.

3 Getting Started

This section describes how to start Paradise. Paradise implements a client-server architecture; therefore, starting Paradise involves starting both the client and the server. In short, the Paradise client provides a graphical user interface that supports querying, browsing, and updating of Paradise objects and the Paradise server is responsible for executing queries and managing the database including providing full concurrency control and recovery services.

The Paradise client has two modes; batch mode, in which the queries are read in from stdin and the results are sent to stdout, and the graphical interface mode. The batch mode must be used for loading the database.

The sections below discuss starting the server, Paradise client usage options, using the client in batch mode, and starting the graphical user interface.

3.1 Starting the Server

There are two scripts for starting the server: pds_init and pds_restart. pds_init deletes any existing logs and reconfigures the disk volume. pds_init should be used the first time you run Paradise, after that you should use pds_restart to start the server. In either case, the message "Server Initialization Complete!" is printed when the server initialization has been successfully completed. Please note that since pds_init reconfigures the disk volume, it will destroy any databases that have been previously loaded into Paradise when it is run. After its usage the first time, pds_init should be used only if you wish to delete the whole Paradise database and log and start from scratch.

To shut down the server, simply type the command 'shutdown' followed by a carriage return in the window where the server is running. The server will print some statistics to the screen and shutdown.

3.2 Paradise Client Usage Options

The Paradise client has the following usage options:

-usage Print out the usage options.

-v Run the front end with debugging messages turned on.

-b Run the front end in batch mode.

-t Print out the execution time for each query. Available only for batch mode.

-q Do not print query results. Available only for batch mode. This option is primarily for debugging purposes.

-color Select a color scheme for the graphical interface. The valid color schemes are: SGIGray, Gray, TK and Blue. SGIGray is the default.

-font Select a font for the graphical interface. The valid fonts are: 14Point, 12Point, and TK. 14Point is the default.

file Session file name. To start a previously-saved session, you can simply specify the session file name on the command line. The file name should include a relative pathname.

3.3 Using the Paradise Client in Batch Mode

When the Paradise front end is used in batch mode, it reads queries in from stdin and sends results to stdout. The results are printed on stdout in an ASCII format. Batch mode can be used either in conjunction with an ASCII file containing Paradise SQL statements or it can be used to enter queries directly from the command line.

For example, to load a database into Paradise, you could use the Paradise client in batch mode with file redirection as follows:

```
pdfe -b < loadfilename
```

The load file, indicated by 'loadfilename' above, is an ASCII file containing Paradise SQL statements for creating and loading the database. The file could also contain SQL queries to be executed on a database that was previously loaded into Paradise.

The batch mode can also be used to enter queries directly from the command line. To enter queries from the command line, use the following command to start the front end:

```
pdfe -b
```

In this mode, the front end will provide a prompt, 'Ready>', at which you can enter SQL statements.

Whenever you use the front end in batch mode, all statements and queries must be terminated with semi-colons. This is true when you are using batch mode with file redirection and also when you are entering statements from the command line. The command 'quit;' is used to exit batch mode and should be the final command in the load file.

Section 8 gives a description of Paradise SQL commands and includes examples of statements that load a sample database. In addition, there are several load files included in the demo directory of the Paradise distribution that can be used as examples.

3.4 Starting the Graphical Interface

Once a database has been loaded into Paradise using the batch mode, you can run Paradise using the graphical interface. To start the Paradise graphical interface, use the following command:

```
pdfe
```

At this point, you will see the Paradise Startup Dialog Box which reads 'Welcome to Paradise' and has four buttons New, Open, Quit, and Help. The New button opens a new Paradise session, the Open button opens a previously-saved Paradise session, and the Quit button exits Paradise. The Help button accesses the main menu of Paradise on-line help.

Each time you close a database or exit Paradise, you have the option of saving the current session's queries, layers, and view options in a Paradise session file. The saved session can then be reopened later. When you reopen a saved session, the queries that you had specified in that session are automatically executed and the Paradise Map View is put into the same state as when you saved the session.

3.4.1 Starting a New Paradise Session

The New Session Information Dialog shown in Figure 1 allows you to open a new Paradise session. To do this, you must enter a name for this session in the Session Name field and select a database. If you click on the Choose a database button you will see a list of databases currently loaded in Paradise from which you can choose. Click on the name of the database you wish to use. Once you have selected a database and chosen a session name, click OK to continue.

Figure 1: New Session Information Dialog

3.4.2 Opening a Previously-Saved Paradise Session

The open_file dialog, shown in Figure 2 in is used to open a saved session. The dialog contains four fields Filter, Directories, Files, and Selection and four buttons. The fields and buttons are described below. Alternatively, if you know the name of the session that you want to open, you can include the session's name, including its relative path from the current directory, as the last argument on the command line when starting the front end.

Filter This field specifies the filter for the file names and directories that appear in the Directories and Files fields. For example, if you entered /paradise/sessions/*.ses in the Filter field, the Directories field would display all subdirectories of /paradise/sessions/ and Files would display all files in the directory /paradise/sessions/ that end with the extension '.ses'. The Filter field can be edited to change the files and directories displayed.

Directories This field shows all the subdirectories of the directory specified by the Filter field. To browse a different directory, double click on the directory name.

Files This field shows all files meeting the filter specified by the Filter field. To choose a session file, click on the file name and the full file name will appear in the Selection field. In addition, you can just double click on a session file name name to select and open that session.

OK This button causes paradise to open the session file currently specified in the Selection Field.

Filter This button causes the filter specified in the Filter field to be applied and the Directories and Files fields to be updated with the necessary changes.

Cancel The Cancel button returns you to the Paradise Start Up Dialog Box.

Help This button is not currently operational.

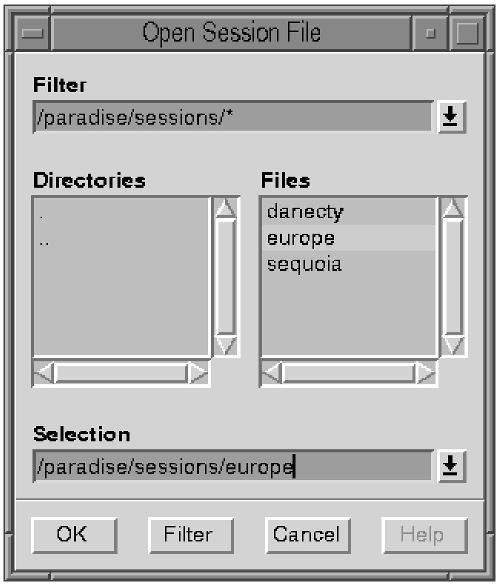

Figure 2: File Select Box Dialog

4 Demonstration

This section contains a demonstration of Paradise. It will walk you through some standard Paradise operations on a sample database. In addition to this demonstration, a directory of demonstration data is included with the Paradise release. This directory is called 'demo' and is a subdirectory of the main paradise directory. The README file in the 'demo' directory explains how to load the demonstration data into Paradise and the file example.sql, also in the 'demo' directory, provides some sample SQL queries for the data.

To begin this demonstration you must start the Paradise server, load the sample database, and start the Paradise graphical interface. If you have not already configured Paradise, you should do so now. Instructions for configuring Paradise are provided in Section 2.3.

- Open two x-term windows.
- In the first window change directory to the directory paradise/installed/bin. If you have already initialized the server, restart the server by typing the following command:

  ```
  pds_restart
  ```

  Otherwise, if you have not initialized the server, use the following command:

  ```
  pds_init
  ```

- Wait until the message "Server initialization complete!" appears on the screen.
- In the second x-term window, change directory to the directory paradise/installed/bin and type the following command to load the demo database. Then wait until the UNIX prompt returns.

```
pdfe -b < ../../demo/europe.load
```

You may need to modify the path for the europe.load file based on your installation of Paradise.

- Type the following command to start the Paradise graphical user interface:

```
pdfe
```

- The Paradise Startup Dialog Box should now appear. Click on the New button to open the New Session Information Dialog Box. Enter the name europe in the Session Name field and click Ok to proceed and open the new session.

