A Gopher Interface to Relational Databases

Paul Lindner

April 18, 1994

At the University of Minnesota we've developed software that allows a Gopher user to access the data in an SQL database. This piece of software is called a gateway. In general a gateway translates the operations and data of one system into operations supported by a different, incompatible system.

Our gateway translates Gopher operations into SQL statements and the SQL results are translated into Gopher data. The gateway method of accessing the database simplifies the allowable operations to a limited, yet useful subset of the allowable SQL queries.

We've been using this software for some time now. We demonstrate some sample databases that the gateway is currently using and draw some conclusions about the software.

1.0 Introduction

The Internet Gopher is a suite of software that allows for easy access to network-based information. Initially developed at the University of Minnesota in early 1991, it has spread to over 1600 sites worldwide as of July 1993.

Gopher is a client-server system that can be used to build a Campus Wide Information System (CWIS). Clients, which browse and search information are available for most major platforms (Macintosh, DOS, Windows, Unix, VMS, MVS, VM/CMS, OS/2). Servers, which translate and publish information, are also available for all of the platforms mentioned above.

This client-server architecture uses the Internet Gopher Protocol [Gopher91, RFC1436]. The Gopher protocol has been described as "brutally simple." It is based on a web/tree metaphor of files and directories. Its basic primitives are a list directory transaction, a retrieve file transaction and a search for directory entries transaction.

Given this design it isn't surprising to see that most implementations of Gopher Servers map a filesystem to GopherSpace. An example of a file system hierarchy would be the registry of all gopher servers. The registry is divided into a number of directories based on geography. At the top level is the directory of continents, then there are subdirectories of countries for each continent, then a state or province subdirectory. After all of this digging you will find the information you want.

For most data this approach works rather well; a good portion of the data people want to publish are in files located in directories. However there is a whole class of data that doesn't fit into a file system that easily: databases, especially relational databases. Databases are better than files for a number of reasons including data consistency and multiple indexes [Lindner93].

To handle this type of data we've developed a gateway that translates Gopher requests into SQL (Structured Query Language) statements and SQL results into Gopher data. This gateway allows a Gopher user to look at the data inside of SQL tables using the gopher browse/search metaphor. It simplifies the allowable operations on the database to a limited, yet useful subset of the allowable SQL queries.

Since most of our campus knows how to use Gopher already, training time is minimized. Also, Gopher is cost-effective: commercial software packages usually require you to spend over 200 dollars per machine for software to access the SQL database. This software may be useful for some, but for most people on our campus the Gopher interface is sufficient.

2.0 Features of the SQL Gateway

The SQL gateway allows people using a Gopher Client to access the data contained in an SQL database without having to know SQL. The gateway is the only portion of code that needs to know any SQL. The clients can be used as is with the gateway, no modifications for the clients are needed.

The SQL gateway accepts gopher requests and translates them into SQL statements that get passed via TCP to either a Sybase or Oracle database.

The SQL gateway allows the Gopher Client to:

- View the tables of a database as a Gopher directory
- View the columns of a given table as a Gopher directory
- View the contents of a column as a Gopher directory
- See how many records will result from a query before viewing records
- View records as formatted text.
- View/import records as tab-separated-values
- Add records to a table. o Search the table by filling out a Gopher+ form.

The server/gateway administrator has control over the configuration. The administrator can give column/table names more descriptive titles, and can link columns together to make a subdirectories via implicit joins.

3.0 Semantics of a Gateway, or how a Gateway works

A gateway is a simple thing. It translates commands and data from one format to another. Gopher has long used gateways to lash together disparate information systems such as USENET, Archie, X.500, FTP, WAIS and others. In fact Gopher has been referred to as the "Duct Tape of the Internet" by some.

The SQL gateway translates Gopher Operations into SQL statements. We do lose functionality when doing this however. The large set of possible operations in SQL would be hard to present using the very simple Gopher protocol operations. (It might be possible with a large enough directory tree though..)

3.1 Semantics of the Gopher Protocol

The Gopher Protocol is very simple information retrieval tool based on the client-server model. It uses three basic transactions/commands.

- Directory listing
- File retrieval
- Search and return a directory listing

These simple directives are quite powerful, over 1500 sites over the world now use the base Gopher protocol. In the Spring of 1992 we proposed a suite of upward/downward compatible extensions to the Gopher protocol called "Gopher+". Gopher+ adds the following features to the base gopher protocol:

- Forms input
- Multiple alternate document representations (VIEWS)
- Metainformation about an object (administrator, size, etc..)

The SQL Gateway uses the Gopher+ protocol for some of it's features: optional tab-separated documents and solicitation of searches via a form.

