Programmer's Guide

ProSQL

Simple SQL Access to
Progress RDBMS

Written by

Scott Auge

sauge@amduus.com

Table of Contents

Introduction....................3
License....................4
Architecture....................5
The Protocol....................6
The Logfile....................7
About Transactions....................8
The set of processes and their communications....................8
Installing....................10
About the distribution file....................10
Procedure....................10
Configuration....................12
Programmer Notes....................13
main.p....................13
delete.p 13
update.p....................13
select.p....................13
insert.p....................13
objlog.p....................14

Introduction

Often I have needed to integrate Progress Databases with other applications. These applications have a horrible time talking to Progress. Yes, I know, "But, but, but there is the ODBC and JDBC connections!"

Well, that is nice if you are on Windows. I am on UNIX or some non-Progress supported operating system or using some form of middle-ware (read that as PHP, Perl, etc.) to make the integration with. Sometimes it has ODBC, sometimes it doesn't.

Sometimes it is a device that needs to interact with Progress DBs and the infrastructure just is not there.

I am also usually on a budget (hence no message-ware.)

If all you want to be able to do is open a connection to the database, throw some SQL statements to the thing, get some results and error information, and be done with it – then this is the software for you.

Hence this tool's creation. It most certainly can be improved upon – but it **IS functional.**

There is a training video you can order from Scott Auge.
It costs $20.00.

It is open source licensed under the BSD license. Basically do as you please, you you need to give me credit for the base code and indemnify me from any legal troubles.

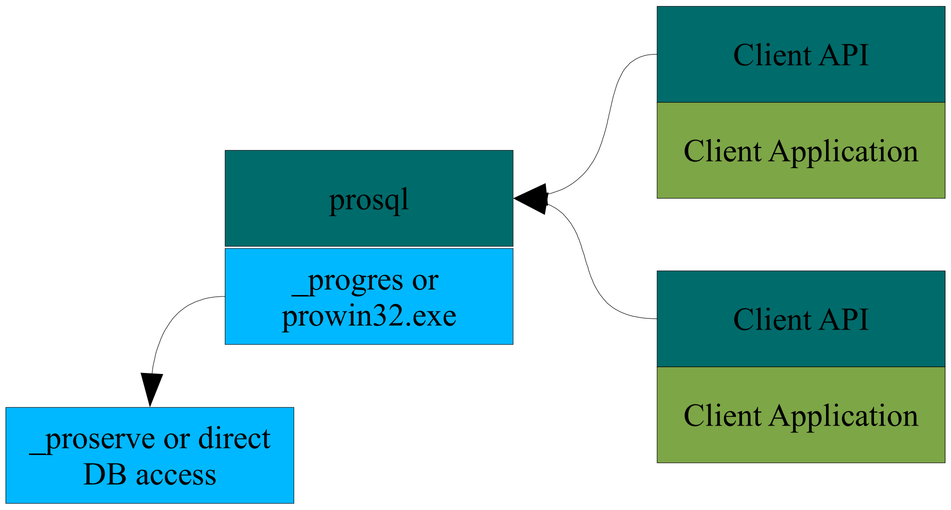
If you have bug fixes or features to add, send em along to me at sauge@amduus.com.

License

Architecture

This version's architecture is very simple – there is a Progress RDBMS server, a progress client as part of the ProSQL process, and you have socket connections from non-Progress technologies and unsupported devices/technologies to the ProSQL process via TCP/IP.

Yes, the server process can handle multiple connections.



The software was developed on UNIX, but with some tweeks a user should be able to get it running on a Windows computer just fine. Most of the tweeking needs to be in the start up script.

ProSQL is a piece of 4GL code that runs on top of the Progress Client. It is via the

Progress client that we are able to interact with the Progress DB.

The ProSQL program will await socket connections from any technology that can open a socket connection and read or write through it.

Through this socket connection one can throw SQL statements and commands to the ProSQL server via a protocol. The good thing about this protocol – is that I am making it public – unlike the other protocols that are proprietary or hidden.

This means the developer in the other technology merely needs to write routines to interact with ProSQL in the protocol it expects to speak in.

Keep aware of your progress licensing obligations regarding user counts.

The Protocol

This system has a protocol that is very XML like. In fact, I hope to mold the system into an SQL web service of sorts. By using XML, hopefully we can change throughout the versions without breaking anything by adding new nodes to the document.

The ProSQL server will denote a message when it is ready for additional processing. Commands and SQL statements are placed into the system between the tags <cmd> and </cmd> such as below:

```
<cmd>INSERT INTO table (a,b) VALUES (1,'this')</cmd>
```

After each command you need to send a CR.

The server is administered by certain commands:

- STOP – Will shutdown the server. Be sure to let other's know you are going to shut it down!
- CLOSE – Will close your connection to the server but leave it running.

The following SQL statements are implemented:

- SELECT
- DELETE
- UPDATE
- INSERT

The DELETE statement is pretty stable and complete. Deletes are noted in the log file, but currently the protocol doesn't mention how many records were effected or any errors.