The Paradise Map View should now appear. The Paradise Map View displays objects with spatial attributes on a 2-D map. To illustrate the usage of the Paradise Map View, this demonstration will show you how to display some layers in the Paradise Map View. Each layer in the Map View is associated with a query. The first layer that will be added to the Map View is a layer displaying the land in Europe.

The first step in adding the land layer is to use the Query Composer to issue the query. In general, the Query Composer is used to query the database by issuing ad-hoc queries.

- Click on the Layer Mgr button to open the Layer Management Dialog Box.
- Click on the Create New Layer button to open the Query Composer.
- Select land from the View Extent menu in the Query Composer. The SQL syntax for selecting the whole land extent will appear in the Query Composer editor.
- Click on the Show on Map button to execute the query.

The next step is to configure the appearance of the land layer using the Presentation Configuration Dialog box. Each time a query that is to be displayed in the Map View is executed, a Presentation Configuration Dialog box will be automatically opened to allow you to configure the appearance of the newly-created layer. The steps below show how to configure the display of the land layer so that the land is colored gold.

- In the Presentation Configuration Dialog, click on the Customize button for the contour attribute to open a Configure Appearance Dialog Box.
- In the Configure Appearance Dialog Box, click on the Choose Color button and select the color gold from the sub-menu that appears
- Click OK to return to the Presentation Configuration Dialog.
- In the Presentation Configuration Dialog, click the DONE button to view the land layer.
- Finally, close the Layer Management Dialog Box using the Ok button.

The Paradise Map View allows you to zoom into a region by sketching a rubber-banded box around the region to be viewed. Zoom into the region containing England, Spain, and the lower half of Norway and Sweden by completing the following steps:

- Use the middle mouse button to sketch a rubber-banded box around a region containing England, Spain, and the lower half of Norway and Sweden. Detailed instructions for this are provided below:
  - Position the mouse cursor above and to the left of England.
  - Press the middle mouse button, and drag the mouse until Spain is also enclosed in the box.
  - Release the mouse button.

Paradise will zoom into the selected region. If you make a mistake and select too small of a region, use the Zoom Out option in the View window to correct the mistake.

The next step is to add a layer to display the borders of the European countries in the Map View. The steps below show how to add this layer.

- Open the Layer Management Dialog by using the Layer Mgr button.
- Open the Query Composer by using the Create New Layer button.
- Select border from the View Extent menu. The SQL syntax for selecting the whole border extent will appear in the editor.
- Click the Show on Map button to execute the query.
- When the Presentation Configuration Dialog Box appears, simply click DONE to display the borders on the Map View.

Paradise allows query results to be displayed on the Map View or in a tabular format. To illustrate this capability, the next few steps show how to browse the city extent.

- Open the Query Composer and select city from the View Extent menu.
- Click on the Show in Table button to execute the query and browse the result.
- When you are done browsing the result, click the Dismiss button to return to the Query Composer.

The next step is to add a layer displaying all cities in Europe with a population greater than 3,000,000 and that are within 500 meters of a river.

- Type the following query in the Query Composer editor; you can use the SQL Command and Catalog Info menus to help you:

```
select city.name, city.country, city.population, city.location,
       river.shape
from city,river
where river.shape overlaps city.location.makeBox(500)
  and city.population > 3000000
```

- After the query has been entered, click on the Show on Map button to execute the query.

- A Presentation Configuration Dialog Box will appear to allow you to customize the display of this layer. Follow the steps below to change the query name, label the cities by their name, and change the display color for the rivers to blue:
  - Change the query name to citiesrivers by editing the Query Name field.
  - To label the cities by their name, click on the Customize button for city.location. Then click on the Choose Label button and select city.name. Click OK to accept the change.
  - To color the rivers blue, click on the Customize button for the river.shape attribute. Then click on the Choose Color button and select blue. Select OK to accept the change.
- Finally, click on the DONE button in the Presentation Configuration Dialog to display the new layer.

The Paradise Map View provides a multi-layered display to handle objects with overlapping spatial attributes. The following steps illustrate the layering mechanism:

- Click on the 'land' text in the Show box and then click on the Make Top button.
- Notice that the land layer now covers the border and citiesrivers layers where the land layer overlaps the other two layers.
- To return to the original view, click on the 'land' text in the Show box and click on the Make Bottom button.
- When you are finished, click on the Ok button to close the Layer Management Dialog and return to the Map View.

You can click on objects in the Paradise Map View to see their non-spatial attributes. The non-spatial attributes are displayed in a Tuple Dialog Box. The following steps show how to view the non-spatial attributes of the city Paris.

- Click the first mouse button on the circle icon marking the city of Paris.
- A Tuple Dialog Box will appear that displays all the attributes of the Paris object.

This Tuple Dialog Box can also be used to customize the display of the Map View. The following steps show how to change the icons used to represent cities in the Map View so that they appear transparent.

- In the Tuple Dialog Box that you just opened for Paris, click on the button labeled Point; this will open a Configure Appearance Dialog box.
- Click on the transparent button then click OK to accept the customization and close the Configure Appearance Dialog.
- Click Dismiss to close the Tuple Dialog and return to the Map View.
- Notice that the icons representing all cities in the citiesrivers layer have been changed.

The demonstration is now complete. When you are ready to exit Paradise, use the Quit option from the File menu in the Paradise Map View.

5 The Paradise Map View

This section describes the Paradise Map View which is shown in Figure 3. The Paradise Map View displays objects with spatial attributes on a 2-D map. You can point and click on displayed objects to view their non-spatial attributes. You can also zoom into a selected region by sketching a rubber-banded box. In addition, the current position of the cursor is continuously displayed in the Control Window in units of the map projection system. Various menus and buttons are provided for controlling the display. The heading of the Paradise Map View displays the name of the session that is currently being viewed.

There are several parts to the Paradise Map View Window: the Map Window; the Map View Menu; the Map View Control; and the Status Box.

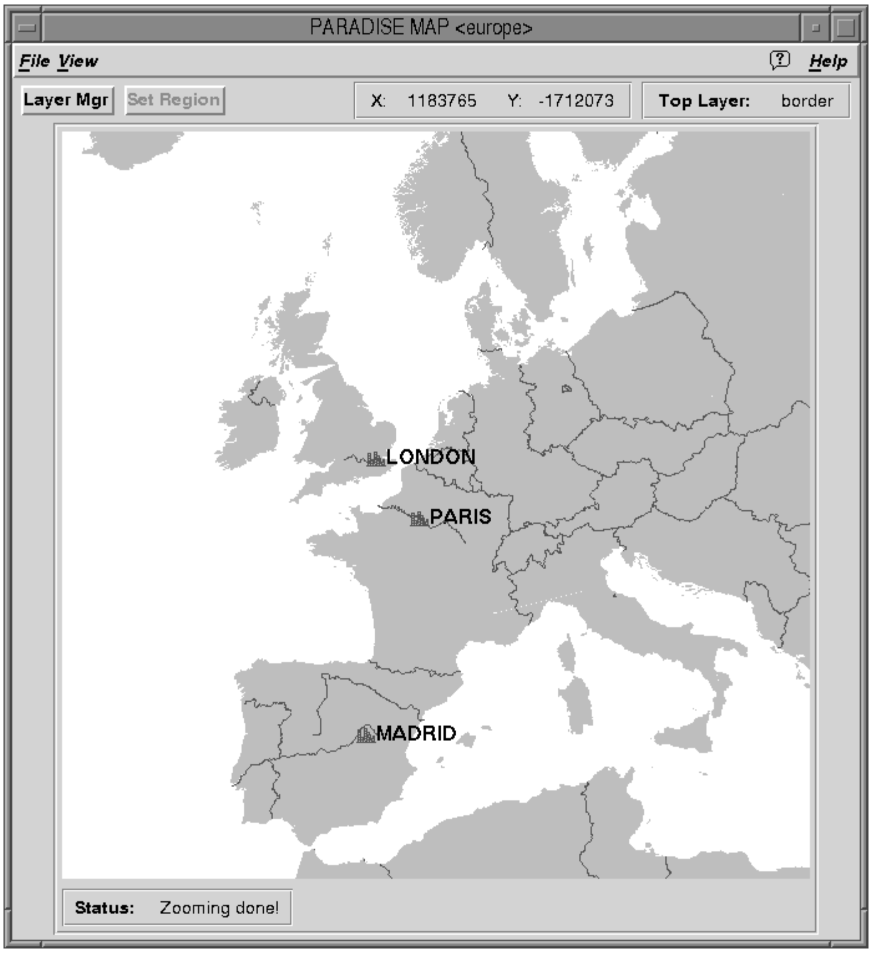

Figure 3: Paradise Map View Window

5.1 Map Window

The Map Window displays a 2-D map of objects with spatial attributes from the database associated with the current session. The Map Window contains a multilayer display to handle objects with overlapping spatial attributes. Each layer in the display corresponds to an extent of objects produced by executing one query on the database. (An extent is a set of tuples and is similar to a relation.) The layers can be raised and lowered to control which layer is displayed when the spatial attributes of objects in two or more layers overlap. See Sections 5.3 and 6.1 for information on controlling the layer display.

