HOWTO use C-JDBC with Apache Derby

Version 0.2
23 November 2004

Author: Emmanuel.Cecchet@inria.fr

There are 2 ways of using C-JDBC with Apache Derby:

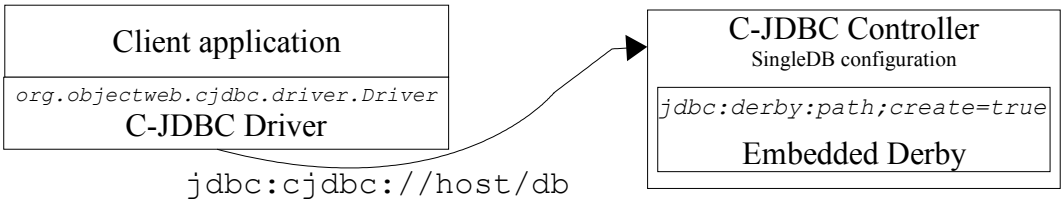
- use the embedded version of Derby started within the C-JDBC controller and use the C-JDBC driver for a multi-user remote access without requiring the use of the closed source DB2 driver.
- use C-JDBC to provide clustering features (performance scalability and high availability) to Derby. This requires remote access to Derby either using the solution proposed above or using Derby with Network server and the IBM DB2 JDBC Universal Driver for Derby.

If you need an introduction to C-JDBC, check the documentation section of the http://c-jdbc.objectweb.org web site.
More information on Derby can be found at http://incubator.apache.org/derby.

Don't hesitate to ask further questions on the c-jdbc@objectweb.org or derby-user@nagoya.apache.org mailing lists.

1. C-JDBC for Derby remote access

This section explains how to use C-JDBC to remotely access Derby. This is an alternative to the Network Server + IBM DB2 JDBC Universal Driver for Derby. The figure below gives an overview of this configuration.



1.1. Client side setting

The client application uses the C-JDBC driver to access Derby. Make sure that c-jdbc-driver.jar is accessible from the classpath. Here is an example of how to load the C-JDBC driver and get a connection:

```
Class.forName("org.objectweb.cjdbc.driver.Driver");
Connection c = DriverManager.getConnection("jdbc:cjdbc://host/db",
"login", "password");
```

1.2. C-JDBC Controller configuration

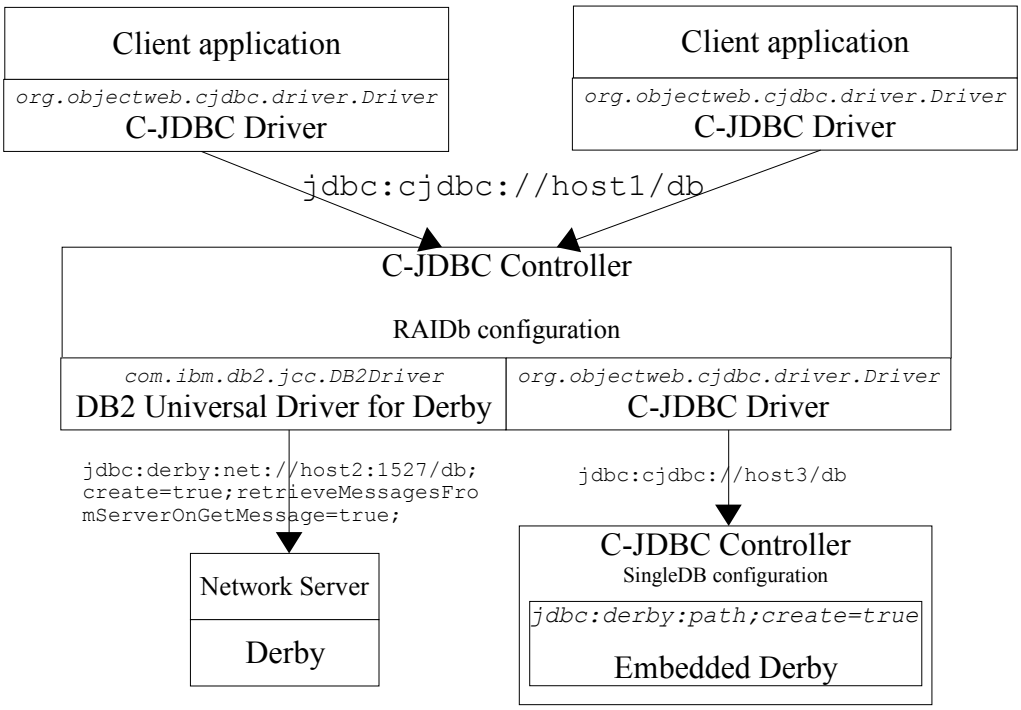
Derby must be configured as the single database in the C-JDBC controller. In order for the C-JDBC controller to start Derby, you must copy `derby.jar` in `$CJDBC_HOME/drivers`. If you experience problems in loading Derby, you can also

