C-JDBC Tutorial – A quick start

Version 0.4
Date 04/11/05

Authors:
Nicolas Modrzyk (Nicolas.Modrzyk@inrialpes.fr)
Emmanuel Cecchet (Emmanuel.Cecchet@inrialpes.fr)

Table of Contents

Introduction

This is a document to help you learn the basics and get started with the database clustering software called C-JDBC.

To proceed with this tutorial you will only need a Java Virtual Machine compliant to J2SE 1.3 or greater.

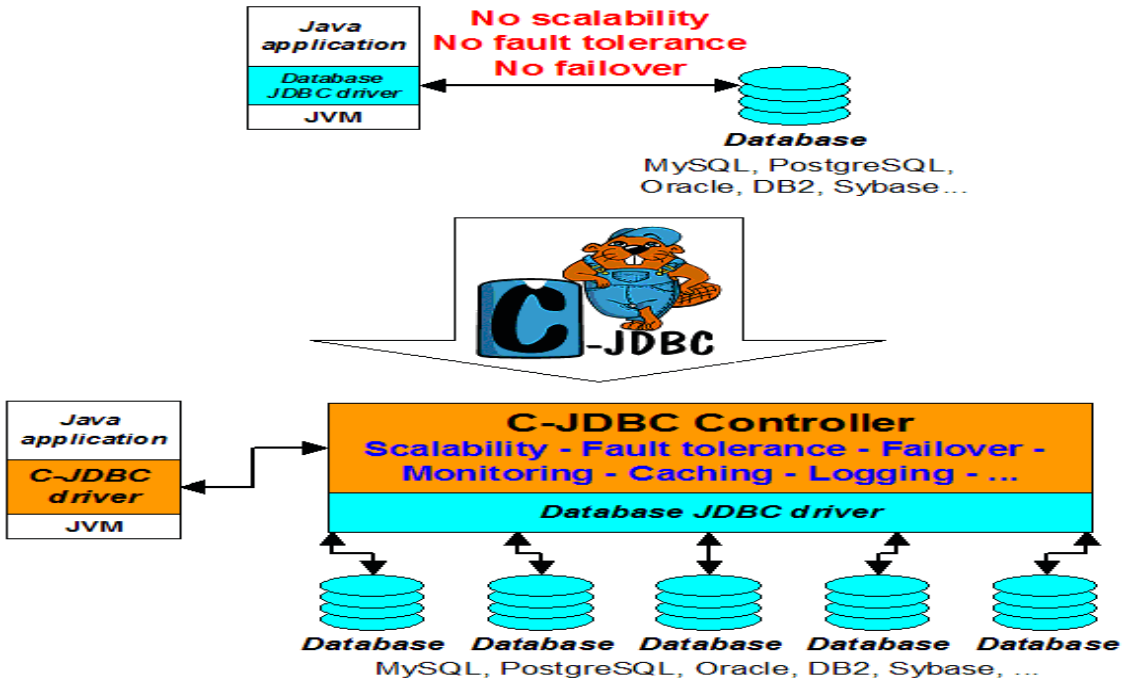
C-JDBC is a database cluster middleware that allows any Java application (standalone application, servlet or EJB container, ...) to transparently access a cluster of databases through JDBC(tm).

You do not have to modify client applications, application servers or database server software. You just have to ensure that all database accesses are performed through JDBC.

C-JDBC provides a flexible architecture that allows you to achieve scalability, high availability and failover with your database tiers. C-JDBC instantiates the concept of RAIDb : Redundant Array of Inexpensive Databases. The database is distributed and replicated among several nodes and C-JDBC load balance the queries between these nodes.

C-JDBC is a *free*, *open source* initiative.

C-JDBC provides a generic JDBC driver to be used by the clients. The client drivers forward the SQL requests to the C-JDBC controller that balances them on a cluster of replicated databases (reads are load balanced and writes are broadcasted). C-JDBC can be used with any RDBMS providing a JDBC driver, that is to say almost all existing open source and commercial databases.



C-JDBC principle

C-JDBC allows to build any cluster configuration including mixing heterogeneous databases. The main features provided by C-JDBC are performance scalability, fault tolerance and high availability. Additional features such as monitoring, logging, SQL requests caching can be provided as well.

The architecture is widely open to allow anyone to plug custom requests schedulers, load balancers, connection managers, caching policies, ...

User application

Recovery log
HSQL 9003

JDBC

JDBC

Backend 1
HSQL 9001

JDBC

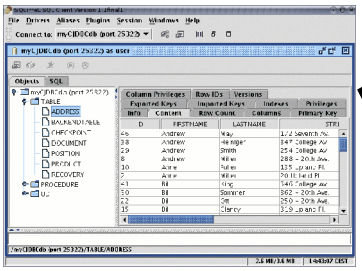
C-JDBC

JDBC

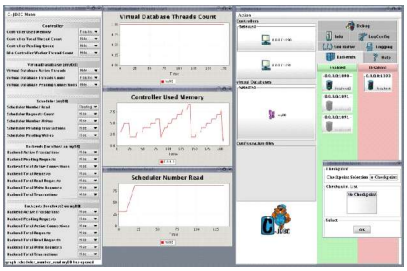
Backend 2
HSQL 9002

JDBC

JMX



Squirrel SQL console



C-JDBC administration console

Getting C-JDBC

You can get the latest release of C-JDBC on the Objectweb forge site:

http://forge.objectweb.org/project/showfiles.php?group_id=42

Choose the latest installer distribution by clicking on the link:

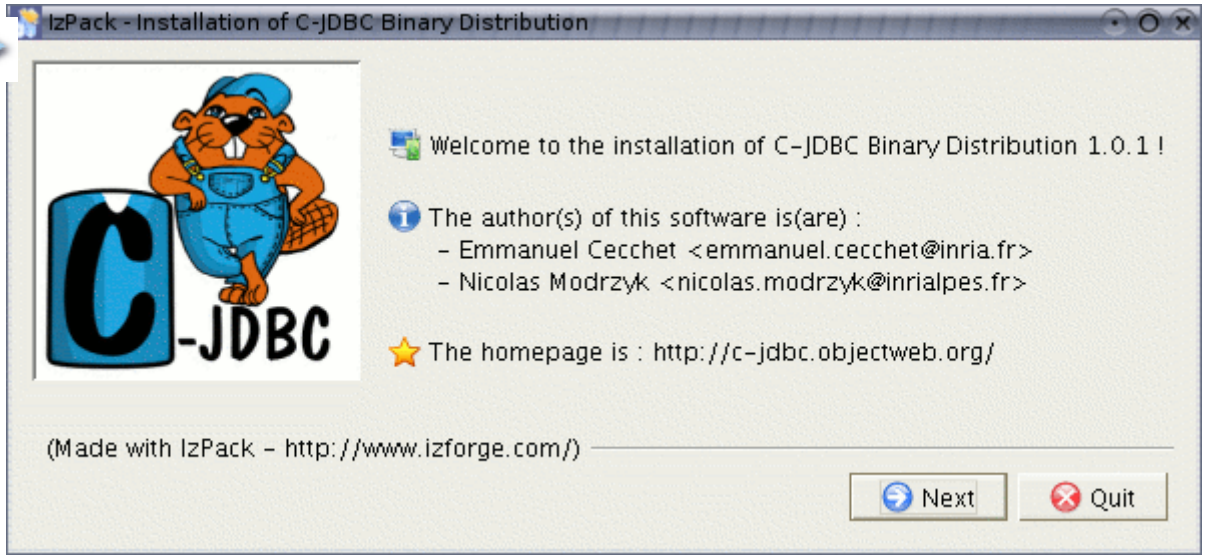
c-jdbc-x.y-bin-installer.jar

where x and y stand for the major and minor version number. In this tutorial, we'll assume we are working with version 1.0.1

When the download is completed, double click on the jar file. If nothing happens, you have to type the command:

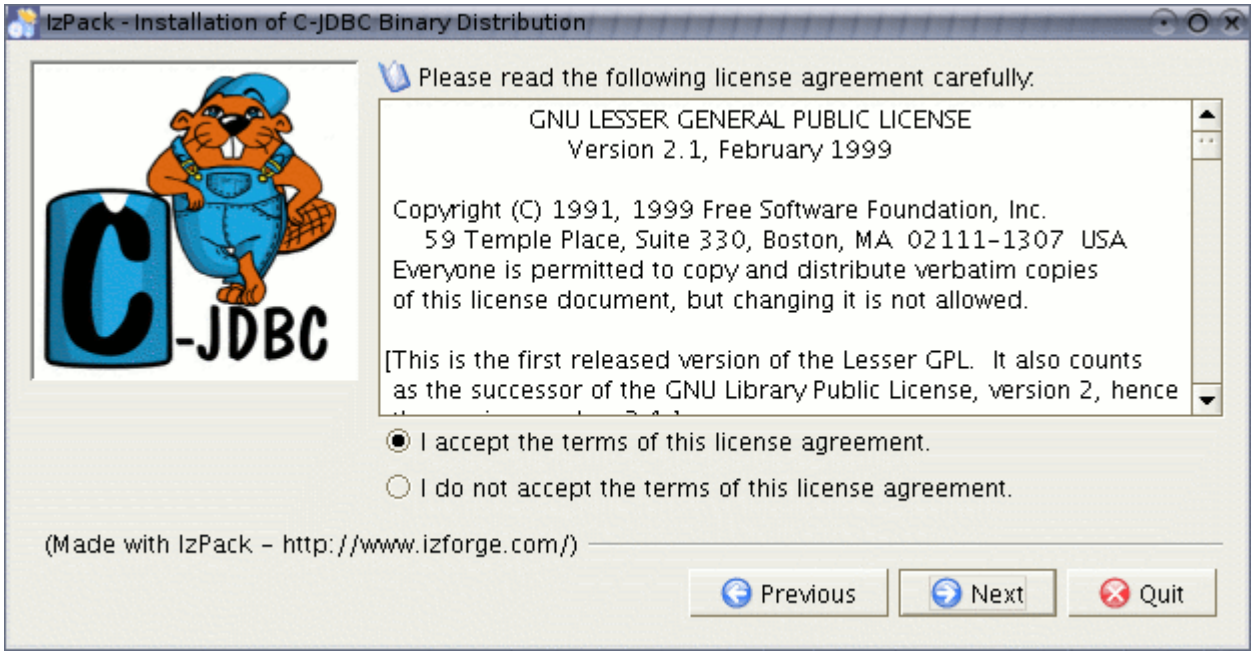
java -jar c-jdbc-1.0-1-bin-installer.jar

Then the installation screen should appear:

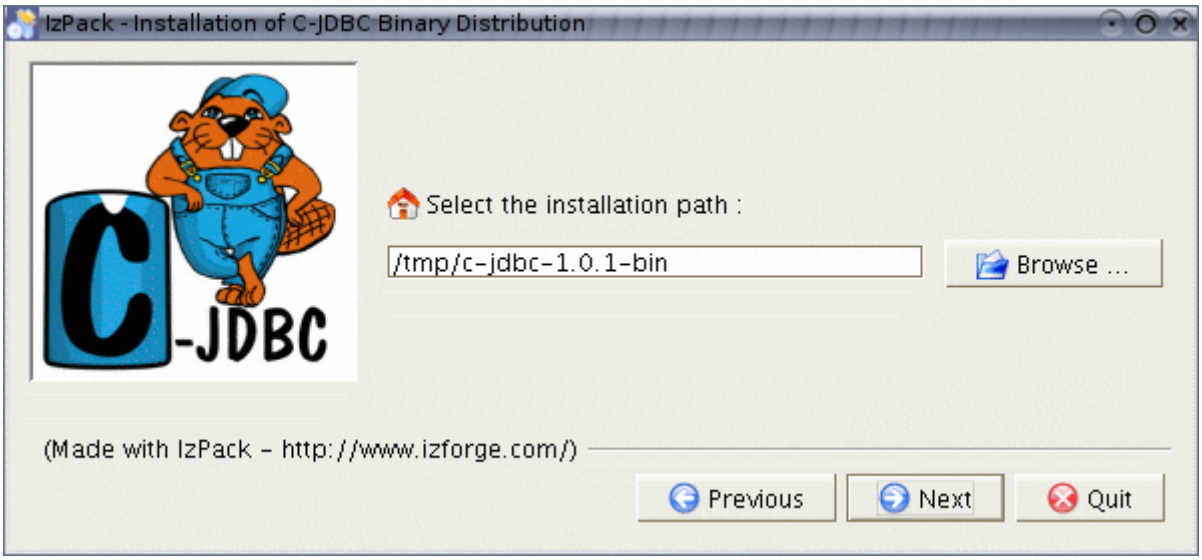


Installing C-JDBC components

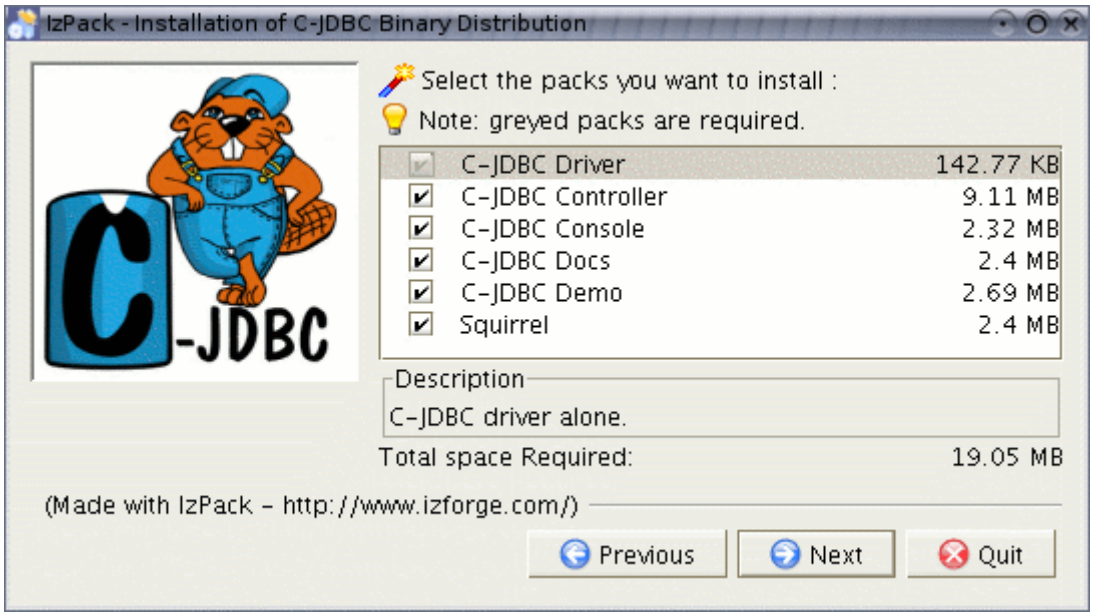
First step is to read and accept the LGPL license used by C-JDBC.



Then choose the place where you want all the files to be installed. The directory will be created if it does not exist. All the files will be overwritten, if it was previously created.



Next step is to select the different components to be installed:



The different components are:

- **C-JDBC Driver**

  This is the JDBC driver that will be used by the client application talking to the database. (ie: Tomcat, JOnAS,JBOSS,...)

- **C-JDBC Controller**

  This is the main component of C-JDBC. This is the C-JDBC controller that will handle and process requests coming from the C-JDBC driver.

- **C-JDBC Administration Console**

  To administrate, monitor, view statistics, and recover failed database backends, you can use a graphical interface, or just a regular text console.

- **C-JDBC Documentation**

  Contains the C-JDBC reference documentation including an extensive description of how C-JDBC is working, and how to configure it to the needs of the application.

- **C-JDBC Demo**

  A set of demo files to easily set up replication on HypersonicSQL backends. If you do not select this package, HypersonicSQL will not be installed.