5.1.1 Viewing Non-Spatial Attributes

To view an object's non-spatial attributes, click the object with the *left mouse button*. The attributes of the object will then be displayed in a Tuple Dialog Box as shown in Figure 4. The Tuple Dialog Box displays the names, types and values of all of the object's attributes, including spatial attributes, in that order. For spatial attributes, only the coordinates of the first few points of the spatial attribute are displayed.

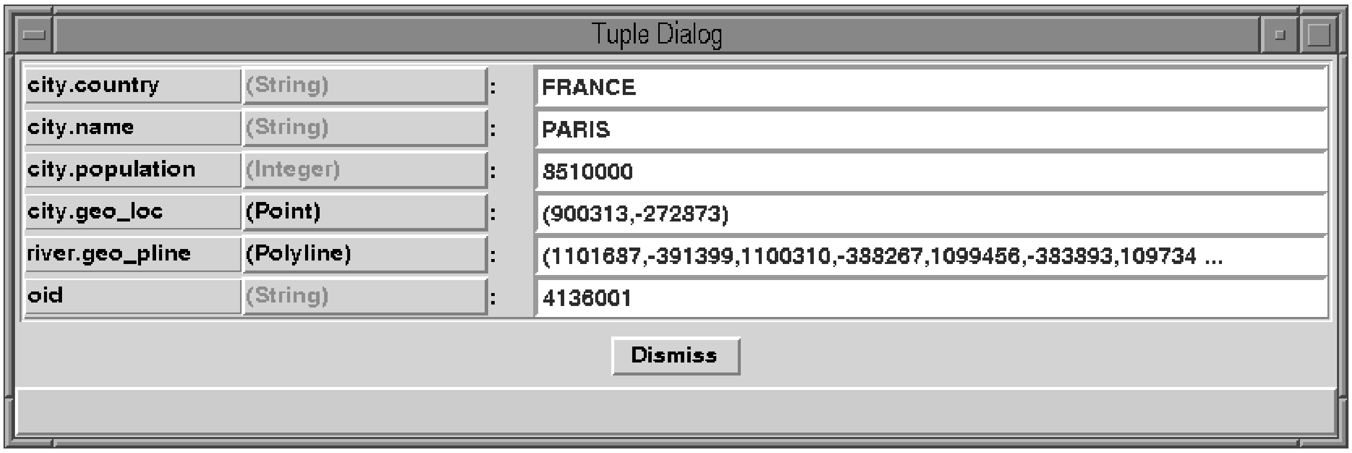

Figure 4: Tuple Dialog Box

5.1.2 Configuring Spatial Attributes

The Tuple Dialog Box described in the previous section (Figure 4) can also be used to configure spatial attributes associated with a layer. When you configure the display using the Tuple Dialog Box, you change the configuration for all the objects of this type in the layer of which this object is a part. You can not configure on an object-by-object basis, unless each object is associated with its own layer. You can, however, choose an option to have the objects in a layer colored by a non-spatial attribute so that objects with different attribute values will have different colors.

In the Tuple Dialog Box, each attribute's type is displayed on a button and by clicking this attribute type button for spatial attributes, you can obtain a Configure Appearance Dialog Box which allows you to configure the layer's display. In particular, you can select the display color, and labelling attribute. In addition, if this layer has more than one spatial attribute, you have the ability to move spatial attribute to the top or bottom of the view hierarchy. Only spatial attributes can be configured; if you attempt to configure a non-spatial attribute by clicking on the non-spatial attribute type, the attribute type will be shaded light gray to indicate that it cannot be configured.

5.1.3 Zooming and Centering

To zoom into a selected region of the map, you can implicitly issue a spatial query by sketching a rubber-banded box on the Map Window using the *middle mouse button*. To sketch this box and zoom into a region, press and hold the middle mouse button and drag the mouse to produce a box. When you release the middle mouse button, Paradise will zoom into the area specified by the sketched box. Note that you are only allowed to sketch square boxes for zooming; this is to avoid distortion of the display.

To change where the Map Window's view is centered, use the *right mouse button* to click on the new center point. Paradise will issue a spatial query and automatically center the view on this new point.

5.2 Map View Menu

This section describes the Map View Menu. The Map View Menu has three main menus File, View, and Help, the options for each of these menus are described below. Some menu options have short-cut keys, these are listed in the menu and are also listed below. In addition, balloons that appear when the pointer appears over a button or menu item and give a brief description of the function of that button or item are available. To turn balloons on or off, click on the question mark next to the Help menu and then click on the Show Balloon Hint button.

File The file menu provides control for opening new Paradise sessions and for exiting Paradise. The file menu options are:

- **New** The New option allows you to open a new Paradise session. This requires closing the current Paradise session. After you select File|New, Paradise will ask you to verify that you want to close the current session. If you respond positively, a Paradise New Session Dialog Box will appear and you can proceed to open a new session. See Section 3.4.1 for details on opening a new Paradise session. Short-cut: Ctrl-n.
- **Open** The Open option allows you to open a previously-saved Paradise session. This requires closing the current Paradise session. After you select File|Open, Paradise will ask you to verify that you want to close the current session. If you respond positively, a Paradise Open Session Dialog Box will appear and you can proceed to open a session. See Section 3.4.2 for details on opening a new Paradise session. Short-cut: Ctrl-o.
- **Save&Quit** Quit Paradise and save the current session in a file. You will be asked to confirm that you want to save the current session. Once you have confirmed, the Save Session File dialog box will appear for you to use to specify the name of the file to save the session in. This dialog can be navigated similar to the Open Session Dialog Box, see Section 3.4.2 for details. If you know the name, including the full pathname, of the file in which you want to save the session, you can type the name in the Selection field and click OK to save the session. Short-cut: Ctrl-s.
- **Quit** Quit Paradise and do not save the current session. Short-cut: Ctrl-q.

View The View menu provides control for some of the view properties. The view menu options are:

- **Refresh** Clicking on the refresh button will cause the display in the Map Window to be redrawn and refreshed. When the display is refreshed, the queries associated with all the display layers are re-executed. The ordering of layers in the Map View is not affected by Refresh. When the refresh is complete, a message to that effect is written in the Status

Box. When a configuration option is changed, sometimes the refresh is automatic, and sometimes the refresh must be done manually by clicking on the refresh button. You will be warned when you need to do the refresh manually.

Zoom Out This option allows you to enlarge, or zoom out, the area shown in the Map Window. You have three options for zooming out: X2, X4, and X8. X2 doubles the view area while leaving the center fixed. X4 and X8 do the same except that they increase the view area by 4 times and 8 times, respectively.

Raise Labels The Raise Labels option causes the map layer containing the labels to be raised so that it is the uppermost, or top, query in the view. This makes the labels visible. If the labels query is not the top query, the labels may be obscured by other object layers.

Show Grids Toggling this button causes a grid to be displayed in the Map Window. To remove the grid, toggle the button again.

Label Grids Toggling this button causes labels for the grid coordinates to be displayed on the Map Window. To remove the labels, toggle the button again.

