HOWTO
Verwendung von C-JDBC mit Apache Derby

Version 0.2
27 November 2004

Autor: Emmanuel.Cecchet@inria.fr

Grundsätzlich gibt es zwei Möglichkeiten, C-JDBC mit Apache Derby zu verwenden:
- Verwendung der Embedded Version von Derby, die mit Hilfe des C-JDBC Controllers gestartet wird und den C-JDBC Treiber für den Multi-User Remote-Zugriff ermöglicht, ohne dass hierfür der proprietäre IBM DB2 Treiber erforderlich ist.
- Verwendung von C-JDBC, um Derby in Punkto Performanz-Skalierbarkeit und Hochverfügbarkeit zu erweitern. Dies erfordert den Remote-Zugriff auf Derby, der entweder durch die vorgenannte Methode oder alternativ mit Hilfe des Netzwerkservers in Verbindung mit dem IBM DB2 JDBC Universal Treiber für Derby erreicht werden kann.

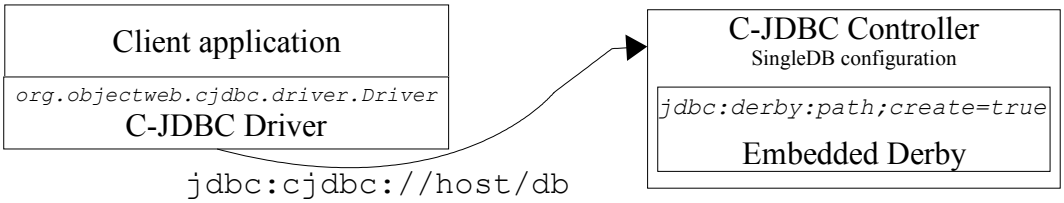
Die Einführungsdokumentation zu C-JDBC finden Sie im Dokumentationsbereich der Internetseite http://c-jdbc.objectweb.org.

Weitere Informationen zu Derby finden Sie unter http://incubator.apache.org/derby.

Zur Beantwortung weitergehender Fragen stehen Ihnen die beiden Mailinglisten c-jdbc@objectweb.org oder derby-user@nagoya.apache.org zur Verfügung.

1. C-JDBC für den Remote-Zugriff auf Derby

In diesem Abschnitt wird erläutert, wie Sie C-JDBC zum Remote-Zugriff auf Derby verwenden können. Dies ist eine Alternative zur Verwendung des Netzwerkservers in Verbindung mit dem IBM DB2 JDBC Universal Treiber für Derby. Die nachfolgende Abbildung verdeutlicht diesen Ansatz:



1.1. Clientseitige Einstellungen

Die Clientanwendung verwendet den C-JDBC Treiber, um auf Derby zuzugreifen. Bitte stellen Sie sicher, dass sich die Datei c-jdbc-driver.jar im Classpath Ihres Systems befindet. Im nachfolgenden Beispiel sehen Sie, wie C-JDBC Treiber geladen und zum Verbindungsaufbau verwendet wird:

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

1.2. Konfiguration des C-JDBC Controllers

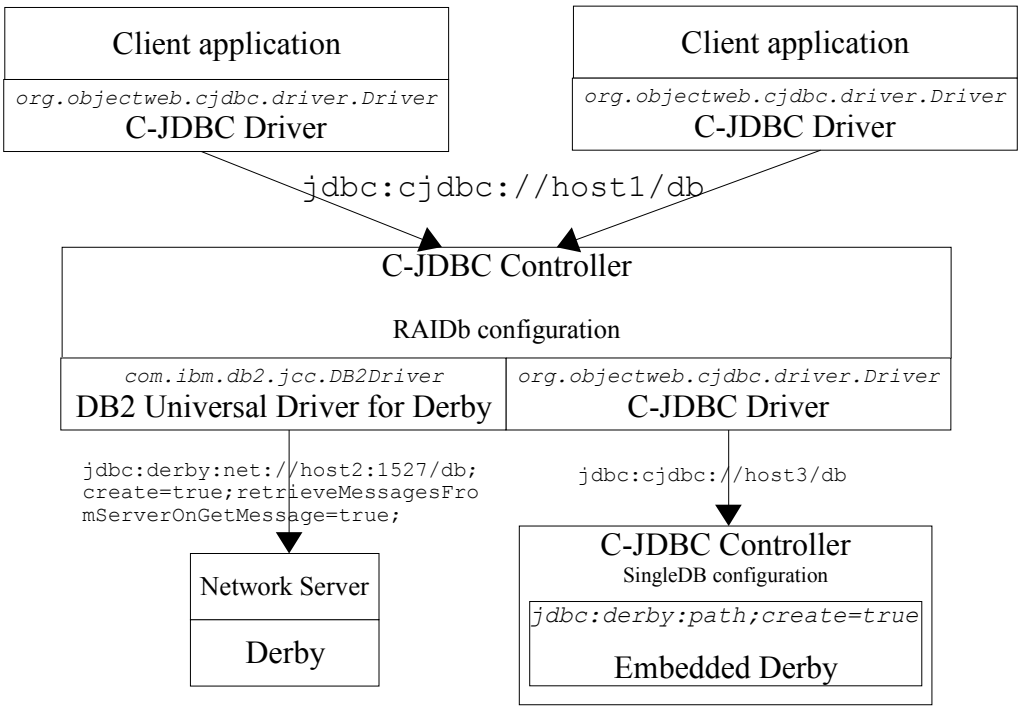
Derby muss im C-JDBC Controller als einzelne Datenbank konfiguriert werden. Um dem C-JDBC Controller das Starten von Derby zu ermöglichen, müssen Sie die Datei `derby.jar` in das Verzeichnis `$CJDBC_HOME/drivers` kopieren. Sollten beim Laden von Derby Fehler auftreten, liegt hier möglicherweise ein Problem des C-JDBC Classloaders vor. In diesem Fall sollten Sie die Datei `derby.jar` vollständig in das Verzeichnis `$CJDBC_HOME/drivers` entpacken.
Die JDBC URL Verwendung von Derby ist wie folgt aufgebaut: *`jdbc:derby:path;create=true`*. Beachten Sie hierbei, dass Derby zusammen mit der virtuellen Datenbank gestartet wird und auf der selben virtuellen Maschine wie der C-JDBC Controller ausgeführt wird.
Nachfolgend sehen Sie eine Beispielkonfiguration für eine virtuelle Datenbank namens `xpetstore`, auf die mit dem Login `xpetuser` und dem Passwort `secret` zugegriffen werden soll. Diese Accountinformationen werden entsprechend auf die Derby Accountinformationen (Login `APP`, Passwort `APP`) abgebildet.

```
<?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. Clustering von Derby mit C-JDBC

In diesem Abschnitt wird erläutert, wie Sie mit Hilfe von C-JDBC einen Cluster aus Derby Datenbanken aufbauen können. Für den Zugriff auf Derby können hierbei können beide Varianten (Netzwerkserver in Verbindung mit dem IBM DB2 JDBC Universal Treiber für Derby oder C-JDBC) verwendet werden. Die nachfolgende Abbildung zeigt die unterschiedlichen Konfigurationsmöglichkeiten zum Aufbau eines Derby Clusters.



2.1. Clientseitige Einstellungen

Die Clientanwendung verwendet den C-JDBC Treiber, um auf den Derby Cluster zugreifen zu können. Stellen Sie hierzu sicher, dass sich die Datei c-jdbc-driver.jar im Classpath befindet. Im nachfolgenden Beispiel sehen Sie, wie C-JDBC Treiber geladen und zum Verbindungsaufbau verwendet wird:

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

2.2. Konfiguration des C-JDBC Controllers

Derby kann ebenso wie jede andere Datenbank von C-JDBC verwendet werden. Sie können Derby dabei mit Hilfe des Netzwerkservers in Verbindung mit dem IBM DB/2 bzw. Cloudscape-Treiber oder alternativ wie im vorhergehenden Abschnitt beschrieben verwenden. Die Konfiguration der Derby Backends muss dabei entsprechend der gewünschten Zugriffsmethode angepasst werden.

2.2.1. Derby als Netzwerkserver

Sollen mehrere Derby-Instanzen auf einem Rechner gestartet werden, muss für jede dieser Instanzen ein eigener Port angegeben werden. Nachfolgend ist ein Beispiel zum Start einer Derby-Instanz auf Port 1528 abgebildet:

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

Sie müssen die beiden Jar-Dateien `db2jcc.jar` und `db2jcc_license_c.jar` in das Verzeichnis `$CJDBC_HOME/drivers` entpacken. Bitte beachten Sie, dass der C-JDBC Classloader die in den Jar-Dateien enthaltenen Klassen-Dateien momentan nur dann erfolgreich laden kann, wenn sie zuvor entsprechend entpackt wurden. In einer zukünftigen Version von C-JDBC wird es möglich sein, die Klassen-Dateien auch im nicht entpackten Zustand bereitzustellen.

Ein Derby Backend wird wie folgt in der Konfigurationsdatei der virtuellen Datenbank deklariert:

```
   <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 und C-JDBC

Wenn Sie C-JDBC anstelle der Netzwerkserver-Lösung verwenden möchten, müssen Sie mindestens einen Controller für das Clustering und je einen weiteren Controller für jede Derby-Instanz einrichten. Der C-JDBC Controller verfügt bereits über den erforderlichen Treiber (`c-jdbc-driver.jar` im `drivers-Verzeichnis`), so dass keine weiteren Dateien für den Betrieb des Controllers bereitgestellt werden müssen.
Bitte beachten Sie hierbei, dass die C-JDBC Controller zur Verwaltung der Derby -Instanzen wie in Abschnitt 1 beschrieben konfiguriert werden müssen. Weiterhin ist es erforderlich, diese Instanzen zu starten, bevor die virtuelle Datenbank des für das Clustering verantwortlichen Controllers geladen wird.

Ein Derby Backend wird folgendermaßen als C-JDBC Controller konfiguriert:

```
   <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. Verwendung von Derby für das Wiederherstellungs-Log

Mit Hilfe der in 2.2.1 und 2.2.2 beschriebenen Konfiguration können Sie Derby für das Wiederherstellungs-Log von C-JDBC verwenden. Bitte beachten Sie hierbei, dass `sql` in Derby ein reserviertes Schlüsselwort ist und Sie daher das Attribut `sqlColumnName` des Elements `RecoveryLogTable` entsprechend anpassen müssen:

```
      <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 Beispielanwendung

Nachfolgend sehen Sie eine Abbildung der Konfigurationsdatei für die xPetstore-Beispielanwendung (zu finden unter http://xpetstore.sourceforge.net). Diese Beispielanwendung verwendet einen RAIDb-1 Cluster mit zwei Derby-Instanzen (Ports 1527 und 1528) und einer weiteren Derby-Instanz (Port 1529) zur Aufnahme und Verwaltung des Wiederherstellungs-Logs.

```
<?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>
```