Introducing Apache Derby

Dan Debrunner
djd@debrunners.com

An Apology – Derby vs. Derby

- **71-72, 74-75 – Division One Champions**

Derby

- **Derby provides developers a small footprint, standards-based Java database that can be tightly embedded into any Java based solution**

Brief History

- **1996 – Cloudscape, Inc. startup – Oakland, CA**
- **1997 – JBMS 1.0**
- **Apr 1999 – Cloudscape 2.0**
- **Dec 1999 – Acquired by Informix Software**
- **June 2001 – Cloudscape 4.0**
- **July 2001 – Acquired by IBM**
- **Dec 2001 – IBM Cloudscape 5.0**
- **2003 – IBM Cloudscape 5.1, FP1 & FP2**
  - Significant IBM use as a component
- **Aug 2004 – Open Sourced**
  - IBM contributes code to Apache as Derby

Apache Derby

- **IBM contributed the Cloudscape code to Apache Software Foundation as Derby**
- **Apache DB project sponsored Derby into incubation at Apache**
- **Derby up and running at Apache**
  - http://incubator.apache.org/derby

- **Derby is an effort undergoing incubation at the Apache Software Foundation. Incubation is required of all newly accepted projects until a further review indicates that the infrastructure, communications, and decision-making process have stabilised in a manner consistent with other successful ASF projects. While incubation status is not necessarily a reflection of the completeness or stability of the code, it does indicate that the project has yet to be fully endorsed by the ASF.**

6 Hours 26 minutes !!!!

- Thu, 26 Aug 2004 10:17:04 _0400
  - *okey, the necesary paperwork has been executed, so i just uploaded the derby source code into the subversion repository.*
  - Rodent of Unusual Size
- Thu, 26 Aug 2004 22:43:37 +0200 (CEST)
  - *This is absolutely superb! I have now downloaded the source code, compiled it, build jars, made a test project in eclipse and am now able to debug/single step thru the derby code. All done within a couple of hours. :-)*
  - Steen Jansdal

First Bugs & Patches Within Hours

- 7h7m - Thu, 26 Aug 2004 23:24:47 +0200
  - First documentation bug
    Christian d'Heureuse
- 7h43m - Fri, 27 Aug 2004 00:00:00 +0200
  - First code bug
    Christian d'Heureuse
- 7h46m - Fri, 27 Aug 2004 00:03:27 +0200
  - First patch (from someone new to Derby code)
    Jan Hlavat_

Derby – Embedded Engine

Keys Points to Remember About Derby

- **Pure Java**
- **Embedded database**
- **Small footprint**
- **Standards based**
- **Complete relational database engine**

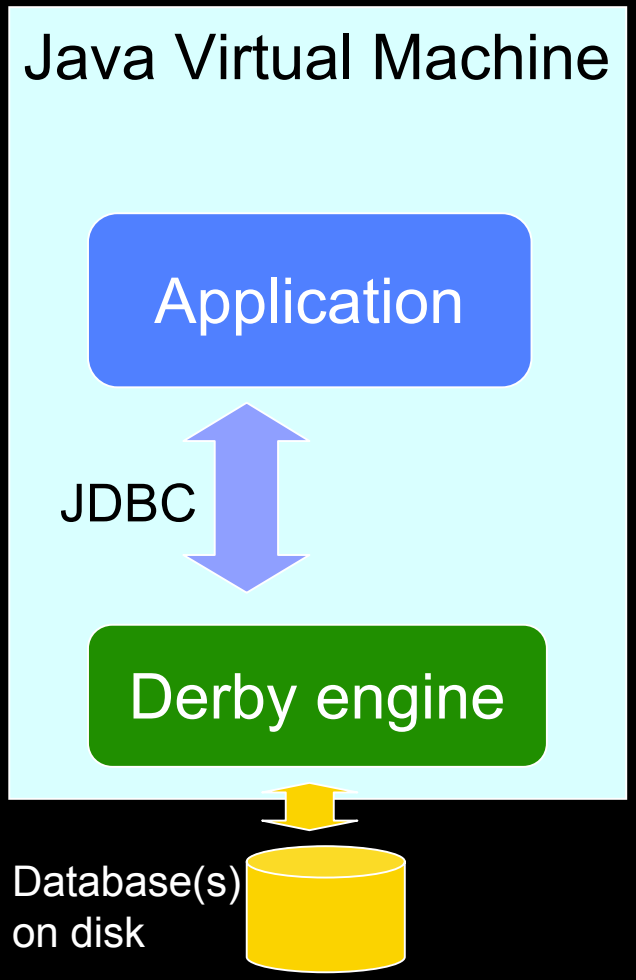
Pure Java

- **Database code written using the Java programming language**
- **Write Once Run Anywhere**
  - Requires a J2SE 1.3 or 1.4 virtual machine
  - Any hardware, any operating system
- **Single binary does run everywhere**
  - Linux, Windows, MacOs, AIX, Solaris, Z/OS, AS400, OS/390, …
- **Database on-disk format is platform independent too!**

