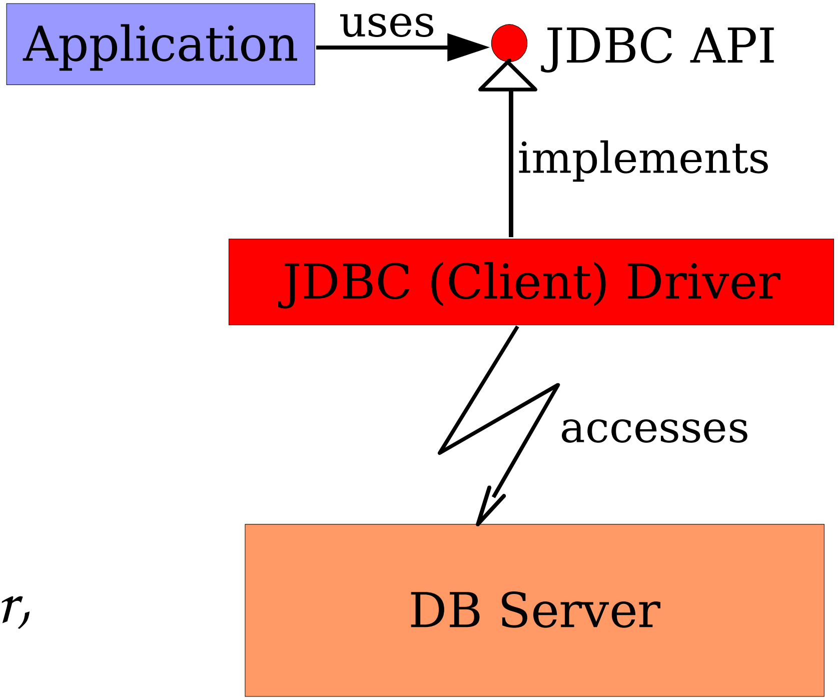
Distributed SQL Clusters

(C-JDBC)

- JDBC
- C-JDBC:
  - Architecture
  - Standard Features
  - Possible Plugins
  - Tools
- JDBC in Other Languages

JDBC

- Universal accesses API for relational databases.
- Written in Java, but easily accessible from other languages (Python, Ruby, Groovy, PNuts, C/C++,...).
- Drivers exist for all SQL database and some other storages (XML files, etc.).
- Very large number of tools based on JDBC.
- Very large number of users.
- Several Atlas Applications use JDBC (Tag Collector, AMI, AtCom, SQLTuple/ColMan, Octopus Replicator, ...).
- There is no support for JDBC from LCG/AA/Pool/LCG3D/....



C-JDBC Architecture

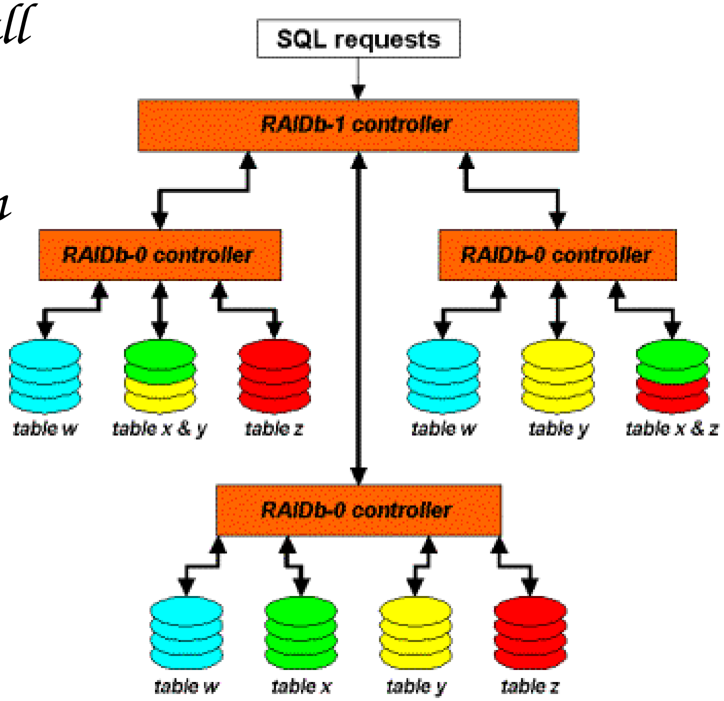
- SQL tables can be spread over several database Servers, some tables may be replicated. User wants a single front-end.