Help The help menu has the following options:

Contents Accesses the Paradise on-line help. When you select Contents, you are placed into the Map View section of the Paradise on-line help. You can navigate from this section to the rest of the Paradise on-line help.

Current BBox Displays the current bounding box of what is being displayed in the Map Window in the Status Box.

5.3 Map View Control Box

The Map View Control provides information about the Map View display and buttons that can be used to control the display. The Map View Control appears directly below the Map View Menu and is described below.

Layer Mgr Clicking on the Layer Mgr button opens the Layer Management Dialog box. The Layer Management Dialog Box allows you to add, delete, hide, and reorder layers displayed by the Map Window. Each layer corresponds to an extent of objects produced by executing some query. From the Layer Management Dialog Box, you can access the Query Composer which allows you to compose a SQL query using a simple text editor. The Query Composer also allows you to browse extents in the database. In addition, the Layer Management Dialog Box has options that allow you to configure the presentation of layers in the Map Window. See Section 6.1 for a complete description of the Layer Management Dialog Box.

Set Region This button is currently not used.

Cursor Coordinates The current coordinates of the cursor are continuously displayed in this window in units of the map projection system.

Top Layer This box displays the name of the layer displayed in the Map Window. The top layer is the layer that is highest in the hierarchy of layers currently being displayed, so wherever the top layer spatially overlaps other layers, the objects from the top layer will always be displayed. Recall that each layer corresponds to an extent of objects produced by executing some query. Each layer is identified by the name of the query corresponding to that layer.

5.4 Status Box

The Status Box appears at the bottom of the Map View Window and displays Paradise status information, such as a message to indicate that a refresh operation has been completed, or a message to indicate that the Layer Management Dialog Box has been opened and layer management is under way. When the display of the current view's bounding box is requested via the Help menu, this is also displayed in the Status Box.

6 Managing the Map Layers

This section describes how to manage map layers which includes adding, deleting, and hiding layers, configuring layer display, and raising and lowering layers to control which objects are displayed when spatial attributes of multiple layers overlap. The Layer Management Dialog Box, Presentation Configuration Dialog Box, and Configure Appearance Dialog Box are described in this section. The Layer Management Dialog Box itself is used to delete, hide and reorder map layers. In addition, this dialog provides access to the Query Composer Dialog Box (see Section 7.1), which is used to add layers to the Map View, and to the Presentation Configuration Dialog Box, which is used to configure spatial attributes of a layer via the Configure Appearance Dialog Box.

6.1 Layer Management Dialog Box

The primary usage of the Layer Management Dialog box (Figure 5) is to add, delete, configure, hide, and reorder map layers. It is accessed by clicking on the Layer Mgr button in the Paradise Map View. The top portion of the Layer Management Dialog Box consists of Show and Not Show boxes which list the layers currently visible and not visible, respectively, in the Paradise Map View. It also contains four buttons appearing to the left of the Show Box which can be used to control the order of the layers in the Map View. The bottom portion consists of buttons which can be used to create, edit, and delete layers, refresh the Map View, obtain help, and exit the dialog box.

Figure 5: Paradise Layer Management Dialog

With the exception of the Create New Layer, Refresh, Done, and Help buttons, all the buttons are used to modify a specific layer. To use them, you must first select a layer by clicking on the layer's name in the Show or Not Show box. When a layer is selected, its name appears highlighted. The layer's name is the same as the name of the query associated with the layer. When buttons are unavailable for use because there is no selected layer, their labels are shaded light gray.

The buttons appearing in the bottom portion of the Layer Management Dialog Box are described below:

Create New Layer This button opens the Query Composer which can be used to compose and issue queries. See Section 7.1 for a complete description of the Query Composer.

Edit Layer Display This button allows you to configure the display of the selected layer using the Presentation Configuration Dialog Box which is described in Section 6.2.

Edit Layer Query This button allows you to edit the query associated with the selected layer using the Query Composer which is described in Section 7.1.

Delete Layer This button deletes the selected layer. If you wish to view this layer again, you will have to reissue a query. If you want only to temporarily hide the layer from view, see Section 6.1.1 for more information on hiding and redisplaying layers.

Refresh This button will cause the display in the Map Window to be redrawn and refreshed. When the display is refreshed, the queries associated with all the display layers are re-executed. When a configuration option is changed, sometimes the refresh is automatic, and sometimes the refresh must be done manually by clicking on the refresh button. You will be warned when you need to do the refresh manually.

Ok This button closes the Layer Management Dialog Box and returns you to the Paradise Map View.

Help This button accesses the Paradise on-line help. When you select this button, you are placed into the Layer Management section of the Paradise on-line help. You can navigate from this section to the rest of the Paradise on-line help.

6.1.1 Choosing and Reordering the Visible Layers

The Map Window contains a multilayer display to handle objects with overlapping spatial attributes. The layers are ordered in a view hierarchy, ranging from the top layer to the bottom layer. Wherever the top layer spatially overlaps other layers, the objects from the top layer are displayed. For other layers, only those portions of its spatial attributes that are not overlapped by spatial attributes from higher layers are displayed. The only exception is that you can use the View—Raise Labels option in the Paradise Map View to force labels to appear on top of all spatial attributes.

There are two boxes in the Layer Management Dialog Box that provide information about the current layer display: Show and Not Show.

Show The Show box lists all layers currently visible in the Map View. Layers are listed from the bottom layer to the top layer, with the bottom layer appearing at the top of the Show box.

Not Show The Not Show box lists all layers that are not currently visible in the Map View.

The four buttons appearing to the left of the Show Box: Up, Down, Make Top, and Make Bottom allow you to control the order of the layers in the view hierarchy. Only layers that are currently visible can be reordered. To use the Up, Down, Make Top, and Make Bottom buttons, you must first select a layer by clicking on the layer's name in the Show box. The four buttons function as follows:

Up Move the selected layer up one position in the view hierarchy.

Down Move the selected layer down one position in the view hierarchy.

Make Top Make the selected layer the top layer in the view hierarchy.

Make Bottom Make the selected layer the bottom layer in the view hierarchy.

The actions of these buttons take effect immediately and do not cause any queries to be executed.

The two arrows appearing between the Show and Not Show boxes are used to control which layers are visible in the Map View. To hide a layer from view, select the layer by clicking on the layer's name and click the upper arrow which points to the Not Show box. The layer's name will be moved to the Not Show box and the layer will be removed from the Map View immediately. Similarly, to redisplay a hidden layer, select the layer and click on the bottom arrow which points to the Show box. The layer will be inserted as the top layer in the view hierarchy and will immediately be made visible. The hiding and redisplaying of layers in this manner does not result in the execution of any queries.

6.2 Presentation Configuration Dialog Box

The Presentation Configuration Dialog Box show in Figure 6 is used primarily to configure spatial attributes. It can be accessed from the Layer Management Dialog Box via the Edit Layer Display button and is also automatically opened after a query has been issued with the Query Composer.

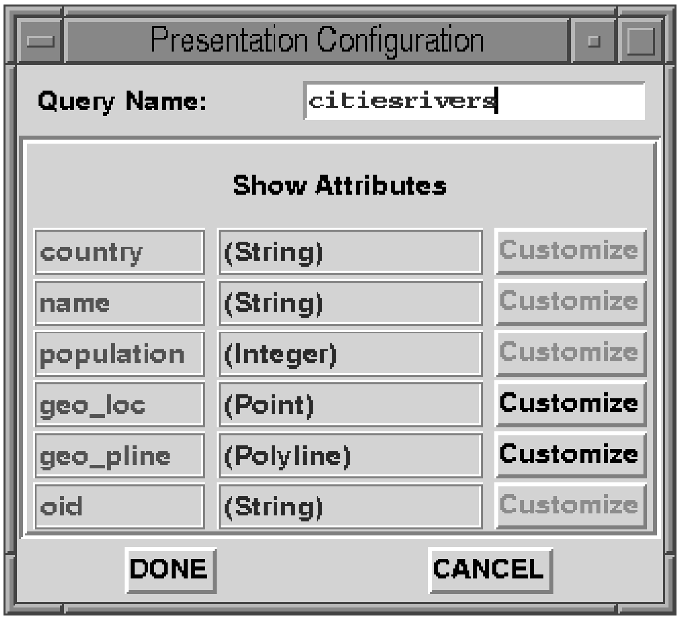

Figure 6: Paradise Presentation Configuration Dialog

The name of the layer being configured appears in the field labeled Query Name. To change the layer and associated query's name, simply click the cursor in this field and proceed to edit the name. The Show Attributes box lists the attribute names and types of all attributes of this layer as shown in Figure 6. Each attribute name, type pair is succeeded by a Customize button. The label on the Customize button appears light gray if the attribute cannot be configured because it is not a spatial attribute. Clicking on a Customize button for a spatial attribute opens a Configure Appearance Dialog Box for that attribute.