Embedded Database

- **Database engine becomes an integral part of the Java application**
- **No additional process**
  - Runs in application's virtual machine
  - Database requests now just method calls within the JVM
- **Start & shutdown controlled by application**
- **Just a library to Java applications**
  - Shipped in a single jar file
- **Becomes invisible to the user**
  - No battles over must use DB2, Oracle, MS-SQL etc.

Embedded Usage



- Database only accessible from single JVM
- Java/JDBC only
- No network connectivity

- Typically is single application per JVM (but could be multiple)

Small Footprint

- **Engine jar file is around 2Mb**
  - Optional Jar files
    - Network server ~150k
    - Tools ~200k
- **Runtime memory use**
  - Dependent on application, data caching etc.
  - Can run when Java heap memory restricted to 4Mb
  - Have run in machines with only 16Mb physical memory

Standards

- **SQL**
  - SQL92, SQL99, SQL2003, SQL/XML, …
- **Java**
  - J2SE 1.3, 1.4
  - JDBC 2.0 & 3.0
  - J2EE – JDBC driver for J2EE 1.4 & 1.3
  - J2ME/OSGi
- **DRDA**

Standards to Allow Migration

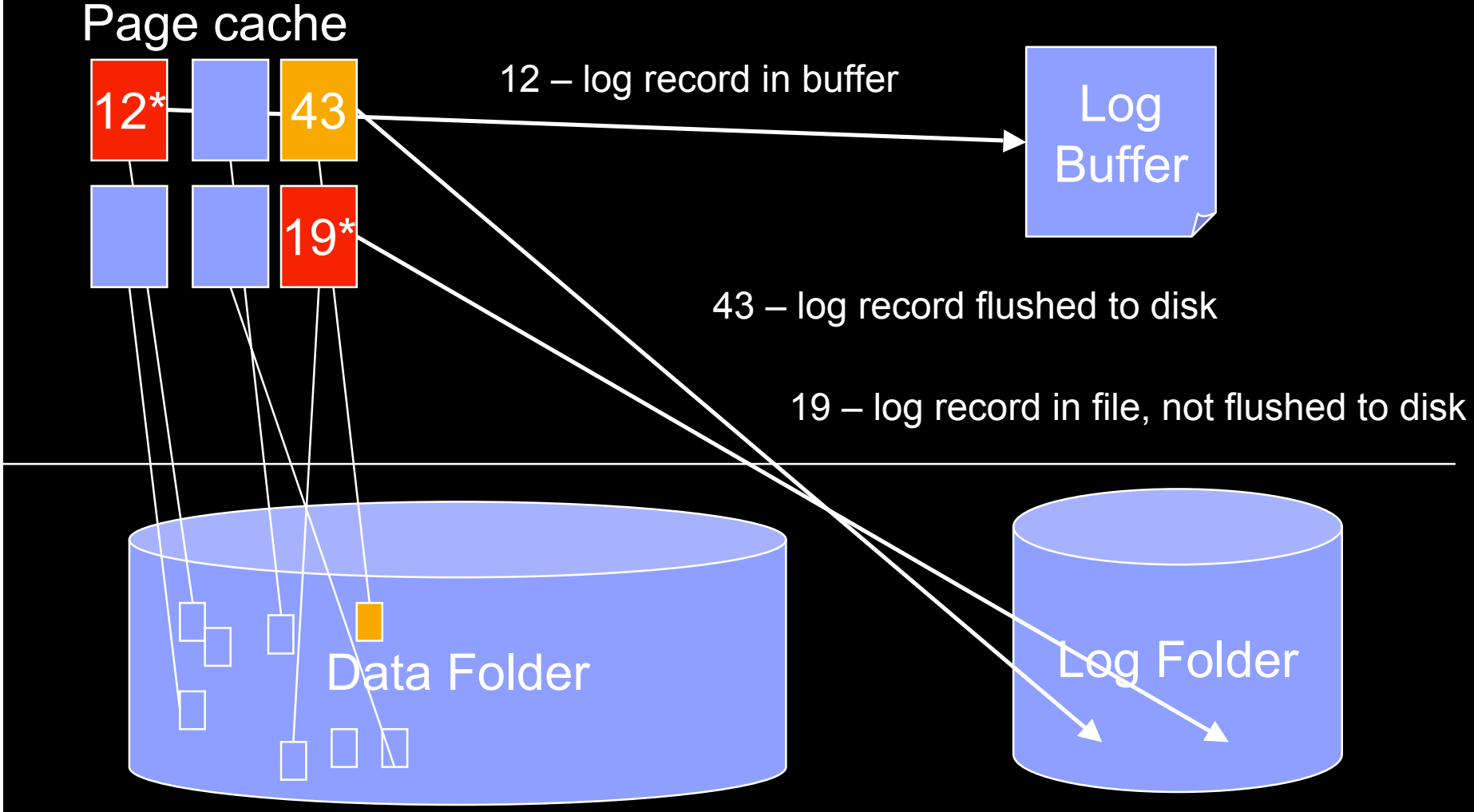
- **Derby does perform well in its own right**
- **Lacking features of an enterprise DB**