completely unjar `derby.jar` in `$CJDBC_HOME/drivers` (there might be some classloader issues due to C-JDBC).
The JDBC url to be used for Derby is of the form *`jdbc:derby:path;create=true`*.
Note that Derby will be started when the virtual database is loaded and will execute in the same virtual machine as the C-JDBC controller.
Here is a virtual database configuration file example for a database named `xpetstore` that the client accesses using `xpetuser/secret` as its login/password. This login/password is mapped to the Derby default `APP/APP` real login/password.

```
<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE C-JDBC PUBLIC "-//ObjectWeb//DTD C-JDBC 1.1//EN" "http://c-
jdbc.objectweb.org/dtds/c-jdbc-1.1.dtd">

<C-JDBC>

  <VirtualDatabase name="xpetstore">

    <AuthenticationManager>
      <Admin>
            <User username="admin" password=""/>
      </Admin>
      <VirtualUsers>
        <VirtualLogin vLogin="xpetuser" vPassword="secret"/>
      </VirtualUsers>
    </AuthenticationManager>

    <DatabaseBackend name="derby1"
      driver="org.apache.derby.jdbc.EmbeddedDriver"
      url="jdbc:derby:c:/xpetstore;create=true"
      connectionTestStatement="values 1">
      <ConnectionManager
            vLogin="xpetuser" rLogin="APP" rPassword="APP">
        <VariablePoolConnectionManager
            initPoolSize="1" minPoolSize="0" maxPoolSize="50"/>
      </ConnectionManager>
    </DatabaseBackend>

    <RequestManager>
      <RequestScheduler>
         <SingleDBScheduler level="query"/>
      </RequestScheduler>

<!-- Uncomment this part if you want to take advantage of C-JDBC
caching features. Note that it is recommended to use at least the
MetadataCache and the ParsingCache even if you don't use the
ResultCache.

      <RequestCache>
         <MetadataCache/>
         <ParsingCache/>
         <ResultCache granularity="table"/>
      </RequestCache>
-->

      <LoadBalancer>
         <SingleDB/>
      </LoadBalancer>
    </RequestManager>

  </VirtualDatabase>

</C-JDBC>
```

2. C-JDBC for Derby clustering

This section explains how to use C-JDBC to build a cluster of Derby database. Both the Network Server + IBM DB2 JDBC Universal Driver for Derby or C-JDBC can be used for Derby remote access. The figure below summarizes the various options to configure a Derby cluster.



2.1. Client side setting

The client application uses the C-JDBC driver to access the Derby cluster. Make sure that c-jdbc-driver.jar is accessible from the classpath. Here is an example of how to load the C-JDBC driver and get a connection:

```
Class.forName("org.objectweb.cjdbc.driver.Driver");
Connection c = DriverManager.getConnection("jdbc:cjdbc://host/db",
"login", "password");
```

2.2. C-JDBC Controller configuration

Derby can be used as any other database in C-JDBC. The configuration of a Derby backend depends if you access Derby using the Network Server with the IBM DB2/Cloudscape driver or using the configuration described in the previous section.

2.2.1. Derby with Network Server

If you want to run multiple instances of Derby on the same machine, you must start them on different ports. Here is an example on how to start instances of Derby on a specific port (1528 in this example):

```
java -cp ..\lib\derby.jar;..\lib\derbynet.jar \
     org.apache.derby.drda.NetworkServerControl start -p 1528
```

You must unjar both `db2jcc.jar` and `db2jcc_license_c.jar` files in `$CJDBC_HOME/drivers`. Note that if you just copy the jar files without unjaring them, the C-JDBC classloader will not be able to load the driver properly. This is a

current C-JDBC issue that might be fixed in future releases.

A Derby backend is declared as follows in the C-JDBC virtual database configuration file:

```
   <DatabaseBackend name="derby1"
      driver="com.ibm.db2.jcc.DB2Driver"
      url="jdbc:derby:net://localhost:1527/xpetstore;create=true;retr
ieveMessagesFromServerOnGetMessage=true;"
      connectionTestStatement="values 1">
     <ConnectionManager vLogin="xpetuser"
            rLogin="APP" rPassword="APP">
       <VariablePoolConnectionManager
            initPoolSize="0" minPoolSize="0" maxPoolSize="50"/>
     </ConnectionManager>
   </DatabaseBackend>
```

2.2.2. Embedded Derby with C-JDBC

If you have chosen to use C-JDBC instead of the Network Server solution, you must start at least one controller for clustering plus one controller for each Derby instance. As the C-JDBC controller already comes with `c-jdbc-driver.jar` in the `drivers/` directory, there is no specific file to copy on the controller used for clustering.
Note that C-JDBC controllers hosting Derby instances must be configured as explained in section 1 and they must be started prior loading the virtual database in the controller used for clustering.