The Presentation Configuration Dialog Box contains the following two buttons:

DONE This button accepts the customization changes and displays the configured layer in the Map View. If you have just created a new layer using the Query Composer and if there are no other layers displayed in the Map View, the bounding box of the Map View will be changed to the bounding box of this layer, otherwise the bounding box of the Map View is not changed. When a configuration option is changed, sometimes the Map View is automatically refreshed, and sometimes the refresh must be done manually by clicking on the refresh button. You will be warned when you need to do the refresh manually.

CANCEL This button closes the Presentation Configuration Dialog box and cancels the customization. If you have just created a new layer using the Query Composer and select CANCEL, the customization will be cancelled and the layer will be deleted. To view the layer, you would have to re-issue the query.

6.3 Configure Appearance Dialog Box

The Configure Appearance Dialog Box shown in Figure 7 is used to configure the appearance of a spatial attribute of a layer. This dialog box can be opened from the Presentation Configuration Dialog Box and from the Tuple Dialog Box. When opened from the Tuple Dialog Box, the Configure Appearance Dialog Box has two additional buttons that do not appear when it is opened from the Presentation Configuration Dialog Box. These buttons are Raise and Lower, and can be used to raise and lower the display of a particular spatial attribute. They are particularly useful when a layer has multiple spatial attributes associated with it. A message appearing in the top portion of the dialog box tells you which attribute you are configuring.

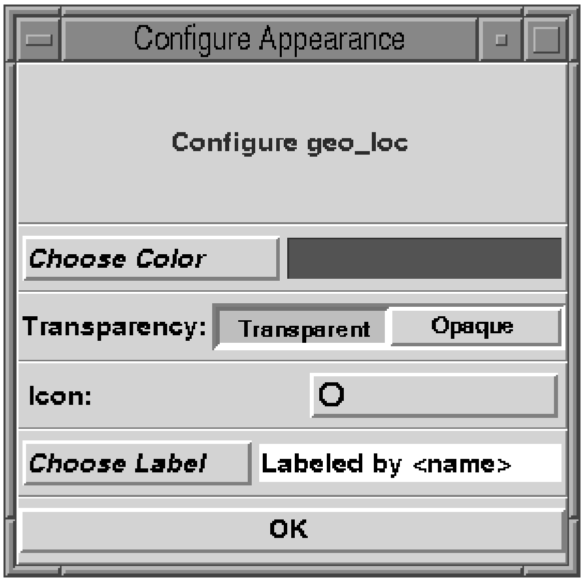

Figure 7: Paradise Configure Appearance Dialog

The buttons of the Configure Appearance Dialog Box are described below. Some buttons have functionality specific to a particular type of spatial attribute and only appear in the dialog box when that type of attribute is being configured. This is indicated in the button descriptions.

Choose Color/Choose Outline Color This button allows you to select the color of the attribute. For polygons, this represents the fill color of the polygon; for polylines, the color of the line; for points, the color of the icon representing the point; and for rasters, the color of the raster outline. There are two coloring options. You can have all the objects colored a specific color, or you can have the objects colored based on an integer attribute of the layer. The mapping between integer attribute value and color is done in a round robin fashion so that two objects with integer attribute values that differ by one will have different colors; however, if there are more integer attribute values than colors, some objects with different integer attributes might have the same color.

When you click on the Choose Color/Choose Outline Color button two choices will appear: Color Palette and Integer Attributes. If the layer has no integer attributes, the only choice will be Color Palette. Placing the mouse pointer over the wording Color Palette reveals a sub-menu of colors. Click on a color to select it. Placing the mouse pointer over Integer Attributes, reveals a sub-menu of integer attribute names. Click on an integer attribute name to select coloring by that attribute. The current color or integer attribute being used for coloring is shown in the field next to the Choose Color/Choose Line Color button.

Transparency Available only for point and raster attributes. For points, this button controls whether the icon representing the point is transparent. For raster attributes, it controls the transparency of the raster image itself. If the raster image is set to transparent, you will be able to partially see other layers through the raster image. The default is Not Transparent. To make the icon or raster transparent, toggle this button, and the radio box will be shaded in to indicate that the Transparent option has been selected. To return to Not Transparent, toggle the button again.

Icon Available only for point attributes. This button allows you to select the icon with which represents the point attribute. To select a different icon, click on this button and click on the icon you want to represent this attribute.

Line Width Available only for polyline and raster attributes. This field allows you to control the width of the lines representing the polylines or the outlines of the raster images. To change the width of the lines, use the arrow buttons to increase or decrease the line width or edit the Line Width field. This field displays the currently selected line width.

Choose Label This button allows you to select how the attribute is labelled. An attribute can be labeled by any or none of the other attributes in its layer. For example a layer corresponding to rivers in North America, might have two attributes: shape, of type polyline, and name, of type string. You could have the shape attribute labeled by the name attribute so that each river would be labeled with its name. To change the labelling of a spatial attribute, click on the Choose Label button to see a list of labelling options. The top item in this list is always None (no label). Click on an attribute name appearing in this list to have that attribute used as a label for the spatial attribute or click None if you don't want the attribute labeled. The current labelling schema is displayed in the field next to the Choose Label button.

Raise Available only when Configure Appearance is opened from Tuple Dialog. This button allows you to make a particular spatial attribute the top layer in the Map View so that you can see all of the spatial attribute. This button only affects the particular spatial attribute which was clicked on to open the Tuple Dialog Box. If a layer has multiple spatial attributes, this button can be used to raise one of those spatial attributes to the top of the map display, so you can distinguish between different spatial attributes associated with one layer.

Lower Available only when Configure Appearance is opened from Tuple Dialog. This button is similar to Raise except that it allows you to force a particular spatial attribute to be the bottom layer in the Map View. This button only affects the particular spatial attribute which was clicked on to open the Tuple Dialog Box.

OK Click OK to accept the configuration of this attribute and return to the Presentation Configuration Dialog Box

7 Issuing Queries

In Paradise, queries are issued using the Query Composer Dialog Box. This dialog box is used both for specifying new queries and editing existing queries. It is opened from the Layer Management Dialog box using either the Create New Query button or the Edit Layer Query button. Paradise accepts queries in a subset of SQL which is described in Section 8. The Query Composer dialog box is described below.

7.1 Query Composer

The Query Composer show in Figure 8 consists of a series of Menus that provide assistance with query composition, a font selector, the query editor, and a series of buttons to control query execution and to access the query history. The Query composer provides assistance with query composition by making schema information, common SQL commands, and simple view extent query syntax accessible via menus. In addition, it provides a query history mechanism so that common queries can be stored in the query history file and retrieved from that file instead of recomposed when they need to be re-issued.