- **Develop on Derby, deploy on enterprise DB**
- **Low-end deployment on Derby, high-end on enterprise DB**
- **Initial deploy on Derby, migrate to enterprise DB as needed**

Complete Relational Engine

- **Multi-user, multi-threaded, transactions, row locking, isolation levels, lock deadlock detections, crash recovery, backup & restore**
- **SQL**
  - Tables, indexes, views, triggers, procedures, functions, temp tables
  - foreign key and check constraints
  - joins, cost based optimizer
- **Data caching, statement caching, write ahead log, group commit**
- **Multiple databases per system**

Aries Logging, Page Cache Based



Remember!

- **Pure Java**
- **Embedded database**
- **Small footprint**
- **Standards based**
- **Complete relational database engine**
- **➔ Easy to use embedded database**

Derby – Guidelines

Derby Guidelines

- **Performance dependent on your application**
- **However, if you fall into these categories you can be successful with Derby**

- **Java is not slow! Today's virtual machines contain Just In Time (JIT) compilers that compile interpreted byte code to native machine code**
  - Optimizing for the execute time code paths

Guidelines -- 1-2 cpu machines

- **Derby is thread safe, and takes advantage of Java threading and synchronization**
- **Not optimized to take advantage of more than 2cpus, scaling will be limited on 4 or more.**
- **Not limited to 2 cpu machines**
  - E.,g. can run Derby in a monitoring application on a 8 way box.

Guidelines – Less than 50Gb of data

- **Derby is limited to a single logical disk**
- **Transaction log can be separated to a separate disk to benefit performance**
- **Hence most applications will be limited by disk throughput for a single disk**
  - Can offset with large page cache
- **Fast disk controller can help**
- **Can use multiple physical disks through RAID devices or OS software striping**
  - Tending away from zero admin

Guidelines – 20-30 Active Connections

- **20-30 connections concurrently executing SQL statements**
- **20-30 is a typical number for an application server connection pool**
- **Connections are limited by memory, low overhead per connection**

Guidelines – 100-500 updates per second

- **Depends on complexity of updates and transactions**
- **Depends on level of read activity**
- **Derby implements group commit for the transaction log to allow a single disk flush to commit multiple transactions**

Guidelines Summary

- **Those are guidelines, not hard and fast rules, not hard limits**
- **Test you own application with Derby to get an idea of performance**
  - Ensure the JIT has kicked in – run for a while
- **Guidelines verified with an industry standard benchmark on a 2cpu box where Derby can match enterprise class databases**

Derby - JDBC

Using Derby

- **Derby's API is JDBC and SQL**
- **If you already know these then using Derby will be easy**
- **If you don't then there are many books on these subjects and Derby is an excellent database to learn on**
- **JDBC is the API for Derby, not an add-on**

Standard JDBC Driver & Connection Code

- **Most JDBC applications have code similar to**

```
// get from properties, ui form etc.
String driverClassName = ...
String databaseURL = ...

// Load the JDBC Driver
Class.forName(driverClassName);

// Open a connection to the database
Connection conn =
     DriverManager.getConnection(databaseURL);
```

Derby Specific Start Up Code …

- **Nothing!**
- **Loading the Derby JDBC driver starts the embedded engine**
  - The engine is the runtime code that supports multiple databases & provides services to those databases such as an error log.
- **Making a connection request to the database starts that database, if it was not already running**
  - Starting a database means its files are opened, initial catalog information loaded into memory etc.