3.2 Semantics of the Structured Query Langauage (SQL).

SQL is a full featured database access language. Some of the more common operations supported by SQL are:

- o “Select”ing records from a table
- “Insert”ing records into a table
- Searching/querying multiple tables for information
- Ordering of results
- Grouping of results
- Creation/deletion of tables or views
- Computations on data (aggregate values)

In addition, most SQL databases support the concept of the Data Dictionary. The data dictionary is a database that describes the contents of other databases, tables and columns. Different vendors have different data dictionary formats, this can cause some problems..

3.3 Semantics of the SQL Gateway

The SQL gateway understands a small limited number of commands. These commands perform the mapping between Gopher operations and SQL statements. The gateway understands the following commands:

- Get a listing of all tables
- Get a listing of columns in specific tables
- Get a list of distinct values in a specific column
- Display records given a search
- Insert a new record
- Delete records

3.4 Mapping Operations and Data Between Different systems

Any gateway maps between multiple sets of available operations and data formats. The Gopher to SQL gateway maps gopher operations (selecting a directory, choosing a file) into it’s internal command set (“tables” command, “get” command) and then into SQL (select tables from dictionary, select * from tablename).

It then translates the results received from the SQL query back down the chain. For instance the results of a “select tablename from dictionary” would generate intermediate gateway operations “columns tablename”. These commands would finally be translated into the gopher directory format and sent to the client.

The following table summarizes the equivalency between the different command sets.

Gateway Command	Gopher Data	SQL Statement
tables	Directory	select tablename from dictionary
columns <table>	Directory	select columnname from dictionary where tablename=<table>
list <table.column>	Directory	select distinct table.column from table
get <table.column> <query>	Text	select * from table where query

4.0 Sample Databases

We have been using this gateway to provide access to a number of databases at the University of Minnesota. We detail the setup of some of them here.

4.1 The Sample “pubs2” database

The Sybase Database comes with a sample database of authors, titles, publishes and stores called “pubs2”. This database demonstrates a number of features, including 1:N and N:N relations.

For this database we set up a few sample queries at the top-most level and added an entry to view the database in it’s raw format: A tree listing of the gatewayed access to the database is below. A complete listing would be quite huge, only portions of the tree are shown below.

```
Authors by City/
  Ann Arbor
  Berkeley (2)
  Cornvalis
  ...
Authors by Name/
  Bennet
  Blochet-Halls
  Carson
  ...
```

Titles by Name/
  But is it User Friendly?
  Computer Phobic and Non-Phobic Individuals: Behavior Variations
  Cooking with Computers: Surreptitious Balance Sheets
  ..
Titles by Publication Date/
  Jun 12 1985 12:00:00:000AM
  Jun 18 1985 12:00:00:000AM
  Jun 30 1985 12:00:00:000AM
Raw DB access /
  Author Pictures/
  Author List/
    All Records (23).
    Add a Record
    Author ID Number/
    Author Last Name/
    Author First Name/
    Phone/
    Address/
    City/
    State/
    Country/
    Postalcode/
    Multiple Field Search, sorted by Author ID Number
    Multiple Field Search, sorted by Author Last Name
    ...
  Author Descriptions/
  Discounts/
  Publishers/
Roysched/
  Sales/
  Salesdetail/
  Stores/
  Author--Title Join Table (Not Interesting...)/
  Book Titles/
  Titleview/

4.2 The Financial Detail Reporting Database

This database is considerably more complex than the PUBS2 database. This database contains all the financial transactions of the University of Minnesota. This is one _big_ database! The monthly transactions are in the four to five hundred thousand range!

We set up a menu system to do specific queries to this database. This keeps users from running inefficiently formed queries. The menu structure was created on a Unix machine running the Unix Gopher Server software. This server also provides explanations about what all the different code numbers mean. All of the SQL gateway functionality is still in the SQL Gateway. The Unix server makes "links" to the SQL Gateway, using it's functionality at specific points in the hierarchy.

This is the tree listing of the directory structure of the Financial Reporting Database System.

About The Financial Reporting Database
  Terms used in the Financial Reporting Database
  Areas and Organizations/
    List of Areas and Organizations by Area Name
      COMPUTER AND INFO SYSTEMS
      COMPUTER SCIENCE
      ...
    List of Areas and Organizations by Function/

  Balance Sheets/
    List of Balance Sheets by Area Number/
      0510
      0620
      ...
  This Months Transactions/
    Search Transaction Table (form)
    Transactions by Fund Number

5.0 Technical Details

It isn't too difficult to set up a Gopher to SQL database. This section gives an overview of how the software is constructed.

To run the server you will need some special software.

If using Sybase:

You will need a machine with the Sybase client libraries (usually stored in /usr/sybase) and a special version of perl called sybperl [Peppler92].

If using Oracle:

You will need a machine with the Oracle client libraries and a special version of perl called oraperl [Stock93].