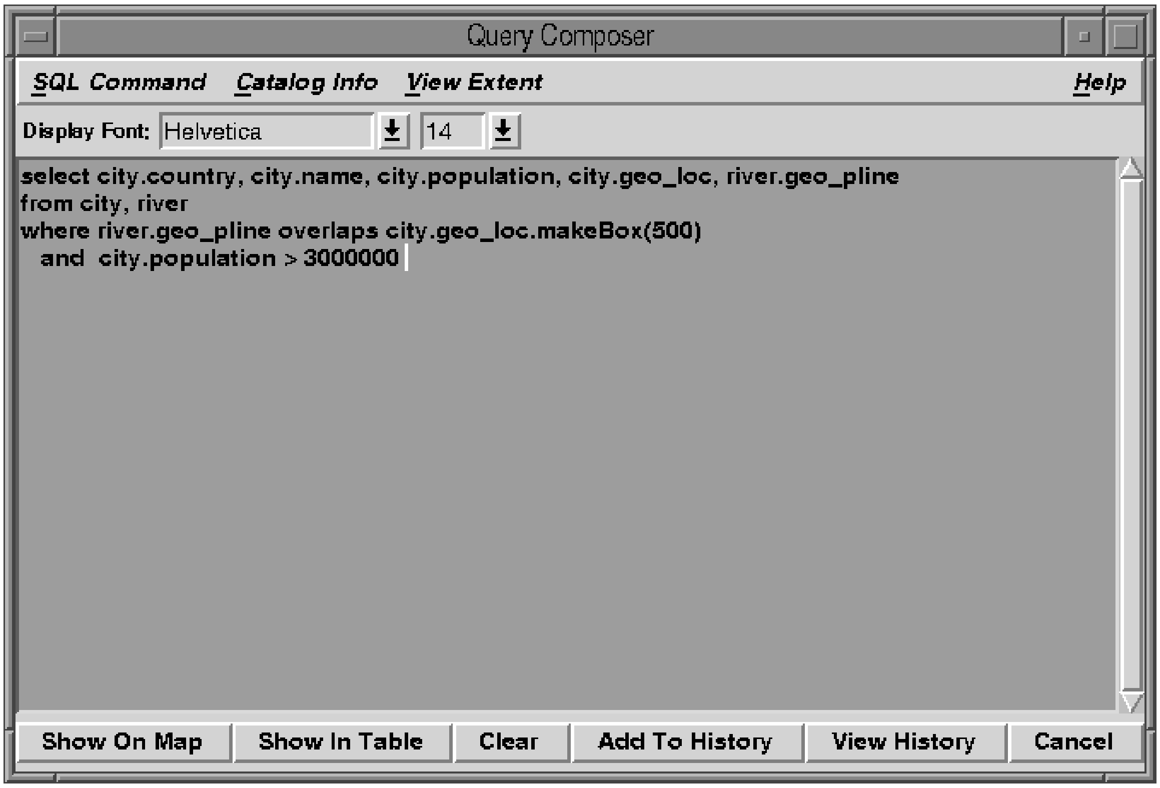

Figure 8: Paradise Query Composer Dialog Box

When you enter the Query Composer to edit a previously-specified query, you might notice that some additional selection predicates have been added to the where clause of the query. These additional predicates have been added by Paradise as it is executing implicitly-issued queries for zooming and centering. These additional predicates usually involve the identifier ##PDMAP## or a bounding box. You should simply delete these additional predicates before editing your query.

The Query Composer Editor, which makes up the main portion of the Query Composer dialog box, is a simple editor with a small set of emacs key bindings.

The function of the menus in the Query Composer are described below:

SQL Command This menu provides access to common SQL commands. The commands currently available are select, where, and ##PDMAP##. ##PDMAP## is an identifier for the current bounding box of the Paradise Map View. It is used to restrict the query to the area overlapping the current Map View. Paradise automatically adds an 'overlaps ##PDMAP##' predicate to all queries.

Catalog Info This menu provides access to the database schema which you can use to insert wording into the query. Clicking on Catalog Info reveals a list of extents in the database. Placing the mouse pointer over an extent name reveals a sub-menu of the attributes of that extent. To insert an extent.attribute_name string into the editor window, click on the attribute name. The string will be inserted into the editor window at the current location of the cursor.

View Extent This menu allows you to quickly compose a query for viewing an extent without typing in the SQL syntax. To view an extent, click on the View Extent menu, you will then see a sub-menu of all the extents in the database. Click on an extent name and the SQL syntax for viewing that extent will automatically appear in the editor. To display this extent on the Map View, use the Show on Map button.

Help This menu has one option, Contents, which allows you to access Paradise on-line help for using the Query Composer Dialog Box. You can navigate from the Query Composer section to the rest of the Paradise on-line help.

The font selector allows you to choose the font for the Query Editor. The Display Font field displays the type and size of the current font selection. To change the font type, click on the arrow button next to the font name. A list of options will appear, click on an option to select it. The procedure for changing the size of the font is similar, except that you click on the arrow button next to the current font size.

The function of the buttons in the Query Composer are described below:

Show On Map This button causes the query shown in the editor window to be executed and displayed on the Paradise Map View. You will be given the option to name the query/layer and customize its spatial attributes before the query is actually displayed.

Show in Table This button causes the query shown in the editor window to be executed and displayed in a tabular format in the Show Table dialog box allowing you to browse the query result. The name of the query being viewed is displayed in the title bar and attribute names appear at the top of each column. If there are more than 300 rows in the result, only the first 300 rows are displayed. There is a Dismiss button in the Show Table dialog box which closes the dialog and returns you to the Query Composer.

Clear This button erases all text in the query editor.

Add To History This button causes the current query to be added to the Query History which is described in Section 7.2.

View History This button opens the Query History dialog box which is described in Section 7.2.

Cancel This button closes the Query Composer dialog box, without executing any queries, and returns you to the Layer Management dialog box.

7.2 Query History

Query History allows you to save commonly-used queries in a query history file which is saved even after the Paradise session is exited. The query history can be viewed from within the Query Composer and you can cut and paste queries from the Query History dialog box into the query editor, instead of retyping the queries.

The Query History dialog box, which is opened by using the View History button in the Query Composer, displays all of the saved queries. To copy a query from the Query History dialog to the query editor, first highlight the query, or portion of query, you want to move, by pressing the left mouse button and dragging over the query. Then move the cursor into the Query Editor and click the middle mouse button to paste. The query is pasted in at the current cursor location. The Query History dialog box also contains two buttons Rescan and Dismiss. Rescan rescans the query history file and updates the Query History dialog box. Dismiss closes the Query History dialog and returns you to the Query Composer.

8 SQL Extensions

This section describes the syntax of Paradise SQL. It assumes a basic knowledge of SQL and only describes the differences between standard SQL and Paradise SQL. SQL can be used as a data manipulation language for issuing ad-hoc queries using the Query Composer and as a data definition language for creating and loading a database.

Paradise supports the standard database operations including create and drop database, create and drop extent, create and drop index, insert, delete, and update. Some features of Paradise SQL are the ability to invoke methods defined on the data types, the ability to use 'constructor' functions in place of literals, and the implementation of spatial operators for querying spatial data. Paradise supports some aggregate operations including group by and order by. For group by and order by operations you are limited to grouping by or ordering by one attribute. Paradise does not currently support nested queries.

This section first describes the commands for creating and loading databases, extents, and indices. Then the Paradise data types are described followed by descriptions of constructor functions and commands for updating the database.

8.1 Creating and Loading the Database, Extents, and Indices

Paradise supports the SQL statements for creation of databases, extents, and clustered and non-clustered indices and for loading files into databases.

Paradise supports two types of indices B+-trees and R*-trees[BKSS90]. B+-trees are used to index non-spatial data types such as int, real, and string while R*-trees are used to index the spatial data types. When you create an index, Paradise selects the type of index to be built based on the data type. Both the B+-tree and R*-tree indices can be clustered or non-clustered; however, clustered indices must be built before the extent is loaded.

A series of statements creating a sample database, its extents and indices, and loading the database is shown below. Recall that if you are using the Paradise front end in batch mode, you must separate all SQL statements with semi-colons.

The following two statements create and open the sample database:

```
create db usa
open db usa
```

The statements below demonstrate the creation of three extents and their associated indices in the sample database. Note that when you create an extent, you do not specify a key. Paradise does not have the notion of keys. You can use any of the words table, relation, or extent when creating a Paradise extent. They are all equivalent.