Derby Embedded Driver & URL

- **Driver**
  `org.apache.derby.jdbc.EmbeddedDriver`
- **`jdbc:derby:[`*`database`*`][;`*`attribute`*`=`*`value`*`]*`**
- **Attributes can also be passed in the Properties parameter of `DriverManager.getConnection` methods.**
- **user,password,create,databaseName, logDevce, territory, *encryption options, recovery options***

Derby Specific Shut-down Options

- **Standard JDBC API but specific to Derby**

- **1) Do nothing – exiting VM will stop Derby**
  - Recovery will be run on next start
- **2) Shutdown a single database**
  - Make a connection request to the database with the attribute shutdown=true – jdbc:derby:db1;shutdown=true
- **3) Shutdown all databases and the engine**
  - Make a connection request without a database name but with the attribute shutdown=true - jdbc:derby:;shutdown=true

Embedded JDBC Support

- **JDBC 2.0 and 3.0 driver implementations**
  - Single driver to the application, automatically detects required JDBC level and loads correct code
- **DataSource support including Connection Pool and XA for integration with application servers**
- **Statements, PreparedStatements, CallableStatements, ParameterMetaData, Savepoints, holdable ResultSets, statement batching, isolation levels, ...**

DataSource API

- **J2EE applications access connections using javax.sql.DataSource API**
- **Hides specific of back-end database to application server administrator**
- **Derby provides base, pooling and XA implementations**
- **Connection Pooling is provided by App Server**
- **XA Transaction Manager is provided by App Server**
- **Requesting a connection from a Derby DataSource will start the engine and/or database if required**

DataSource Classes

- **org.apache.derby.jdbc.**
  - EmbeddedDataSource
  - EmbeddedConnectionPoolDataSource
  - EmbeddedXADataSource
- **Derby specific DataSource properties**
  - connectionAttributes – list of JDBC URL attributes
  - createDatabase – “create” – connection request will create database
  - shutdownDatabase – “shutdown” - connection request will shut down database

Embedded JDBC – What's Missing

- **Updateable ResultSets – to-do**
- **Statement.cancel() - to-do**
- **Statement.setQueryTimeout – Derby-31, to-do**
- **Connection pooling in CPDS implementation**
- **JSR169 (J2ME/CDC/Foundation) – to-do**

Derby – SQL Language

Derby Database

- **A database is a single folder containing set of Derby data files**
- **Created by making a connection request with the create=true attribute**
- **Database name maps to relative or absolute path**
- **No pre-allocation of space**
- **Transaction log contained in folder, or optionally in another folder (disk) for recoverability**

SQL Data Types

- **CHAR, VARCHAR, LONG VARCHAR, CLOB**
- **SMALLINT, INT, BIGINT, DECIMAL, REAL, FLOAT,DOUBLE**
- **CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, LONG VARCHAR FOR BIT DATA, BLOB**
- **DATE, TIME, TIMESTAMP**

SQL Tables

- **Up to 1012 columns**
- **No row size limit**
  - Rows can overflow onto multiple pages
  - Columns can overflow onto multiple pages
  - LOBs stored “in-line”
  - All transparently to application
- **Page size defaults to 4k, automatically set to 32k for a table containing LOBs or potentially a large row size**
- **No partitions – all tables just stored in the database**

SQL Indexes

- **BTREE Indexes**
- **Unique and non-unique**
- **Ascending or descending**
- **Up to 16 columns per key**
- **Key size limited to half a page**

SQL Constraints

- **Primary Key**
- **Unique**
- **Foreign key constraints**
  - Referential actions
  - Automatic index creation
- **Check Constraints**
- **Triggers**
  - before and after, row and statement
  - Single statement action (no SQL procedure language)
  - Function calls allowed in action

SQL Routines

- **Procedures and functions written in Java**
  - LANGUAGE JAVA PARAMETER STYLE JAVA
- **SQL Standard part 13**
- **Procedures can call back into engine using JDBC and SQL Standard connection URL**
  - jdbc:default:connection
- **Procedures can return multiple result sets to the caller**
- **Functions are read-only**