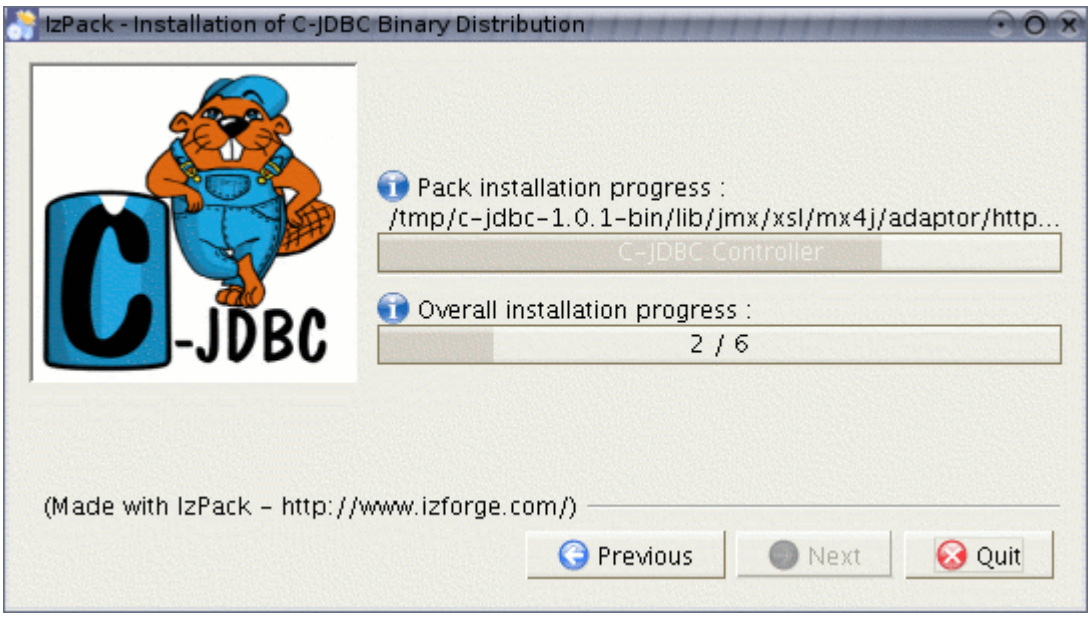
- **SquirrelSQL**

  This is a graphical SQL Console client. You can easily check the content and metadata of any database table. Its main purpose is to execute SQL requests inside or outside transactions.

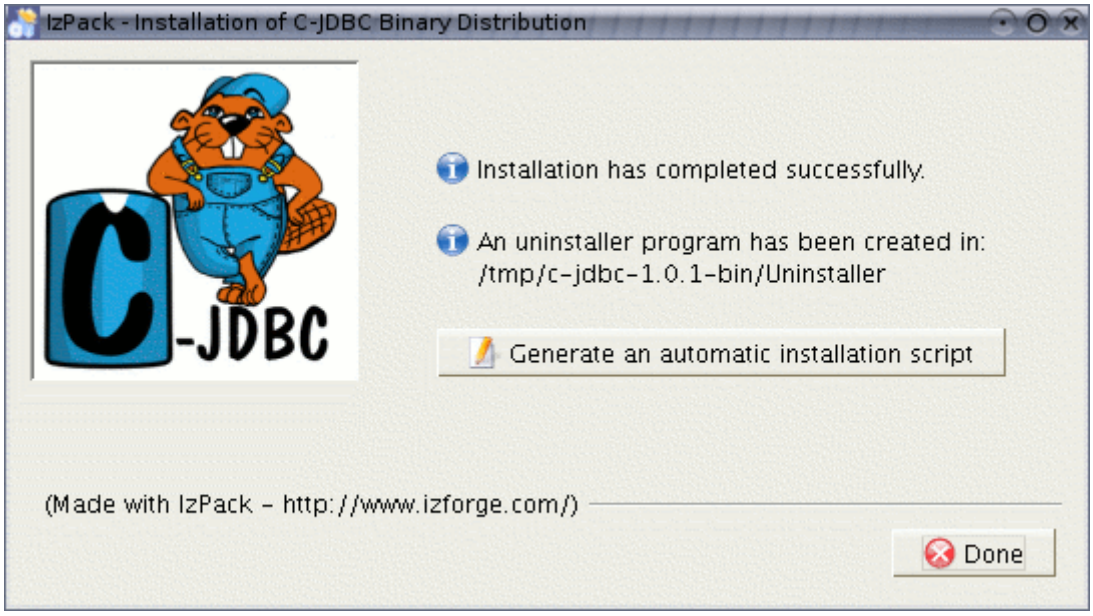
- **iSQL**

  Another graphical SQL Console client.

For the tutorial, we recommend installing all the components, as we will describe them all in the next parts.

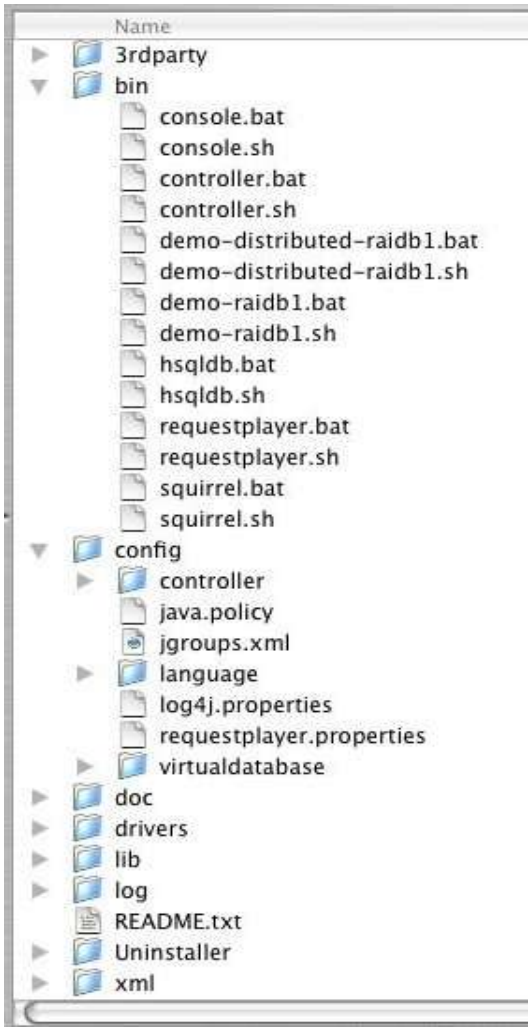


The installation proceeds and then we are done with the installer. It also generates an uninstaller script to easily clean up files.



A quick tour of the C-JDBC directory

After installation, this is the content of the new directory:



3rdParty

This is where HypersonicSQL and Squirrel libraries can be found

bin

All the scripts to start/stop applications are here

demo

All the scripts for the examples are here

config

Location of the different configuration files. (Controller, VirtualDatabase, loggers, language files ...)

doc

The user guide and other configuration example files can be found here

drivers

The C-JDBC driver can be found here. At runtime, other drivers can be unjar in this directory to be found by the java classloader.

lib

All the java libraries used by C-JDBC

log

The logs generated by the logging system can be found here. Also the C-JDBC report will be generated in this folder

Uninstaller

Scripts to remove all the files from this installation.

xml

The DTD and XSL files used by C-JDBC. The two main DTDs can be found there to validate controller configuration files and virtual database configuration files.

Here we will give a short description of the different scripts that can be found in the bin directory of the installation.

- **C-JDBC script files:**

console.bat/.sh: This is to start the administration to administrate the controller. If you have a graphic environment, it will start a swing GUI, otherwise, it will start a text base console.

controller.bat/..sh: This starts the controller with a default configuration file: $*CJDBC_HOME/config/controller/controller.xml*

requestplayer.bat/.sh: This is used to replay a session previously recorded by the controller.

wizard.bat/.sh: This tool is still under development but it allows you to generate virtual database configuration files using a graphical user interface.

- **Demo Files:**

demo-distributed-raidb1-controller1.bat/.sh: This starts the distributed 1st controller for the distributed demo.

demo-distributed-raidb1-controller2.bat/.sh: This starts the second controller demo for the distributed demo.

demo-raidb1.bat/.sh: This is another demo. It will start a single controller with two hypersonic backends.

- **Third Party Software Files:**

isql.bat/.sh A graphical SQL client

squirrel.bat/.sh: The graphical SQL client

hsqldb.bat/.sh: This starts a simple hsqldb DBMS database. It is a lightweight java sql server. It is used for the demos presented in this tutorial.

Description of the RAIDb1 Demo

The raidb1 demo will start a total of three HypersonicSQL database servers.

1. A first backend to use as the main database, it will be named “localhost” in the tutorial
2. A second backend, to use as a failover database, it will be named “localhost2” in the tutorial
3. A third backend, used to store recovery information and to log recovery requests.

When the three databases have been started, the demo will start a controller on port 25323, and will automatically load a virtual database, named “myDB” that will have a schema identical to the two backends.

This is a RAIDb1 replication scheme, meaning that all the backends have the exact same data.

The schema is defined by the following create statements:

```
CREATE TABLE ADDRESS(
ID INTEGER NOT NULL PRIMARY KEY,
FIRSTNAME VARCHAR(255),
LASTNAME VARCHAR(255),
STREET VARCHAR(255),
CITY VARCHAR(255))

CREATE TABLE PRODUCT(
ID INTEGER NOT NULL PRIMARY KEY,
NAME VARCHAR(255),
COST DECIMAL)

CREATE TABLE DOCUMENT(
ID INTEGER NOT NULL PRIMARY KEY,
ADDRESSID INTEGER,
TOTAL DECIMAL)

CREATE TABLE PPOSITION(
DOCUMENTID INTEGER NOT NULL,
PPOSITION INTEGER NOT NULL,
PRODUCTID INTEGER,QUANTITY INTEGER,
PRICE DECIMAL)
```

While using squirrel, connecting to the C-JDBC controller, you will have a database corresponding to this view.