The native Paradise data types are: Int, Real, String, Point, Polygon, PolyLine, Raster, Circle, and Video. Descriptions of these data types are provided in Section 8.2.

```
create extent State (name String,
                     population Int,
                     boundary Polygon)
create clustered index idxStateLand on State(boundary)
```

```
create extent City (name String,
                    population Int,
                    location Point)
create clustered index idxCityLoc on City(location)
create index idxCityPop on city (population)

create extent CloudCover (time Int,
                          freq Int,
                          cloudDensity Raster)
create index idxCCFreq on CloudCover(freq)
```

The following queries demonstrate loading Paradise extents from ASCII data files. In Paradise, when you load data into an extent, you can specify a list of filenames to be loaded instead of just one filename. This facilitates loading large amounts of data into one extent. In this example, the information for different quarters of the United States is kept in separate files.

```
load extent State from
       "/paradise/states/northwest.dat",
       "/paradise/states/northeast.dat",
       "/paradise/states/southwest.dat",
       "/paradise/states/southeast.dat"

load extent City from
       "/paradise/cities/northwest.dat",
       "/paradise/cities/northeast.dat",
       "/paradise/cities/southwest.dat",
       "/paradise/cities/southeast.dat"

load extent CloudCover from
       "/paradise/raster.dat"
```

The following command closes the database.

```
close db usa
```

8.2 Paradise Data Types

Paradise has nine built-in data types: Int, Real, String, Circle, Polygon, Point, PolyLine, Raster, and Video. Currently the spatial data types are limited to two dimensions.

In the Paradise version of SQL, one can use the standard nested dot notation to invoke methods on data types. Methods invoked on the data types can be in both the selection predicate and the where clause. There are a number of methods that can be invoked on various data types; these methods are described in the data type descriptions below.

Another feature of Paradise SQL is the addition of spatial operators. A new operator, `overlaps`, has been added. The `overlaps` operator determines if two spatial attributes overlap. It can also be used as `not overlaps`. In addition, the operators <, <=, >, >=, =, and != have been overloaded to perform spatial operations for spatial data types. When used with spatial data types, the = and

!= operators perform spatial equality and inequality comparisons, the <, and > operators perform spatial containment and the <=, and >= operators perform spatial containment and equality. Valid uses of the operators are included in the data type descriptions.

The `count` aggregate is supported for all data types. The syntax for this aggregate is `select count(*) from ...`. Count distinct is not implemented.

Data is loaded into Paradise from ASCII files. For raster and video data, the data in the ASCII file references the names of the binary files containing the raster and video data. In general, the coordinates for all spatial attributes must be specified as integers and the x-coordinate is always specified first followed by the y-coordinate. Whenever two integers appear on the same line they must be separated by tabs or spaces. All spatial attributes must be delimited by the characters “<” and “>”. The “<” and “>” characters are used to signal the beginning and end, respectively, of spatial attributes. Blank lines must be used to separate tuples in the input file.

Paradise assumes that the coordinate system used is the LAEA coordinate system. However, you can use any integer-based coordinate system. The only requirements are that all spatial data must be specified in integers and all spatial attributes in the database must be represented in the same coordinate system.

Int This data type represents a 4 byte integer.

Aggregates: The `min`, `max`, `sum`, and `avg` aggregates are supported for the Int data type.

Input format: Integers in input files must appear one per line.

Real This data type represents a 4-byte floating point number.

Aggregates: The `min`, `max`, `sum`, and `avg` aggregates are supported for the Real data type.

Input format: Reals in input files must appear one per line.

String This data type represents a character string. It is not necessary to specify the length of the string when declaring a string attribute.

Operators: The <, <=, =, !=, >, and >= operators can be used to compare two strings. These operators perform lexicographical comparison.

Input format: Character strings in input files must be delimited by double quotes and are terminated by new line characters. There must be only one string per line and strings cannot span multiple lines.

Circle This data type represents a circle in two-dimensional space. A circle is represented as three integers, the first two for the coordinates of the center of the circle and the third for the radius.

Spatial Operators: The spatial operators <, <=, >, >=, =, !=, and `overlaps` can be used to spatially compare circles. In addition, the < and > operators can be used to compare an instance of type Polygon to an instance of type Circle to determine if the Polygon is contained within the Circle.

Input format: The ASCII input must consist of one line containing three integers and the spatial attribute delimiters “<” and “>”. The three integers must be the x-coordinate and y-coordinate of the center of the image and the radius of the circle, in that order. In addition, the line must begin with a “<” and end with a “>”.

Point This data type represents a point in two-dimensional space. Points are represented by two integers.

The following methods can be invoked on Points:

Polygon Point.makeBox(int size) This method creates a square box centered at the point with sides equal to size.

Circle Point.makeCircle(int size) This method creates a circle centered at the point with radius equal to size.

Aggregates: The `closest` aggregate is supported for the Point data type. This aggregate finds the closest point to a given point. To illustrate its operation, consider the following query which finds the city which is closest to Madison.

```
select closest(c2.location, c1.location)
from City c1, City c2
where c1.name = "Madison"
  and c2.name != "Madison"
```

Spatial Operators: The =, != operator can be used to determine if two points are equal or not equal, respectively. In addition the <, >, and `overlaps` operators can be used to compare Points to Polygons to determine if a Point is within a Polygon.

Input format: The ASCII input must consist of one line containing two integers and the spatial attribute delimiters "<" and ">". The two integers must be the x-coordinate and y-coordinate of the point, in that order. The line must begin with a "<" and end with a ">".

Polygon This data type represents a closed polygon in two-dimensional space. A closed polygon is represented as a series of integer pairs.

The following methods can be invoked on Polygons:

int Polygon.area() This method returns the area of the polygon. Currently the area returned is the area of the polygon's bounding box; this will be refined in the future.

Polygon Polygon.box() This method returns the bounding box of the polygon. The bounding box is the smallest rectangle within which the polygon is completely contained.

The following query demonstrates the application of the Polygon area method. It selects the name and area of all states with area greater than 1,000,000 square meters.

```
select State.name, State.boundary.area()
from State
where State.boundary.area() > 1000000
```

Aggregates: The `closest` aggregate is supported for Polygon data types. This aggregate finds the closest Polygon to a given point. To illustrate its operation, consider the following query which selects the state whose boundary is closest to the city named Madison.

```
select closest(State.boundary, City.location)
from State, City
where City.name = "Madison"
```

Spatial Operators: The spatial operator `overlaps` and the operators =, !=, <, <=, >, and >= can be used to spatially compare Polygons. In addition, the `overlaps` operator can be used to spatially compare Polygons to Points and PolyLines and the < and > operators can be used to compare Polygons to Points and Circles. The < and > operators are used to determine if a Polygon is contained within a Circle and to determine if a Point is contained within a Polygon.

The following query demonstrates the use of the < operator for spatial comparison. It selects the name and boundary of the state where Madison is located.

```
select State.name, State.boundary
from State, City
where City.location < State.boundary
  and City.name = "Madison"
```

Input format: Polygons must be represented by a series of lines containing two integers each. The x-coordinate must appear first on the line followed by the y-coordinate. The characters "<" and ">" must be used to indicate the beginning and end, respectively, of the Polygon. The "<" should precede the x-coordinate on the first line of the Polygon input, and the ">" should follow the y-coordinate on the last line of the Polygon input. The last point in the polygon does not have to be the same as the first point in the polygon; the Paradise code takes care of making sure that the edge from the last point to the first point is always connected.

PolyLine This data type represents a polyline in two-dimensional space. A polyline is represented as a series of points.

Spatial Operators: The =, !=, and `overlaps` operators can be used to spatially compare PolyLines. In addition, `overlaps` can be used to spatially compare a PolyLine to a Polygon.

Input format: Polylines must be represented by a series of lines containing two integers each. The x-coordinate must appear first on the line followed by the y-coordinate. The characters "<", and ">" must be used to indicate the beginning and end, respectively, of the Polyline. The "<" should precede the x-coordinate on the first line of the Polyline input, and the ">" should follow the y-coordinate on the last line of the Polyline input.

Raster This data type represents a raster image.

Since raster images can be very large, Paradise employs several techniques to improve performance on rasters. Very large raster images are stored in two pieces in Paradise: a *raster header* and the actual *raster image*. The raster header is used to store descriptive data. To further enhance performance, each raster image is decomposed into regular rectangular regions called *tiles*. The data in each tile is stored as a separate object and a map table (one per raster image) is used for maintaining the correspondence between the tiles and the region of the raster image corresponding to that tile. Paradise supports lossless compression for raster images. The unit of compression is a tile and the basic LZW algorithm [Wel84] is used to perform the compression.