- C-JDBC acts as a (Proxy) Virtual SQL Server forwarding all requests to appropriate databases (real or another virtual). Replicated and/or complementary tables are supported (even on heterogeneous Servers), similar do RAID disks.

- C-JDBC is used via its JDBC driver, so any application using JDBC API can directly use C-JDBC. <u>No application modification is required</u> to use C-JDBC.

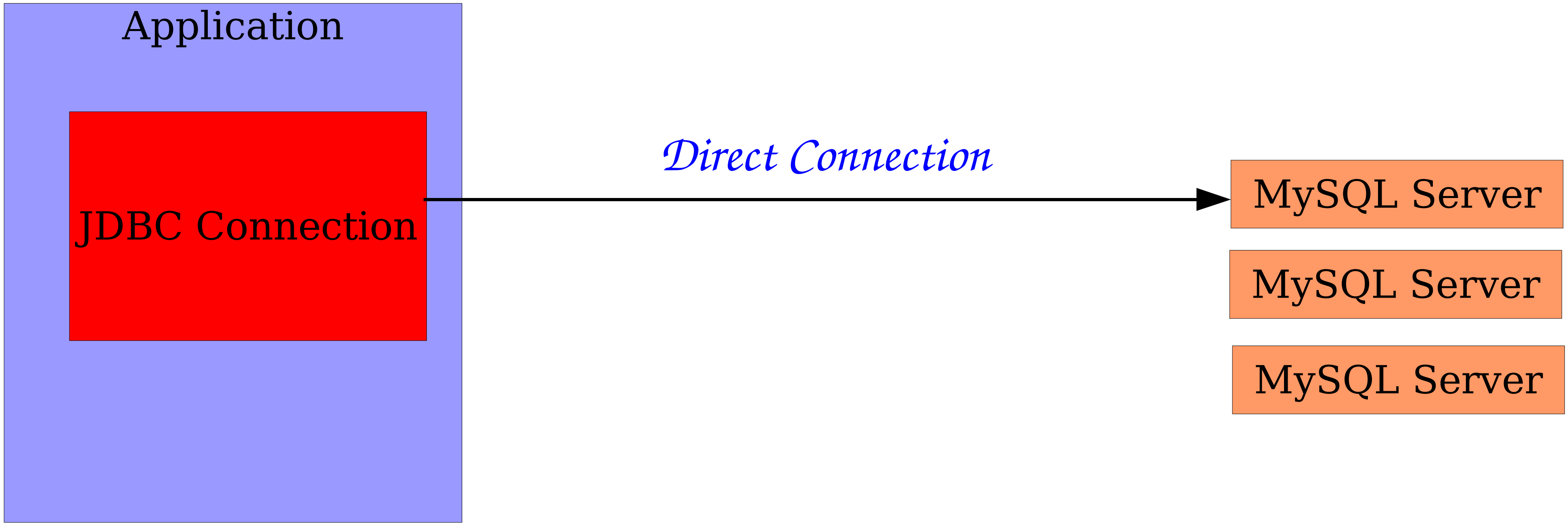
- C-JDBC directly handles <u>any SQL query</u>. No pre-knowledge is needed. No specialized interface is required.

- C-JDBC handles <u>both query (read access) and update (write access)</u>.



C-JDBC Architecture

```
// Direct connection to MySQL server
Connection connection = DriverManager.getConnection("jdbc:mysql://mysqlserver.cern.ch/Tuples",
                                                     "user", "passwd");
```