You can connect to the controller, from the C-JDBC driver using the following URL:

jdbc:cjdbc://localhost/myDB

Description of the distributed-raidb1 Demo

The distributed-raidb1 demo will start a total of four HypersonicSQL database servers.

1. Two backends for one controller, on port 25322.
2. Two other backends for another controller, on port 25323.

There is no recovery log defined in the distributed demo, but you can easily add one by copying the simple raidb1 demo files for each controller.

The demo will then start the two controllers, and in the output log, you will see a group communication will be established between the two.

The database schema of the cluster is the same as the one defined in the RAIDb1 demo, and still we are in a RAIDb1 distribution scheme, so all the four backends, will have the same data.

You can access the cluster with the following URL:

jdbc:cjdbc://localhost:25322,localhost:25323/myDB

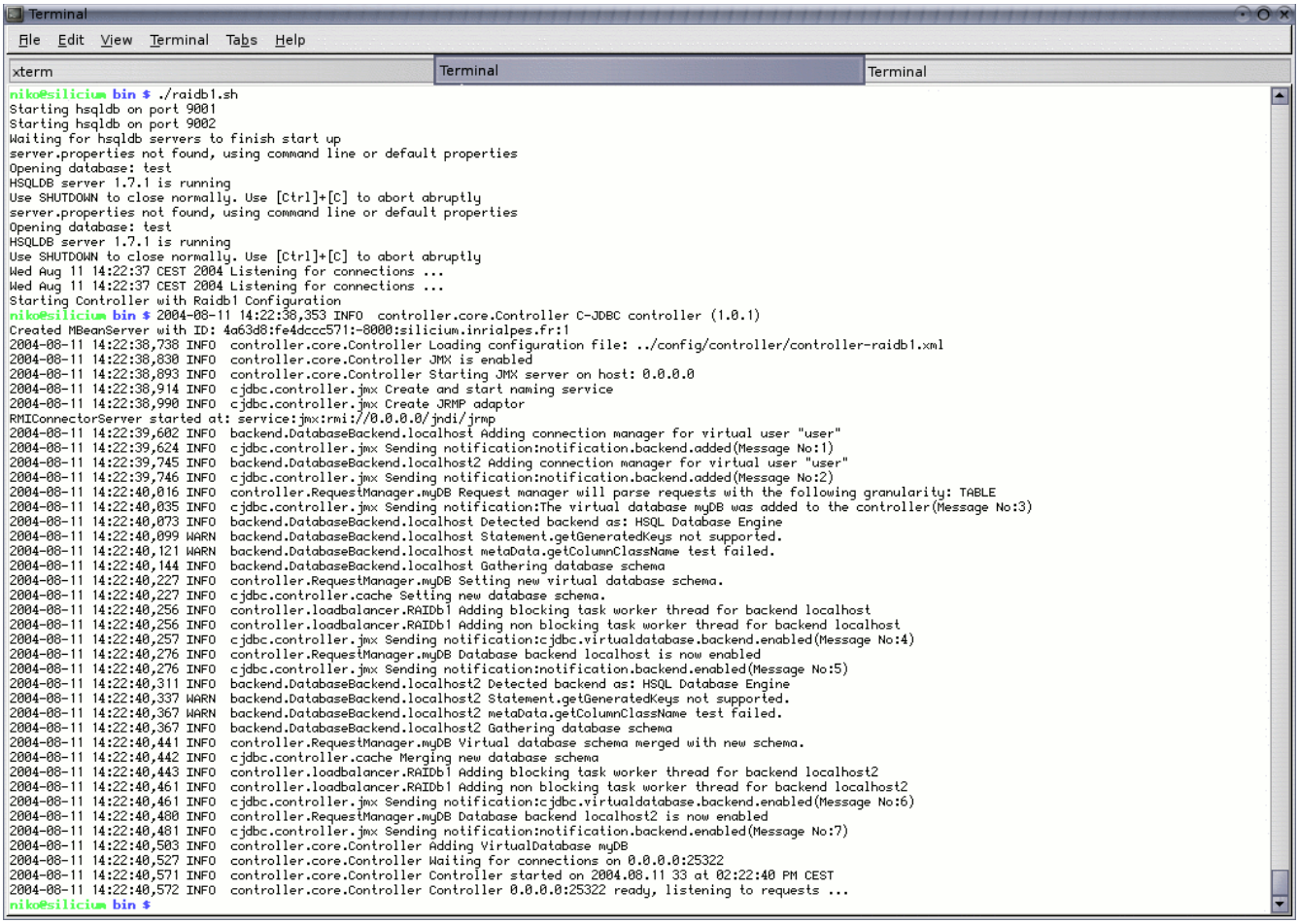
The simple URL described above will also work (jdbc:cjdbc://localhost/myDB) as the driver will find by itself a cluster has been formed.

Start and play with the RAIDb1 demo

Starting the demo

Let's execute the script called demo-raidb1.sh (under Unix) or demo-raidb1.bat (under windows), located in the $CJDBC_HOME/demo directory.

If you have a shell window open, this should have the following output:

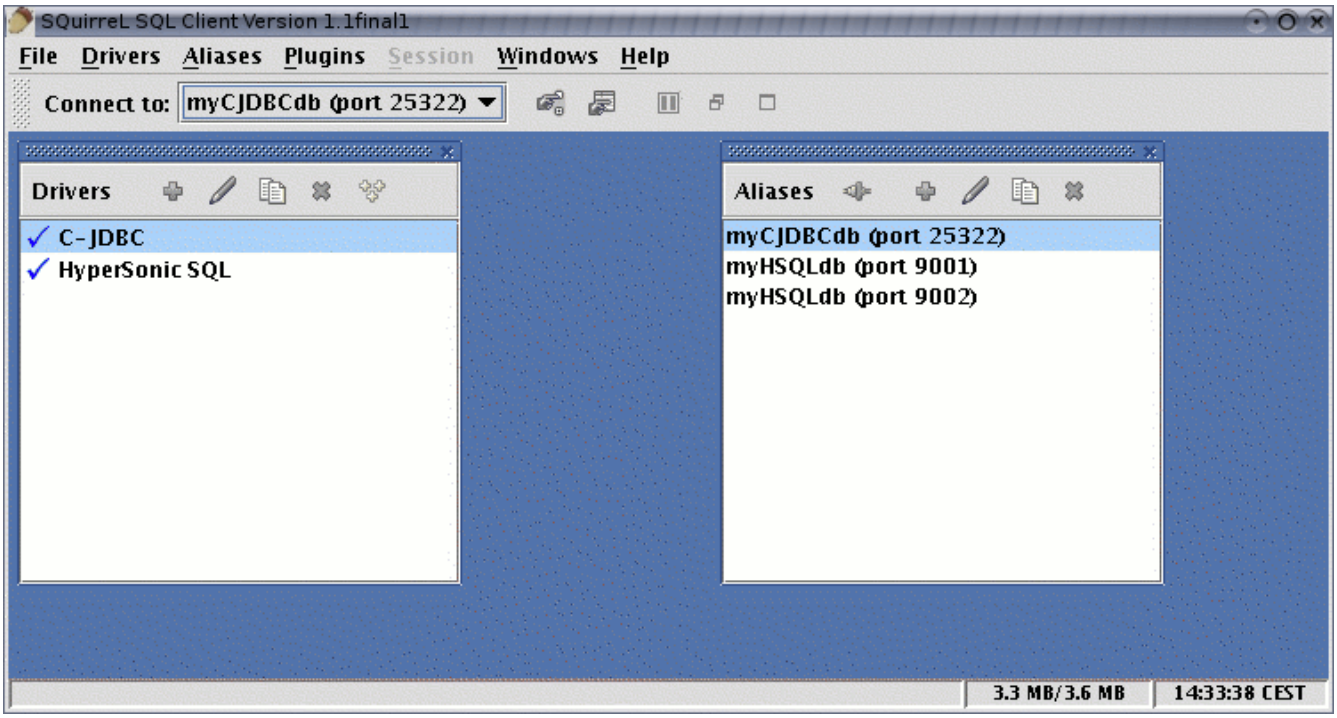


This effectively starts 2 databases HypersonicSQL and a C-JDBC controller. The two databases will be managed and accessed via the C-JDBC controller.