The following methods can be invoked on Rasters:

Raster Raster.clip(Polygon shape) This method returns a raster image clipped by shape.

Raster Raster.lower_res(int factor) This method lowers the resolution of a raster image. This method does not modify the actual raster image, rather it creates a new raster image with the resolution lowered by factor and returns that image. For example, if you call this method with a parameter of two, the resolution of the raster image will be lowered two-fold on the x-axis and two-fold on the y-axis.

Real Raster.average() This method calculates the average value of all pixels in a raster image.

The following query demonstrates the application some of the raster data type methods. This query first selects all rasters that meet the selection predicate frequency = 5 and time = 1. Then these rasters are clipped by the boundary of the state of Wisconsin. The clip method produces a raster image as output and the lower resolution method is applied to that output raster image to produce the final result.

```
select CloudCover.cloudDensity.clip(State.boundary).lower_res(8)
from CloudCover, State
where CloudCover.freq = 5
and   CloudCover.time = 1
and   State.name = "Wisconsin"
```

Aggregates: The `avg` aggregate is supported for the Raster data type. This aggregate averages multiple raster images and produces a raster image as the result.

The following query selects all cloudDensity raster images taken at time 1 and creates a new result raster image from those selected images. Each pixel in the result image is created by averaging the pixel values taken from each of the selected images at the coordinates of the result pixel.

```
select avg(CloudCover.cloudDensity)
from CloudCover
where CloudCover.time = 1
```

Input format: Raster input consists of two parts, an ASCII specification of the raster image and the binary raster file. The ASCII specification must contain four lines as follows: the first line must contain the delimiter "¡" followed by the number of raster cells on the x-axis followed by the number of cells on the y-axis; the second and third lines must contain the coordinates of the lower left-hand and upper right-hand corners of the raster image, respectively; the fourth line must contain the full pathname of the file containing the binary raster image followed by the delimiter "¿". The numbers of cells and coordinates of the lower-left and upper-right hand corners must be specified in integers. The name of the binary raster file must be enclosed in double quotes (i.e. "/paradise/demo/sequoia/raster"). The raster image must be a 16-bit raster image in AVHRR format and must be rectangular in shape.

Video This data type represents a video image. Video images must be in mpeg format to load them into Paradise.

Input format: The ASCII file must contain two lines: the first line must contain the delimiter "¡" followed by the name of the binary mpeg file and the second line must contain the name of the binary mpeg stat file followed by the delimiter "¿". The names must be full pathnames and must be enclosed in double quotes. If you do not have the binary mpeg stat file for a video image, you can use the function mpegidx located in the directory ??? to create the stat file. A README in this directory provides instructions for generating the stat file.

8.3 Constructor Functions

The Paradise version of SQL includes support for constructor functions. These functions can be used to create Circles, Points, Polygons, and PolyLines. These functions can be used anywhere in a query that a literal or an attribute name can normally be used. An example of a constructor function is Polygon(). This function takes a series of pairs of integers and constructs a Polygon instance from those integers. This polygon can then be used anywhere that an attribute of type Polygon could be used. Constructor functions can also be nested. The four constructor functions are described below.

Circle(Point center,int radius) This function creates a circle centered at center with radius radius. This function returns a Circle.

Polygon("int xcoord1 int ycoord1, ...") This function creates a closed polygon. The argument to the constructor should be a string containing a list of pairs of integers. The two integers within a pair should be separated by a space and each pair should be separated from the next pair by a comma. This function returns a Polygon.

Point(int xcoord, int ycoord) This function creates a point with the coordinates as specified. The first argument must be the x-coordinate and the second argument must be the y-coordinate. This function returns a Point.

PolyLine("int xcoord1 int ycoord2, ...") This function creates a polyline. The argument should be a string containing a list of pairs of integers. The two integers within a pair should be separated by a space and each pair should be separated from the next pair by a comma. This function returns a PolyLine.

The following two queries demonstrate the use of constructor functions.

The first query creates a Polygon instance out of the points provided and then selects the names and boundaries of all states whose boundaries overlap the newly-created closed polygon. The closed polygon is automatically deleted after the query has been executed. This query also demonstrates the use of the spatial operator `overlaps`.

```
select State.name, State.boundary
from State
where State.boundary overlaps Polygon("17000 58000,
                                       16000 58000,
                                       16000 48000,
                                       17000 48000")
```

The second query selects the names and boundaries of all states that are within the circle centered at the point 17000, 58000 with radius 325,000 meters. This query also demonstrates the usage of nested constructor functions and the use of the overloaded operator, <, for spatial comparison.

```
select State.name, State.boundary
from State
where State.boundary < Circle ( Point (17000, 58000), 325000)
```

8.4 Updating the Database

Currently, Paradise supports a limited set of update operations. In particular, you can update the value of an attribute by setting it equal to a new value. The SQL `insert` and `delete` operators are also supported. Arithmetic operators are currently not supported.

For example, you might use the following query to update the population of the city Minneapolis:

```
update City
set City.population = 3000000
where City.name = "Minneapolis"
```

The constant 3000000 could be replaced by another attribute from the city extent, provided it was of the same type as City.population. Currently, all updates are processed using a sequential scan of the extent.

8.5 Restrictions

The following operators are not implemented in Paradise: distinct, not, unary minus, unary plus, plus, minus, mult, and division. In addition, Paradise does not support null values and group by and order by operations are limited to one attribute. All sorts are automatically ascending. Paradise does not support nested queries. Note that the != operator can be used to determine inequality.

9 Conclusions

This document describes the client-server version of the Paradise GIS currently under development at the University of Wisconsin Madison and build on SHORE. Paradise provides an extended-relational data model with support for point, raster, polygon, and polyline data types, and a subset of SQL for formulating ad-hoc queries. A graphical user interface based on the Tk toolkit allows the user to query and browse the database graphically.

10 Acknowledgements

Paradise was developed by Navin Kabra, Jun Luo, Jignesh M. Patel, and Jie-Bing Yu under the supervision of David J. DeWitt and Jeffrey F. Naughton. The documentation was prepared by Kristin A. Tufte with help from all of the above. We would like to thank C. K. Tan and Mike Zwilling for their assistance with the SHORE storage manager.

References

[BKSS90] N. Beckmann, Hans-Peter Kriegel, R. Schneider, and B. Seeger. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". In *Proceedings of the 1990 ACM-SIGMOD Conference*, June 1990.

[CDF+94] M. J. Carey, D. J. DeWitt, M. J. Franklin, N. E. Hall, M. McAuliffe, J. F. Naughton, D. T. Schuh, M. H. Solomon, C. K. Tan, O. Tsatalos, S. White, and M. M Zwilling. "Shoring up Persistent Objects". In *Proceedings of the 1994 ACM-SIGMOD Conference*, Minneapolis, MN, May 1994.

[DKLPY94] D. J. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J. Yu. "Client-Server Paradise". In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994

[FK93] C. Faloutsos and I. Kamel. "Packed R-Trees Using Fractals". In *Conference on Intelligence and Knowledge Management*, November 1993.

[Ous90] J. Ousterhout. "Tcl: An embeddable command language". In *Proceedings of the 1990 Winter USENIX Conference*, 1990.

[Ous91] J. Ousterhout. "An X11 toolkit based on the Tcl language". In *Proceedings of the 1991 Winter USENIX Conference*, 1991.

[vOV91] P. van Oosterom and T. Vijlbrief. "Building a GIS on Top of the Open DBMS Postgres". In *Proceedings EGIS'91, Second European Conference on Geographical Information Systems*, April 1991.

[Vij92] Tom Vijlbrief. "The GEO System Version 1.33", Technical Manual, TNO Institute for Perception, P.O Box 23, 3769 ZG Soesterberg, The Netherlands, 1992

[Wel84] T.A. Welch. "A Technique for High-Performance Data Compression". *IEEE Computer*, 17(6), 1984.