Each of these uses specific "glue" routines to implement database specific features (connecting, data dictionary, etc.). Thus it is fairly easy to extend the gateway to deal with other database vendors.

5.1 Command line options understood by the gateway

Major parameters are set via command line switches. The following table summarizes them:

Option	Description
-h	Hostname of the gateway
-p	Port number of the gateway
-T	Database Type (oracle, sybase, etc.)
-S	SQL Server to connect to.
-D	Database to use
-U	Username to use
-P	Password to use

5.2 Selectors understood by the Gateway...

The gateway implements a small internal command set. The following summarizes this small command language.

- tables [<search>]

  This command uses the data dictionary to make a gopher directory list of the different tables in the database. This is most useful for allowing raw database access. This command generates "columns" commands that the client will execute.

  An optional search item will restrict the tables to match the search term.

- columns <tablename> [<search>]

  This command uses the data dictionary to make a gopher directory list of the columns in <tablename>. This command generates "list" commands for each column in the table. It also generates a "get" command that will retrieve all the records of a table as text.

  After the column names are presented, a list of search items is presented. If the client is using Gopher+ it gets a list of forms.

  If the user can write to the table an "insert" item for adding an item to the database is added to the menu.

  An optional search item will restrict the tables to match the search term.

- list <tablename.columnname> [<fromtables> [<query>]] [<search>]

  This command generates a listing of the unique items in the given table and column.

  This command can be part of a multiple series of queries by specifying an optional list of tables to choose from and an query. This gets translated into SQL that looks like this:

  select ..... from <fromtables> where <query>

  An optional search can search for specific titles in the specified table and column.

  An optional form may be specfied if this command is used as *asklist*

- get <tablename.columnname> <fromtables> <query>

  This command actually retrieves a record from the database as a textual item. The default is to print out each column name, a colon and the data contained in it. Multiple records are separated with a line of "dashes".

  Optionally one may define a module for a specific table. This module is a file containing perl code that can do sub-queries and fancy reformatting of the data into any format you desire.

- insert <tablename>

  This command inserts a new record into the specified table. The values for the table come from the ASK Block.

5.3 Files used by the Gateway

The gateway uses a number of configuration files to control and enhance the way it operates. These files and their format are summarized below:

- namelist.<databasename>

This file is basically a local table of mappings between column/table names and Gopher Title Names. The format is:

<table>:<Gopher Table Name> <table.column>:<Gopher Column Name> <.Column>:<Gopher Column Name>

Eventually this table could be kept on the database itself, eliminating the need for a local filename.

- jointable.<databasename>

  This file allows you to link together two tables, allowing you to do implicit joins between the two tables.

  The format is as follows:

  <source-table.column>:<target-table.column>:<additional_tables>:<join query>

- <tablename>.module

  Files in this format can override the default record display routines. Thus, if you want to join data in with the text, this is one way to do it.

  For instance, in the PUBS2 Database the author table display record is overridden so we can display the address correctly, and also join in the au_blurbs table.

6.0 Conclusions:

In the short time we've been operating the gateway we've noticed many things. First, most people seem to like not having to learn SQL to get their data. They like the ease of use of making an SQL query with the click of the mouse.

The gateway allows us to control access to our SQL server in more sophisticated ways. We've set up transactions that we know won't take forever. This is a pitfall if you're using SQL directly. We are enjoying some vendor independence now, since the gateway sits at a layer above the RDBMS. We could migrate our data to another system if we wish.

In short we've been very pleased with this software and will continue to enhance, modify and support it.

7.0 References:

1. [Gopher91] University of Minnesota Gopher Team, (gopher@boombox.micro.umn.edu) The Internet Gopher Protocol {URL=gopher://boombox.micro.umn.edu:70/0/0/gopher/gopher_protocol/}
2. [Gopher92] University of Minnesota Gopher Team, (gopher@boombox.micro.umn.edu) Gopher+, proposed enhancements to the internet Gopher protocol {URL=gopher://boombox.micro.umn.edu:70/0/0/gopher/gopher_protocol/}
3. [RFC1436] University of Minnesota Gopher Team, (gopher@boombox.micro.umn.edu) The Internet Gopher Protocol
4. [Lindner93] Lindner, P. (lindner@boombox.micro.umn.edu), Gopher and Relational Databases, an Interface to Metalbase. Proceedings of GopherCon '93
5. [Peppler92] Michael Peppler (mpeppler@itf.ch), sybperl - Sybase DB-library extensions to Perl,. USENET posting in comp.sources.misc {URL=ftp://ftp.uu.net/usenet/comp.sources.misc/volume30/sybperl}
6. [Stock93] Kevin Stock (kstock@encore.com), Oraperl-v2 - Extensions to Perl to access Oracle databases. USENET posting in comp.sources.misc {URL=ftp://ftp.uu.net/usenet/comp.sources.misc/volume38/oraperl-v2/}