Note: Under windows, you have to press a key once the two HypersonicSQL backends have displayed “Listening ...”.

Playing with SquirrelSQL, the SQL console

Now we can start squirrel, using the script (squirrel.sh/.bat found in the bin directory), and we should quickly get this:



Squirrel with C-JDBC is configured with the C-JDBC driver and the HypersonicSQL drivers.
There are three aliases available straight away:

myCJDBCdb(port 25322)

This is the database that the application will see. To connect to it, the application will use the C-JDBC driver.

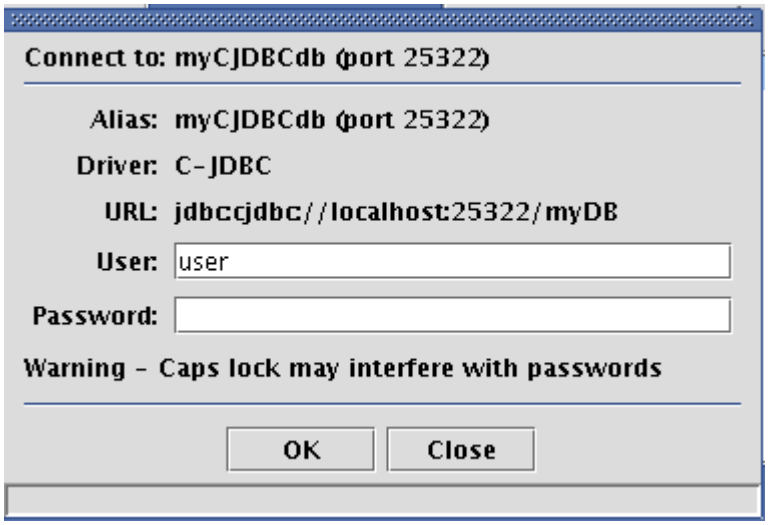
myHSQLdb (port9001)

This is one of the two backends that the controller manages.

myHSQLDB(port9002)

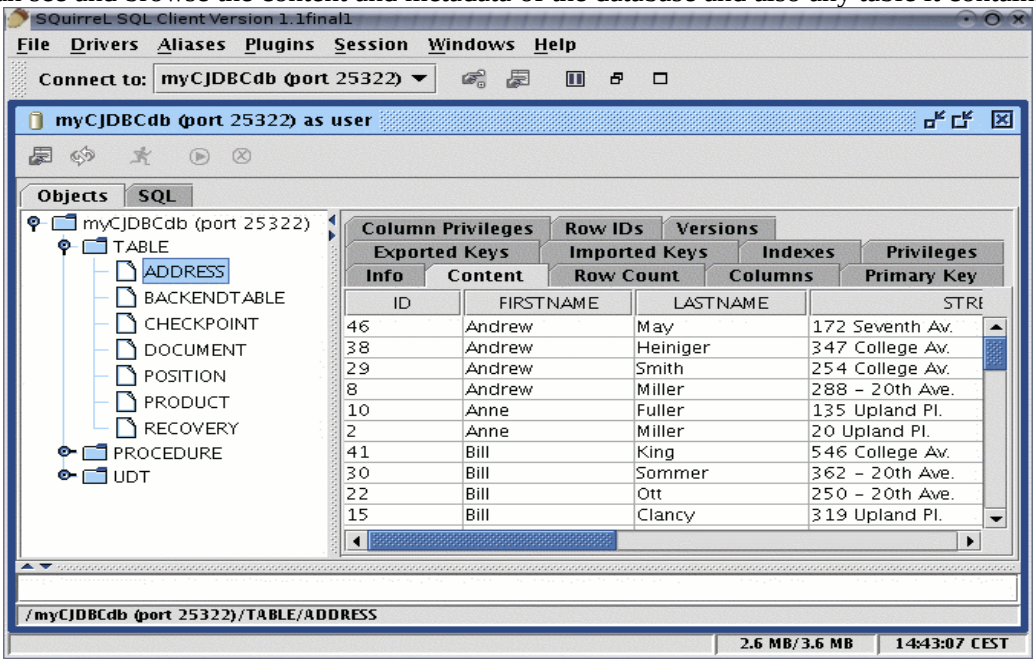
This is the replication of the above database.

Double clicking on any of the alias, will pop up a window for authentication:

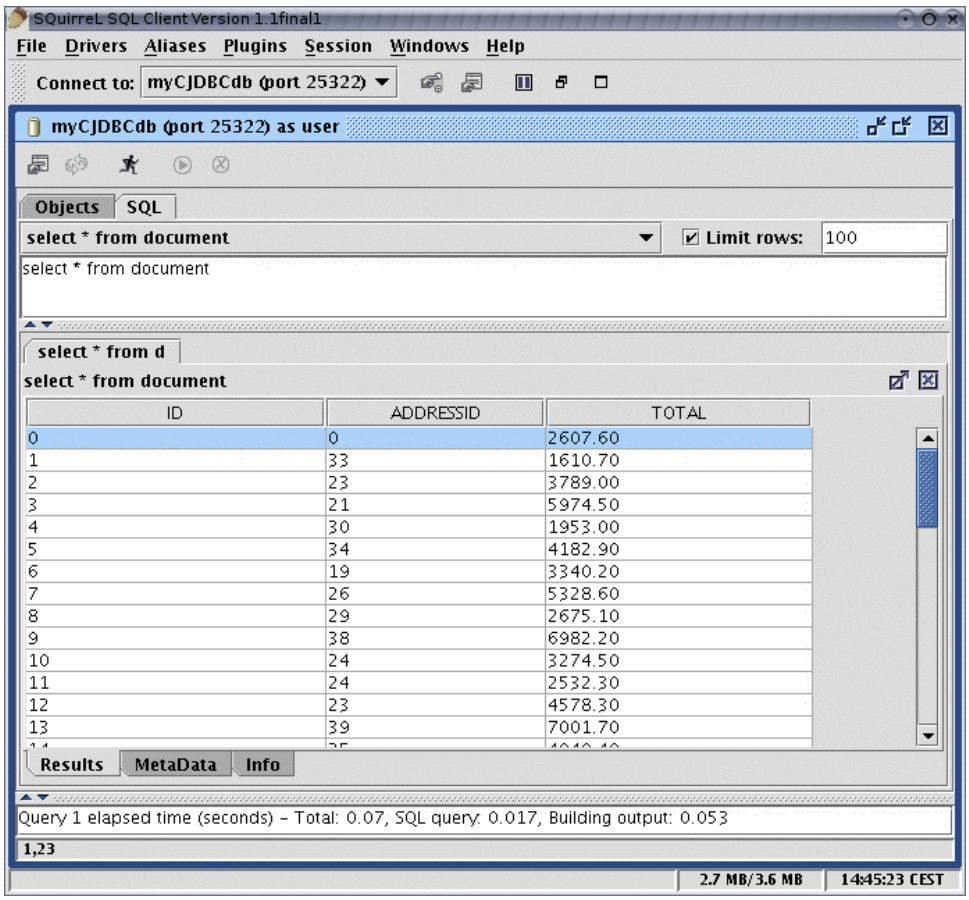


All the fields have been prefilled, so just click OK there.

Now we can see and browse the content and metadata of the database and also any table it contains.



Clicking on the SQL panel allows to access the SQL console itself from where we can execute queries. Just type the query you want in the text panel and click on the running man.