Application

JDBC Connection

Direct Connection

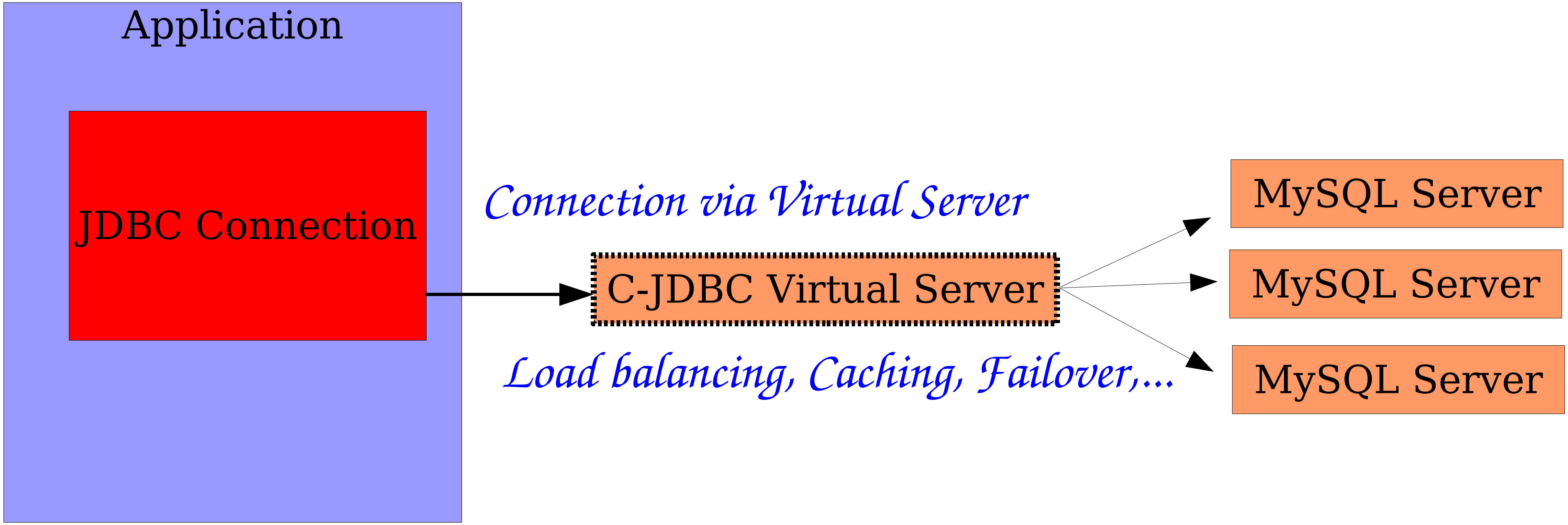
MySQL Server

MySQL Server

MySQL Server

C-JDBC Architecture

```
// Direct connection to MySQL server
Connection connection = DriverManager.getConnection("jdbc:mysql://mysqlserver.cern.ch/Tuples",
                                                     "user", "passwd");
```



```
// Connection via C-JDBC virtual server
Connection connection = DriverManager.getConnection("jdbc:cjdbc://cjdbcserver.cern.ch/Tuples",
                                                     "user", "passwd");
```

The only change in the Application (can be configured via Job Options).

Standard C-JDBC Features

- Load balancing: Several strategies are available (round-robin, round-robin with weights, adaptable round-robin), others can be introduced.
- Caching: Results of SQL queries are cached, with expiration policy depending on chosen strategy.
- Connection Pooling: Connections are reused at the level of C-JDBC Server.
- Failover: Two kinds of Server replication are available:
  - Horizontal Scaling: User connects to a group of C-JDBC Servers, where at least one should be available.
  - Vertical Scaling: Servers with tables replicas are hidden behind one C-JDBC Server.
- Backup/Restore: Tables or whole database can be backuped or replicated (using Enhydra Octopus).
- Journaling/CheckPointing: Database transactions are recorded and saved on request for later recovery.
- Monitoring: All transactions are monitored to allow performance tuning.
- Synchronisation: Write/update modifies all replicas.
- Authentication: C-JDBC Server maps user credentials to all backend Servers.

Possible C-JDBC Plugins

- Parallel Processing: The data are spread over several tables and servers and accessed transparently as one table. (For example AttributeList database.)
- Access to Partially Updated Servers: Some tables have more data then others. (For example, some newer data are not yet replicated elsewhere).
- Active Cache: Cached data update themselves automatically when necessary.
- Query Filtering and Prioritizing: Too expensive queries are postponed, forwarded elsewhere or refused.
- Query Prediction: Cached query results are used to predict future query result.
- Adaptive Indexing and Replication: Monitoring information is used to tune databases for performance.
- Grid Authentication.
- Connection to Catalogs, Service Discovery: Configuration/setup is created automaticaly.