Java Functions

- **Allow easy extension of Derby**
- **CREATE FUNCTION COS(A DOUBLE)**
  RETURNS DOUBLE
  LANGUAGE JAVA PARAMETER STYLE JAVA
  NO SQL
  EXTERNAL NAME
  'java.lang.Math.cos'

ij> VALUES COS(2.1);
1

-0.504846104599857 6
1 row selected

SQL Highlights

- **Views – read-only**
- **Inner & Outer Joins**
- **Schemas**
- **GROUP BY, HAVING**
- **Sub queries**
- **Aggregates (max, min,count,sum,distinct)**
- **Updateable Cursors**
- **VALUES statement**

SQL – What's Missing – all on to-do list

- **Spatial & indexing**
- **GRANT/REVOKE**
- **User Defined Types**
- **SQL/XML**
- **National character types**
- **Overloaded routines**
- **SQL Procedure language**

Derby Security

- **On-disk database Encryption**
- **Builtin, LDAP or pluggable authentication**
- **Simple authorization via configuration**
- **Java 2 Security Manager enabled**

- **See '*Securing Data with Apache Derby*'**

Derby Network Server

Derby's Client/Server Mode

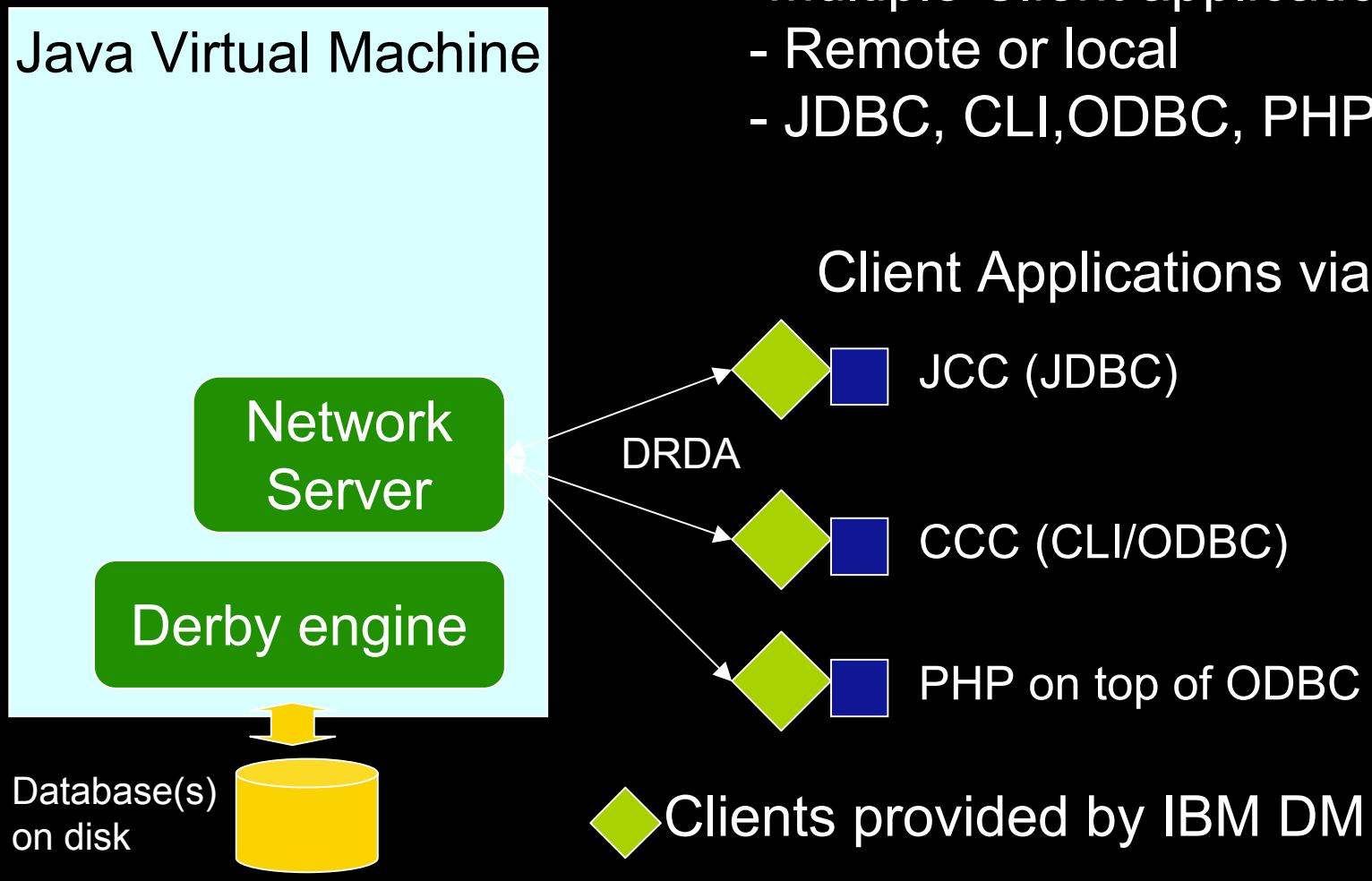
- **Access from a single VM in embedded mode can be a restriction**
- **Network Server allows Derby to act as a traditional client server database**
- **Industry standard DRDA over TCP/IP**
- **Network Server itself is pure Java and embedded**