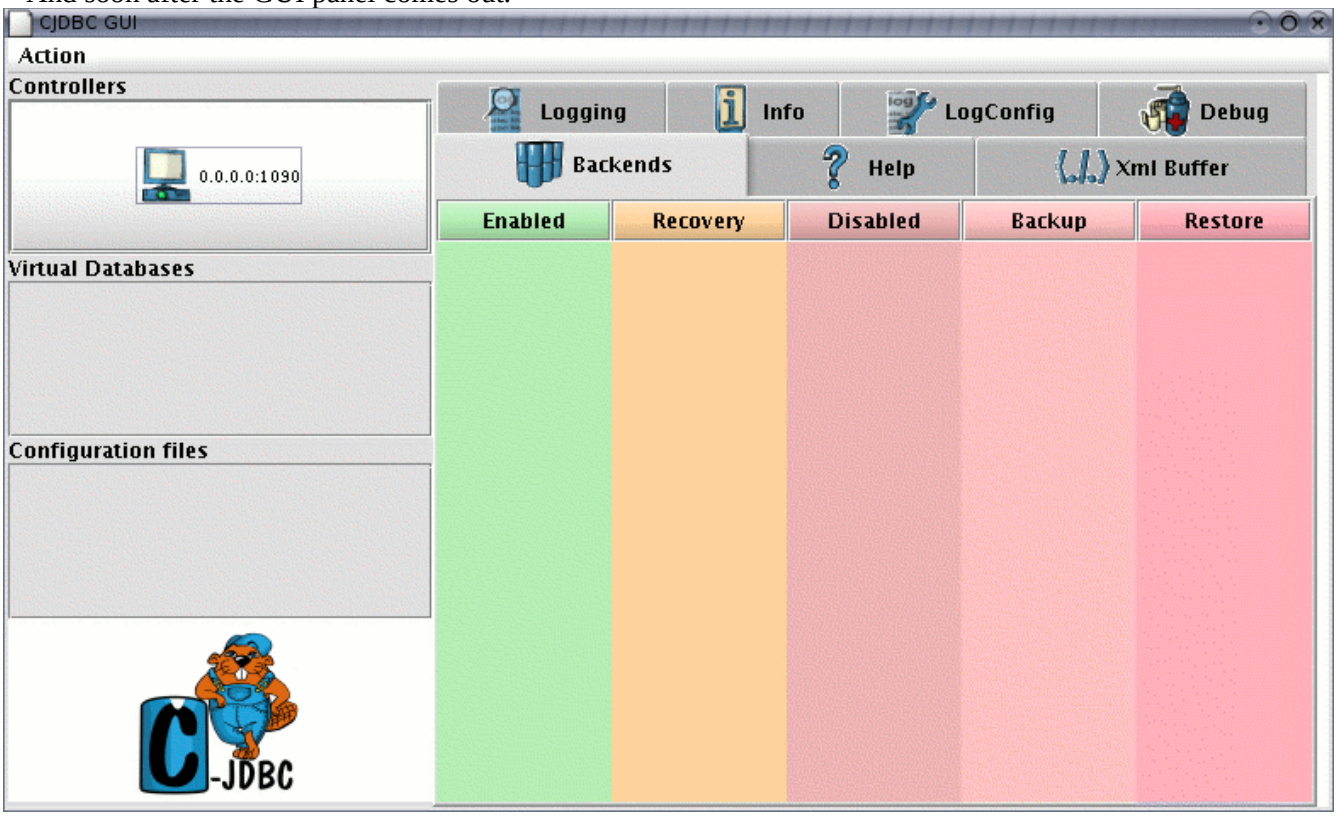

SQuirreL SQL Client Version 1.1final1
File Drivers Aliases Plugins Session Windows Help
Connect to: myCJDBCdb (port 25322)
myCJDBCdb (port 25322) as user
Objects SQL
select * from document
Limit rows: 100
select * from document
select * from d
select * from document
ID ADDRESSID TOTAL
0 0 2607.60
1 33 1610.70
2 23 3789.00
3 21 5974.50
4 30 1953.00
5 34 4182.90
6 19 3340.20
7 26 5328.60
8 29 2675.10
9 38 6982.20
10 24 3274.50
11 24 2532.30
12 23 4578.30
13 39 7001.70
Results MetaData Info
Query 1 elapsed time (seconds) - Total: 0.07, SQL query: 0.017, Building output: 0.053
1,23
2.7 MB/3.6 MB
14:45:23 CEST

First steps with the Graphical Administration Console

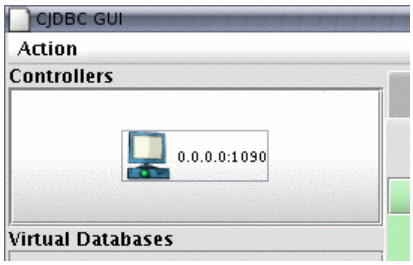
Starting the GUI

To start the GUI, we use the gui.sh/.bat script file.

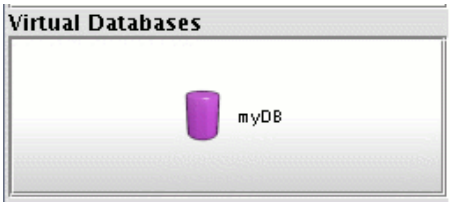
And soon after the GUI panel comes out.



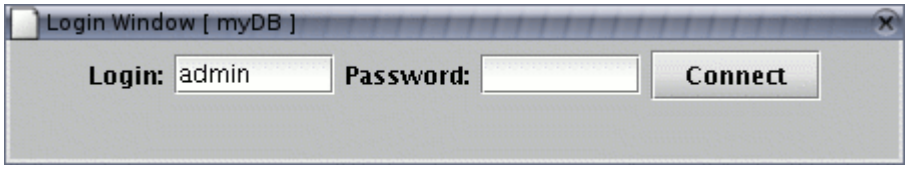
There's not much we can do if we don't have a virtual database to work with.Let's click on the controller on the top left pannel:



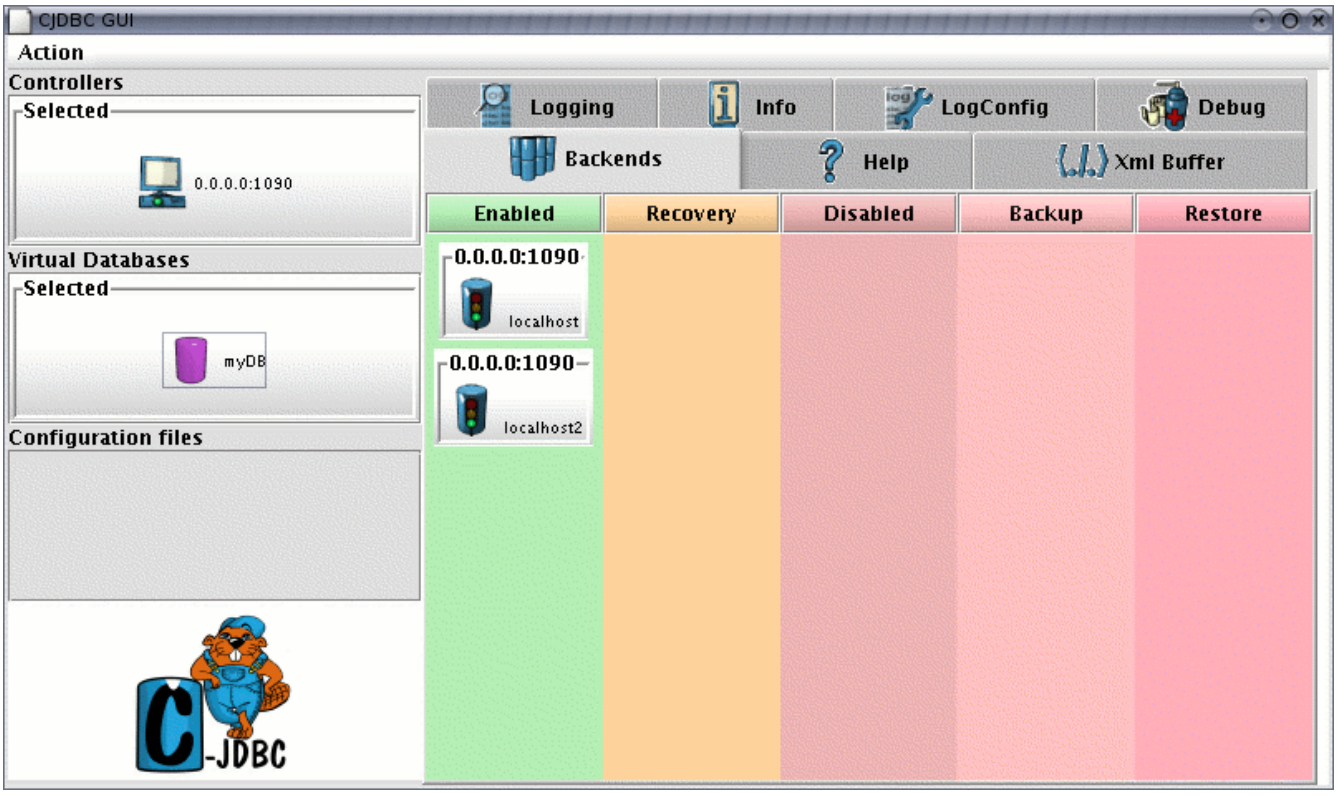
A virtual database will appear in the virtual database panel:



Once, we click on the virtual database icon, an authentication window will appear. As of C-JDBC version 1.2.1, the login is automatic and there is no need to click on the virtual database icon anymore. We have to enter one of the administrator login that has been specified in the configuration file of the virtual database. If you started the raidb1 demo, the login is **admin** with no password.