A Derby backend is then defined as any C-JDBC controller would be:

```
   <DatabaseBackend name="derby1"
      driver="org.objectweb.cjdbc.driver.Driver"
      url="jdbc:cjdbc://host/xpetstore"
      connectionTestStatement="select 1">
     <ConnectionManager vLogin="xpetuser"
            rLogin="APP" rPassword="APP">
       <VariablePoolConnectionManager
            initPoolSize="0" minPoolSize="0" maxPoolSize="50"/>
     </ConnectionManager>
   </DatabaseBackend>
```

2.2.3. Using Derby for the RecoveryLog

It is possible to use Derby for the C-JDBC RecoveryLog using any of the configuration described in 2.2.1 and 2.2.2. Note that as `sql` is a reserved keyword for Derby, so you have to override the `sqlColumnName` attribute of the `RecoveryLogTable` element as in this example:

```
      <RecoveryLog>
         <JDBCRecoveryLog
            driver="com.ibm.db2.jcc.DB2Driver"
            url="jdbc:derby:net://localhost:1529/xpetstore;create=tru
e;retrieveMessagesFromServerOnGetMessage=true;"
            login="APP"
            password="APP">
            <RecoveryLogTable tableName="RECOVERY"
                  idColumnType="BIGINT NOT NULL"
                  sqlColumnName="sqlStmt"
                  sqlColumnType="VARCHAR(8192) NOT NULL"
                  extraStatementDefinition=",PRIMARY KEY (id)"/>
            <CheckpointTable tableName="CHECKPOINT"/>
            <BackendTable tableName="BACKENDTABLE"/>
```

```
         </JDBCRecoveryLog>
      </RecoveryLog>
```

2.3. Petstore example

Here is a configuration file used for the xPetstore (http://xpetstore.sourceforge.net) demo with a RAIDb-1 cluster of 2 Derby instances running on port 1527 and 1528, plus one instance of Derby running on port 1529 for the RecoveryLog.

```
<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE C-JDBC PUBLIC "-//ObjectWeb//DTD C-JDBC 1.0.5pre//EN"
"http://c-jdbc.objectweb.org/dtds/c-jdbc-1.0.5pre.dtd">

<C-JDBC>

  <VirtualDatabase name="xpetstore">

    <Monitoring>
      <SQLMonitoring defaultMonitoring="on"/>
    </Monitoring>

    <AuthenticationManager>
      <Admin>
            <User username="admin" password=""/>
      </Admin>
      <VirtualUsers>
        <VirtualLogin vLogin="xpetuser" vPassword="secret"/>
      </VirtualUsers>
    </AuthenticationManager>

    <DatabaseBackend name="derby1"
      driver="com.ibm.db2.jcc.DB2Driver"
      url="jdbc:derby:net://localhost:1527/xpetstore;create=true;retr
ieveMessagesFromServerOnGetMessage=true;"
      connectionTestStatement="values 1">
      <ConnectionManager vLogin="xpetuser"
            rLogin="APP" rPassword="APP">
        <VariablePoolConnectionManager
            initPoolSize="0" minPoolSize="0" maxPoolSize="50"/>
      </ConnectionManager>
    </DatabaseBackend>

    <DatabaseBackend name="derby2"
      driver="com.ibm.db2.jcc.DB2Driver"
      url="jdbc:derby:net://localhost:1528/xpetstore;create=true;retr
ieveMessagesFromServerOnGetMessage=true;"
      connectionTestStatement="values 1">
      <ConnectionManager vLogin="xpetuser"
            rLogin="APP" rPassword="APP">
        <VariablePoolConnectionManager
            initPoolSize="0" minPoolSize="0" maxPoolSize="50"/>
      </ConnectionManager>
    </DatabaseBackend>

    <RequestManager>
      <RequestScheduler>
         <RAIDb-1Scheduler level="pessimisticTransaction"/>
      </RequestScheduler>

      <RequestCache>
         <MetadataCache/>
         <ParsingCache/>
```

```
<!-- Uncomment to enable ResultCache
         <ResultCache granularity="table" />
-->
      </RequestCache>

      <LoadBalancer>
         <RAIDb-1>
            <WaitForCompletion policy="first"/>
            <RAIDb-1-RoundRobin/>
         </RAIDb-1>
      </LoadBalancer>

      <RecoveryLog>
         <JDBCRecoveryLog
            driver="com.ibm.db2.jcc.DB2Driver"
            url="jdbc:derby:net://localhost:1529/xpetstore;create=tru
e;retrieveMessagesFromServerOnGetMessage=true;"
            login="APP"
            password="APP">
            <RecoveryLogTable tableName="RECOVERY"
                  idColumnType="BIGINT NOT NULL"
                  sqlColumnName="sqlStmt"
                  sqlColumnType="VARCHAR(8192) NOT NULL"
                  extraStatementDefinition=",PRIMARY KEY (id)"/>
            <CheckpointTable tableName="CHECKPOINT"/>
            <BackendTable tableName="BACKENDTABLE"/>
         </JDBCRecoveryLog>
      </RecoveryLog>
    </RequestManager>

  </VirtualDatabase>

</C-JDBC>
```