The UPDATE statement is pretty stable and complete. Updates are noted in the log file, but currently the protocol doesn't mention how many records were effected or any errors.

The INSERT statement is pretty stable and complete. Inserts are noted in the log file, but currently the protocol doesn't mention how many records were effected or any errors.

The SELECT statement is very immature. You can basically specify the fields from a <u>single table</u> with a where clause. You must be careful about your spaces as a simple parser using ENTRY is used to chop up the statement. It sees things as this:

The parser is pretty finiky about how a statement should be formed. A lot of times spaces are used to seperate out various parts of the statement. Just remember there is no yacc or lex for the Progress 4GL yet so these parsers are pretty hand crafted.

SELECT fieldlist FROM table WHERE

using the spaces to separate out the elements.

You can use Progress' functions in the Where clauses, but you should be pretty literal in your data updates.

Here is an example conversation between a telnet user and the ProSQL server.

```
172:~ scottauge$ telnet amduus2 3300
Trying 172.16.1.36...
Connected to amduus2.
Escape character is '^]'.
ProSQL Version %BUILD% Waiting
<cmd>select firstname from contact</cmd>
START RESULTS
<row>
<column name="FirstName">scott'</column>
</row>
END RESULTS
ProSQL Version %BUILD% Waiting
<cmd>insert into contact (firstname) values ('so and so')</cmd>
ProSQL Version %BUILD% Waiting
<cmd>delete from contact where firstname = "so and so"</cmd>
ProSQL Version %BUILD% Waiting
<cmd>update  contact  set  firstname='scott'  where  firstname  =
"scott'"</cmd>
ProSQL Version %BUILD% Waiting
<cmd>select firstname from contact</cmd>
START RESULTS
<row>
<column name="FirstName">scott</column>
</row>
```

```
END RESULTS
ProSQL Version %BUILD% Waiting
<cmd>close</cmd>
Goodbye!
Connection closed by foreign host.
172:~ scottauge$
```

Hopefully this shows how easy it is to work with the connectivity of the ProSQL software.

In future versions,

The Logfile

You specify the log file in the prosql startup command. See the administration area for how to start up and configure ProSQL.

You can also specify the log level the ProSQL server should run under. The following describe the log levels:

Log Level	***Description***
0	No Logging is done.
1	Start, connection, disconnection, and shutdowns are logged.
2	Database records and manipulations are logged
3	SQL parsing routines are logged.
4	Received data is logged

All the lower log levels are included in the higher log levels. So for example, if you run at log level three, you will also receive the information in log levels one and two.

About Transactions

Currently transactions are done in one call to the database. In the future, transactions and locking will hopefully be achieved across multiple calls. For now, it is one statement committed. If it is that important to you, then you will want to invest in the more commercial applications. This system is more oriented to making Progress data available to other applications.

Installing

Installation is simple and straight forward.

About the distribution file

Amduus uses a build system where packages are named as:

build.yyyyjjjhhmmss.zip

where

yyyy is the year of the build in 4 digits of the build time
jjj is the day of the year in three digits of the build time
hh is the hour of the build time
mm is the minute of the build time
ss is the seconds of the build time

The newer the date on the version, the later the version of the software you are using.

Procedure

1. Obtain the proper build from your application supplier or from Amduus Information Works, Inc. at http://amduus.com/OpenSrc/SrcLib/ProSQL/

2. Choose a directory to put the package into.

3. Unzip the package

4. You have an installation now!

Onwards to the configuration.

Configuration

Configuration of the ProSQL server is done in two manners.

The basics of connecting the software to a Progress installation is done in the set.bash script. One basically needs to point at the correct installation of Progress on the DLC variable.

Example:

```
#!/bin/bash

# Set your progress installation directory here
export DLC=/usr/dlc/91C
```

The details of which pf file to use, administration commands, and which port to listen to for the ProSQL process is done via arguments to the prosql script.

Starting and Stopping ProSQL

Starting the ProSQL server is simple. Follow these steps:

Start your Progress RDBMS server on a database.
Execute the prosql script – for example

```
    prosql -port 3300 -loglevel 1 -logfile /tmp/prosql.log
```

Once that is done, your ProSQL server should be running.

Top stop the ProSQL server, you can either kill it lightly with a HUP signal or telnet into the port and issue the command:

```
<cmd>stop</cmd>
```

Which should shutdown the process. If this does not shut down the process, proceed with your normal Progress client shutdown procedures.

Programmer Notes

main.p

This routine performs:

- the listening for socket connections
- will determine type of SQL statement and call the appropriate subroutine
- listen for shutdown requests
- interpret command line arguments for configuration of the process

delete.p

This routine will parse out the SQL statement DELETE and attempt to implement it.

update.p

This routine will parse out the SQL statement UPDATE and attempt to implement it.

select.p

This routine will parse out the SQL statement SELECT and attempt to implement it.

insert.p

This routine will parse out the SQL statement INSERT and attempt to implement it.

objlog.p

Implements logging routines and entry points. This is actually a persistent procedure.