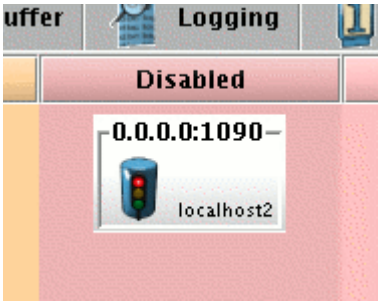
And then the screen will reload itself depending on the configuration of the database.



We can see on the frame, that the two hsqldb backends are enabled and ready to handle requests.

Disabling a backend

If you drag any of the backend icon to the disabled column, a checkpoint will be automatically assigned to it. This will be used later by the recovery service to replay missing updates when putting the backend back in the enabled state.



The backend is going to be disabled, meaning that no more requests will be sent to it from the C-JDBC controller.

Enabling a disabled backend

From the disabled backend, you can play a few requests using squirrel. You could also possibly try to insert some values, and check they still arrive on the hsqldb 'localhost' backend.
Now we want to enable our previously disabled backend 'localhost2'.

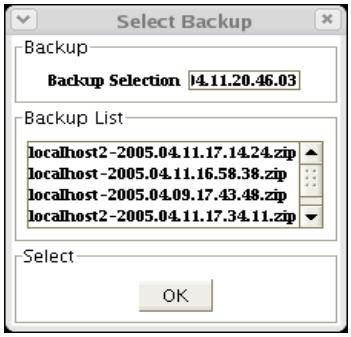
Simply drag and drop the backend from the disabled column to the enabled column.

No need to select a checkpoint, as the last known checkpoint for this backend will be transparently used. After that the backend wil be in a recovery state for a few seconds and its icon will change.

This will actually replay all the request that were missed while the backend was disabled.

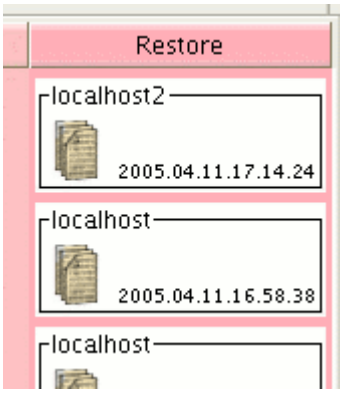
Creating a backup of a backend

The two backends are online again. Let's drag and drop one to the backup column. This will pop up a new window to select a name to give to the dump file that will contain the backup. The default name is made of the backend name along with the current time. We will keep the default name and click on OK.



This will change the Icon of the backend to the following one for the time of the backup process.

Once the backup is finished, a new icon with the dump name will appear in the restore column:



During backup, requests can still be sent to the C-JDBC controller because the backend 'localhost' is still active.

From now on, you can use this dump to add new backends or restore backends that have failed and have come out of sync with the rest of the backends.

Adding a backend to the cluster.

The script to start HypersonicSQL process was: hsqldb.sh/.bat

We will start a new process of these on a new port using the following command:

./hsqldb.sh -port 9010

The command has started a new server process. The output of the command should be:

server.properties not found, using command line or default properties

Opening database: test

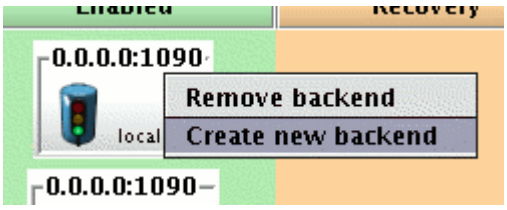
HSQLDB server 1.7.1 is running

Use SHUTDOWN to close normally. Use [Ctrl]+[C] to abort abruptly

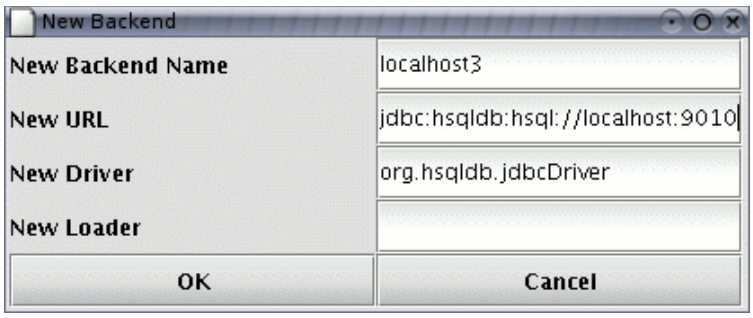
Fri Aug 20 18:50:33 CEST 2004 Listening for connections ...

Now we want to add this backend to the cluster.

If we right click on a backend, we can see a couple of commands. Click on 'Create new backend'



This will pop up a new window with the necessary information to add a new backend to the cluster.



In the field 'New Backend Name', enter the name of the new backend. For example, 'localhost3'.

In the field 'New URL', enter the url to connect to this backend. This should be the same as the other ones except the port that is different. In our example, use: 'jdbc:hsqldb:hsql://localhost:9010' ...

The backend 'localhost3' appears in the column disabled:

We can now drag and drop the backend on the previously created dump file:

And the controller will restore the content of the backup to this new backend.

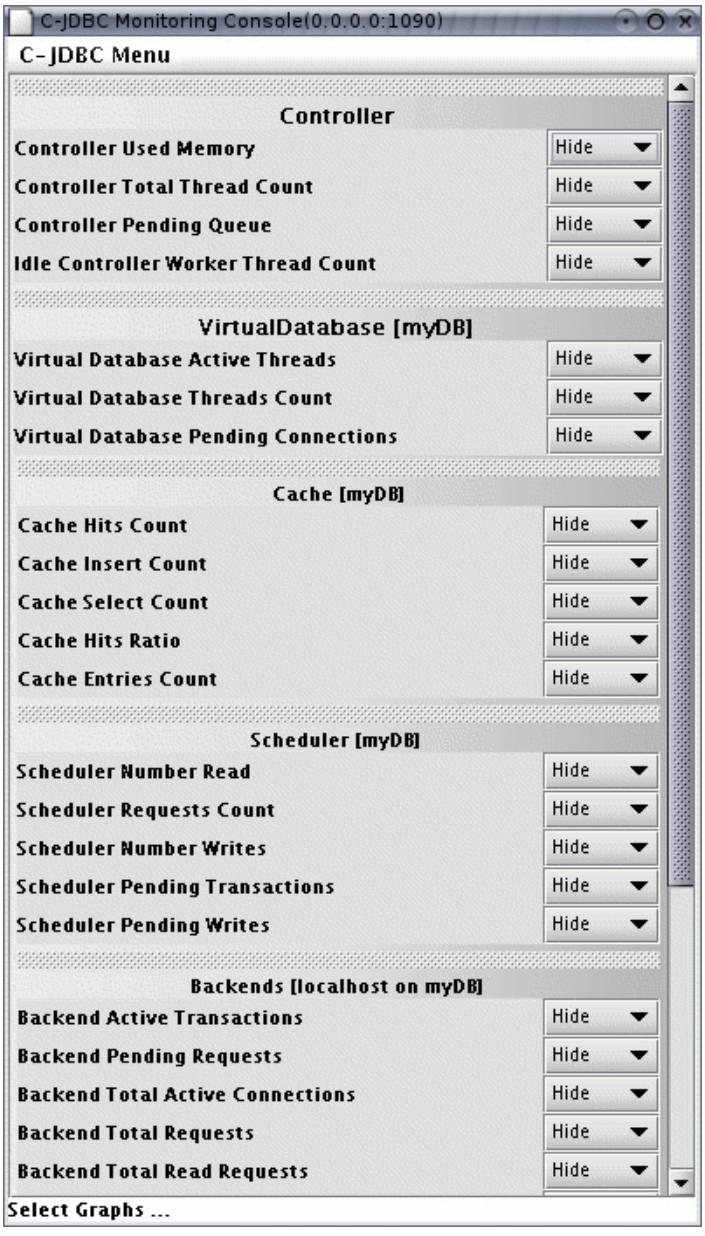
Once the restore process if finished, the backend will be in disabled mode again. We can now safely enable it, the recovery log will replay the missing queries since the dump was made.

We now have a cluster made of three backends. This step can be repeated until the number of necessary backends has been reached.