- **Uses embedded JDBC driver against Derby**
  - It's a Java DRDA to JDBC converter

Network Server Clients

- **JDBC using IBM's DB2 Universal JDBC Client**
  - Different driver and JDBC URL to embedded
  - DataSource and Connection Pooling DataSource
    - Future XA support for Derby
- **ODBC/CLI using IBM's DB2's Universal ODBC/CLI client**
  - Enables PHP / Perl / .NET
- **Universal drivers enabled by use of DRDA**

Traditional Client Server



-Multiple Client applications
- Remote or local
- JDBC, CLI,ODBC, PHP

Client Applications via

Network Server Security Risks

- **Changes security environment – open TCP/IP port on machine running Derby**
- **Derby's limited authorization means any remote read-write user can**
  - Create a Java procedure or function that executes Java code on the server machine
  - Say, shutdown the server using java.lang.System.exit
  - Attempt to read/write files on the system

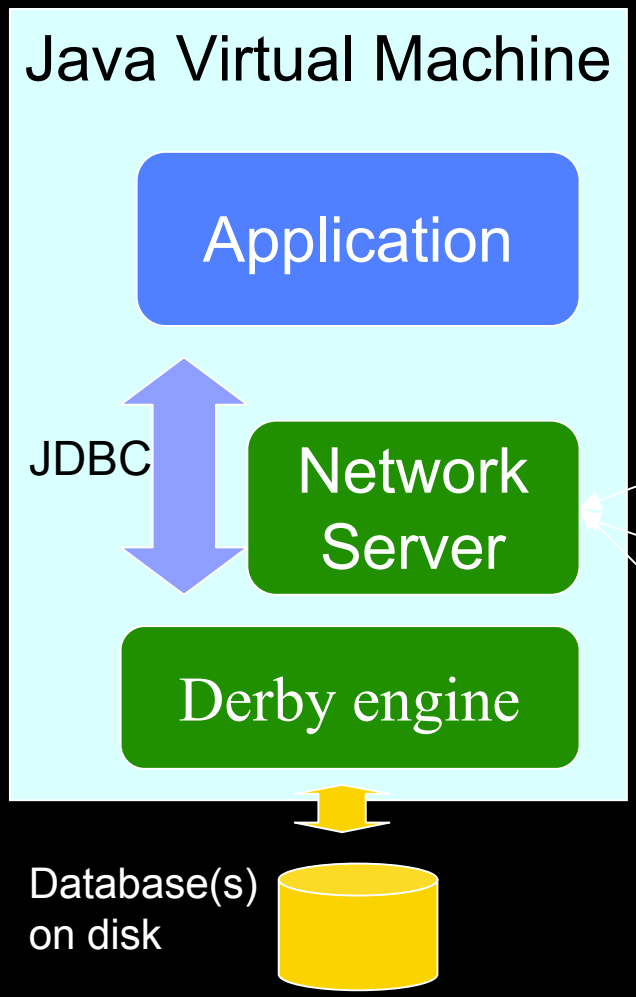
Network Server Security Risk Avoidance

- **By default**
  - Only accepts connections on loop back address
  - Restricts admin commands to local host
- **Before configuring to listen on external address (socket)**
  - Enable user authentication
  - Run the network server with the Java 2 Security Manager enabled
    - Procedures and functions will have no permissions
    - Application code can be granted permissions
  - For intranet applications, ensure your firewall blocks the network server port

Embedded Network Server

- **Adds on to embedded engine to provide access to database from outside the application's VM**
  - Same host or remote host
- **Allows developers to work on database while stand alone application is running**
- **Allows reporting capability to be added onto a stand alone application**
- **Enabled by property setting and additional jar file, no code changes to application**
  - derby.drda.startNetworkServer=true

Embedded Network Server



- Especially useful developing & debugging embedded database usage (i)
- Connect to running application with schema browsers etc.