C-JDBC Tools

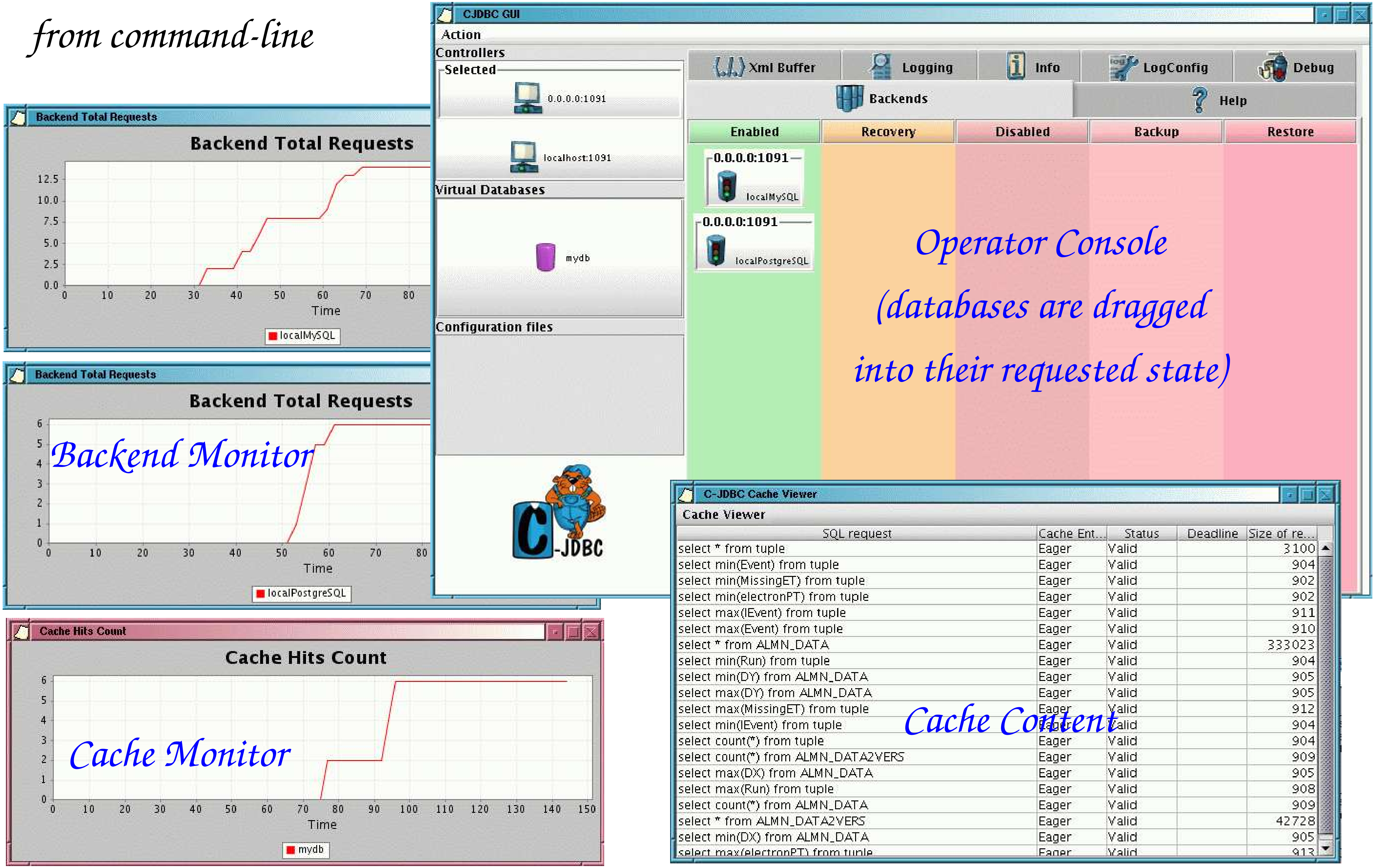
- *C-JDBC Configuration is specified via an XML file.*
- *Management can be performed via:*
  - *GUI*
  - *Command line (and scripts)*
  - *JMX (Java Management eXtension – distributed, Web based)*
  - *Code*

```
<DatabaseBackend name="cernOracle"
                          driver="oracle.jdbc.driver.OracleDriver"
                          driverPath="/opt/Oracle/ojdbc14_g.jar"
                          url="jdbc:oracle:thin:@oradev9.cern.ch:1521:D9"
                          connectionTestStatement="select * from dual">
    <ConnectionManager vLogin="test" rLogin="user" rPassword="password">
            <VariablePoolConnectionManager initPoolSize="40"/>
            </ConnectionManager>
    </DatabaseBackend>

<RequestManager beginTimeout="0" commitTimeout="0" rollbackTimeout="0">
    <RequestScheduler>
            <RAIDb-2Scheduler level="pessimisticTransaction"/>
            </RequestScheduler>
    <RequestCache>
            <ResultCache granularity="database">
                    <DefaultResultCacheRule>
                             <EagerCaching/>
                             </DefaultResultCacheRule>
                    </ResultCache>
            </RequestCache>
    <LoadBalancer>
            <RAIDb-2>
                    <CreateTable policy="roundRobin" numberOfNodes="1">
                             <BackendName name="local"/>
                             <BackendName name="cern"/>
                             </CreateTable>
                     <RAIDb-2-RoundRobin/>
                     </RAIDb-2>
             </LoadBalancer>
   </RequestManager>
```