Monitoring C-JDBC components

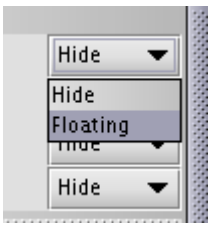
From the above gui, if we right click on the controller, we will get the following contextual menu:

Select Monitor controller and the monitoring frame will appear:

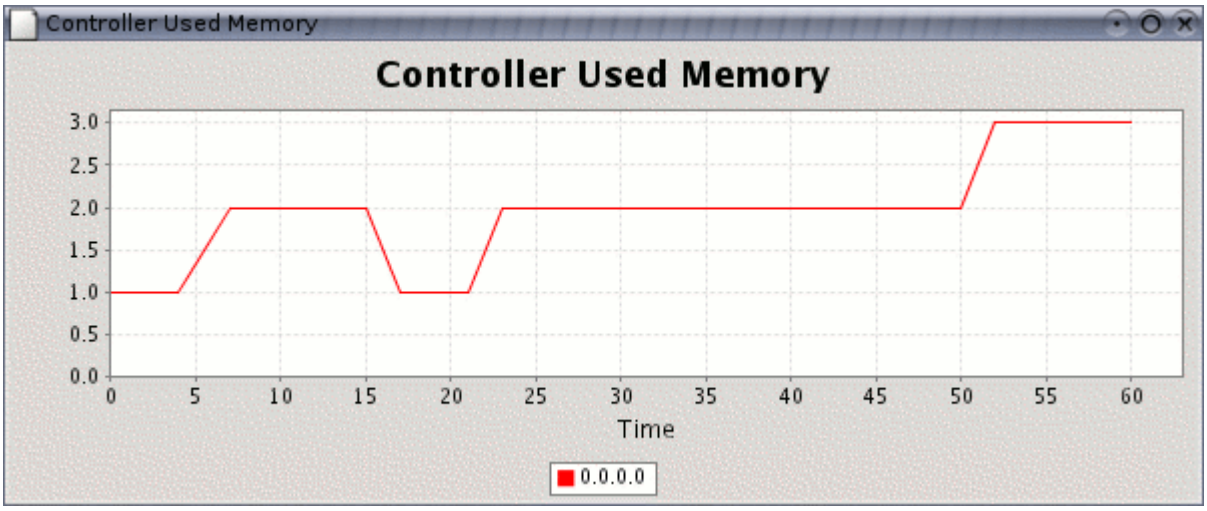


You can then select any of the monitoring window:

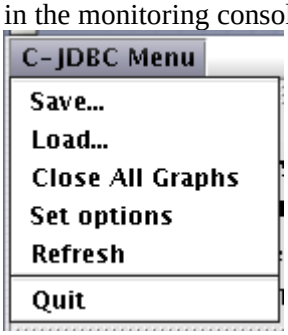
The monitoring is done online, meaning that it will only get information when the window is opened.



Here is an example when monitoring the controller memory usage:



There is a menu with a few useful options in the monitoring console:



Save...

It will save the position of all the monitoring window, so you don't have to open and re-position them one by one after quitting.

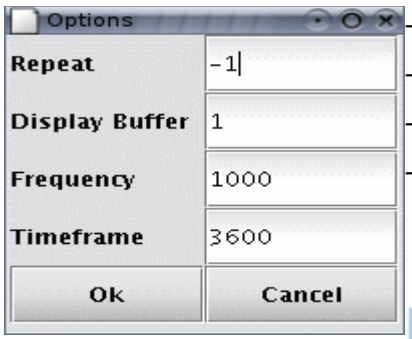
Load...

This restores the positions that were saved before

Close All graphs

Speaks for itself.

Set options. This will let you set a few monitoring options:



--> How many times must we collect data (-1 means for ever ...)

--> How many times we bufferize data before display?

--> Frequency of the data collection. (in milliseconds)

--> Timeframe of the monitoring window. (in milliseconds)

Refresh

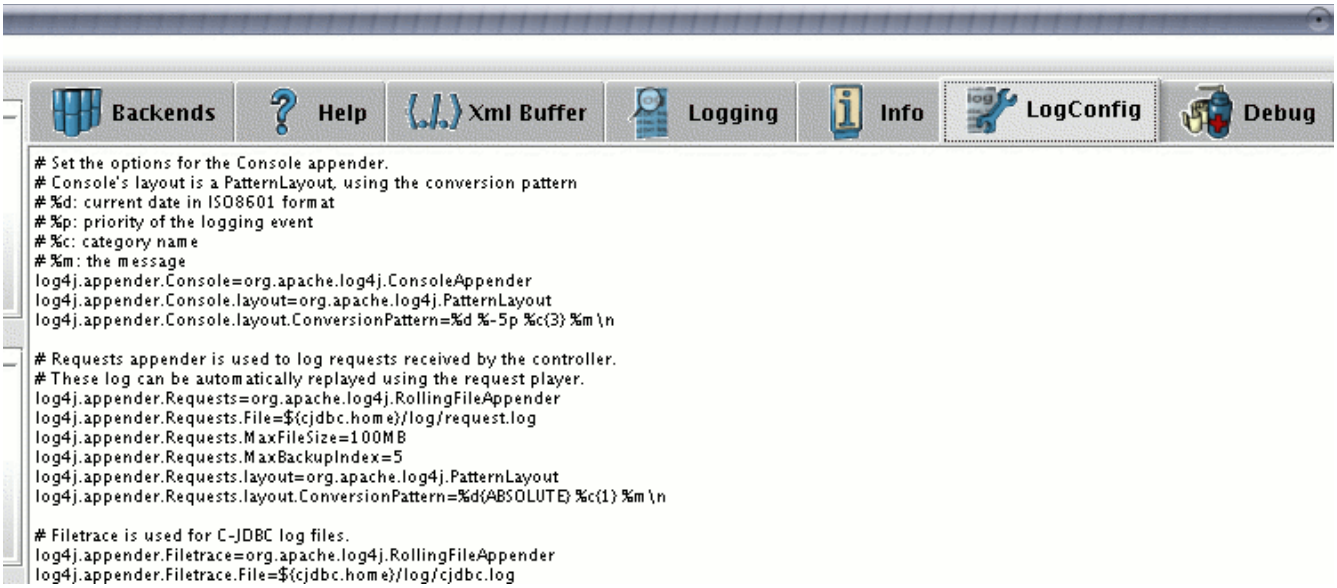
If the configuration of the controller has changed, then you can call this method to reload the backends or virtual databases that have been added or removed.

Generate a bug report

To help the C-JDBC team with debugging you can generate report when you have a problem.

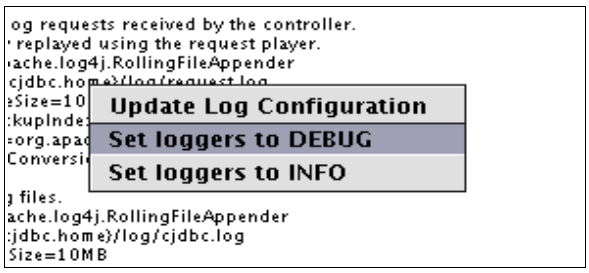
From the GUI, you can always view the log configuration of the controller, by clicking on the view log configuration line in the controller's contextual menu:

This will bring the content of the remote log4j.properties file into the LogConfig panel.



This panel is editable and you can make all the changes you want to the different loggers used in C-JDBC. What we are interested in now, is to set them all to debug,

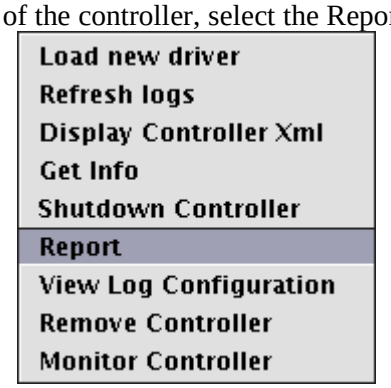
clicking on the proper submenu:



This will change all the loggers on the controller and tell them to emit debugging information.

Now replay the code, or part of the application that was causing the bug on C-JDBC.

Then, always on the contextual menu of the controller, select the Report option:



This will display all the content of the report in the Info panel shown below:

The content of the panel is actually the content of the file cjdbc.report that can be found in the log directory of the C-JDBC installation.

This documents contains:

- java properties (machine, os, jvm version ...)
- controller and virtual database configurations
- all the loggers output

This is what we need when debugging the application in complex situations.

Further readings

All the new documentation can be found online on the C-JDBC website in the documentation section at:

http://c-jdbc.objectweb.org/doc/index.html

The complete user guide can be found at:

http://c-jdbc.objectweb.org/current/doc/userGuide/userGuide.pdf

And a more general presentation of C-JDBC:

http://c-jdbc.objectweb.org/current/doc/C-JDBC_Solutions_Linux_2004.ppt

If you have any question concerning the use of C-JDBC or features you would like to see included, please write an email to:

c-jdbc@objectweb.org

Thanks for reading !