Clients provided by IBM DM

Potential Open Source Clients

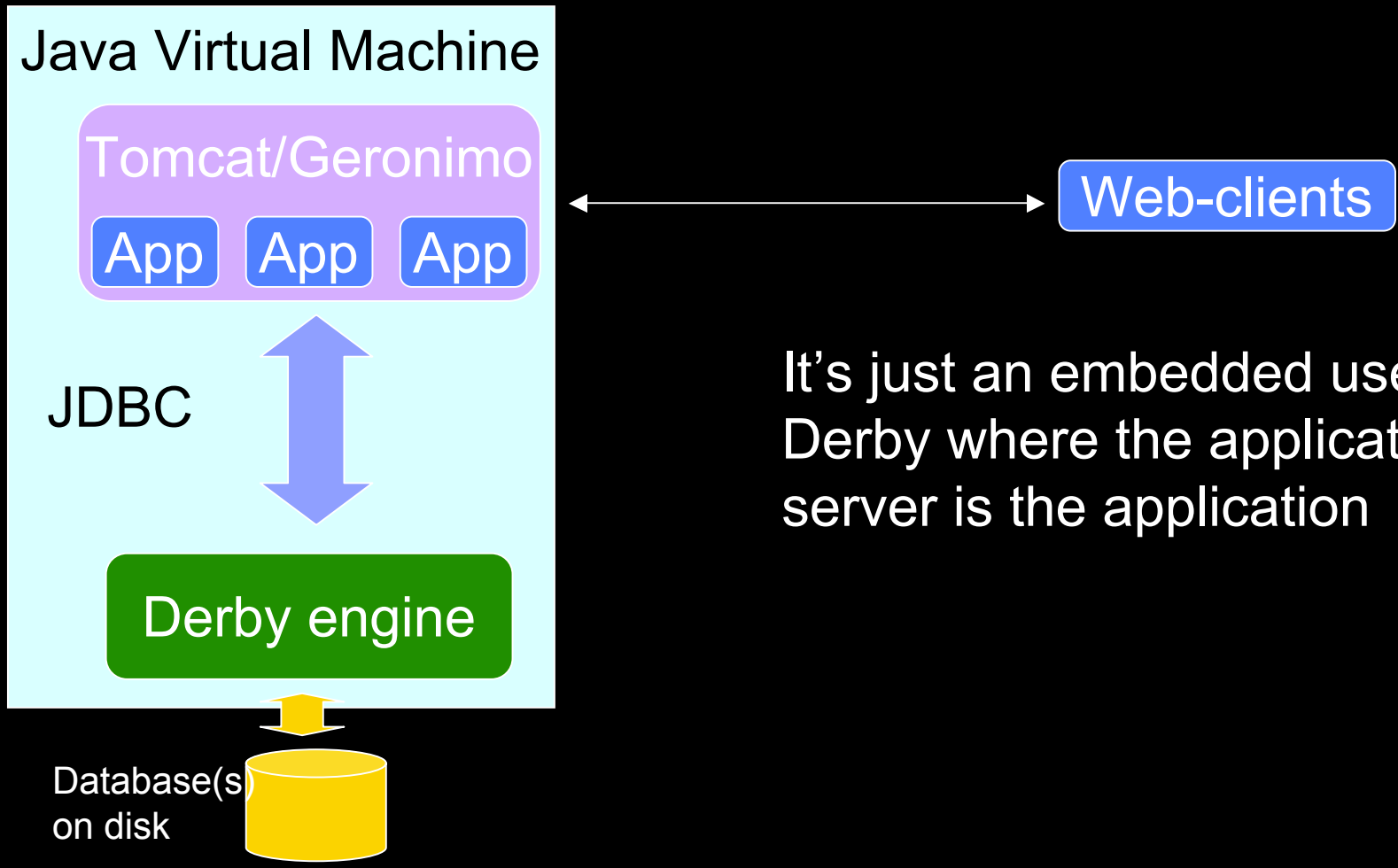
- **OpenDRDA**
  - opendrda.sourceforge.net
  - Low (almost zero) activity in project
- **JTOpen**
  - oss.software.ibm.com/developerworks/projects/jt400
  - IBM's open JDBC driver for AS/400 platform, may be too specific to AS/400 for Derby
  - Could form basis of Derby client???
- **RMIJDBC**
  - rmijdbc.objectweb.org
  - Alternative to Derby's network server