C-JDBC Tools

- *Many other components exist*
- *Everything (and more) possible from command-line*



SQL request	Cache Ent...	Status	Deadline	Size of re...
select * from tuple	Eager	Valid		3100
select min(Event) from tuple	Eager	Valid		904
select min(MissingET) from tuple	Eager	Valid		902
select min(electronPT) from tuple	Eager	Valid		902
select max(lEvent) from tuple	Eager	Valid		911
select max(Event) from tuple	Eager	Valid		910
select * from ALMN_DATA	Eager	Valid		333023
select min(Run) from tuple	Eager	Valid		904
select min(DY) from ALMN_DATA	Eager	Valid		905
select max(DY) from ALMN_DATA	Eager	Valid		905
select max(MissingET) from tuple	Eager	Valid		912
select min(lEvent) from tuple	Eager	Valid		904
select count(*) from tuple	Eager	Valid		904
select count(*) from ALMN_DATA2VERS	Eager	Valid		909
select max(DX) from ALMN_DATA	Eager	Valid		905
select max(Run) from tuple	Eager	Valid		908
select count(*) from ALMN_DATA	Eager	Valid		909
select * from ALMN_DATA2VERS	Eager	Valid		42728
select min(DX) from ALMN_DATA	Eager	Valid		905
select max(electronPT) from tuple	Eager	Valid		913

JDBC in Other Languages

Python (proxies generated transparently at runtime)

```
from java.lang import System
from java.sql  import DriverManager

System.setProperty("jdbc.drivers", "org.gjt.mm.mysql.Driver")
connection = DriverManager.getConnection("jdbc:mysql://localhost/Tuples", "test", "test");
rs = connection.createStatement().executeQuery("SELECT * from tuple;");
while rs.next():
  print rs.getDouble(4)
```

C++ (proxies generated and compiled in advance by JACE)

```
// many #include statements and JVM initialisation

jace::helper::createVm(loader, options);

System::setProperty("jdbc.drivers", "org.gjt.mm.mysql.Driver");
Connection connection = DriverManager::getConnection("jdbc:mysql://localhost/Tuples", "test", "test");
ResultSet rs = connection.createStatement().executeQuery("SELECT * from tuple;");
while (rs.next()) {
  cout << rs.getDouble(4) << endl;
  }
```

Groovy (no proxies needed)

```
import groovy.sql.Sql

sql = Sql.newInstance("jdbc:mysql://localhost/Tuples", "test", "test",  "org.gjt.mm.mysql.Driver")

sql.eachRow("select * from tuple") {
  println it.Event
  }
```

Documentation

- C-JDBC is provided by ObjectWeb Consortium, released under GPL, it has active user base and responsive developers. It is probably the only (so the best) such OpenSource Tool.
- C-JDBC works well with other ObjectWeb Tools, like Octopus Replicator, JOnAS Application Server, Speedo JDO, JORAM Messaging, etc.
- Direct support for ODBC is scheduled for C-JDBC 2.0.
- Documentation:
    - Home: http://c-jdbc.objectweb.org
    - Article in Programmez! N°69/2004
    - "Run-You-Own-Server" playground available, look at ~hrivnac/public/CJDBC.conf/000.txt