Application Servers

Derby & J2EE Application Servers

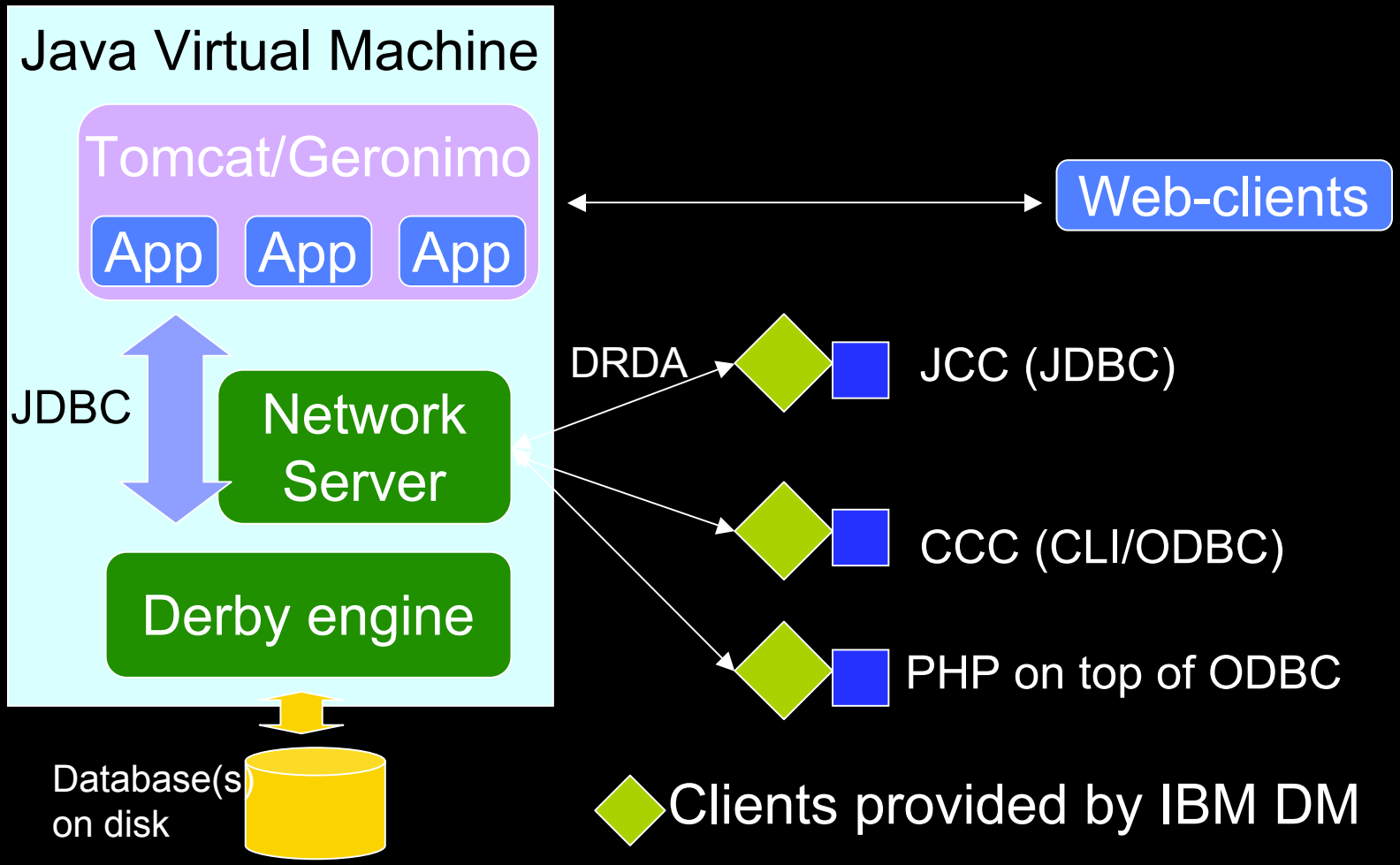
- **Complete JDBC driver for J2EE 1.3 and 1.4**
- **Complete application server and database in a single package (node)**
- **Web/Application server can be the mechanism to provide network access to a database through business logic with Servlets, JSPs, EJBs**

Embedded in Application Server



It's just an embedded use of Derby where the application server is the application

Add in Network Server if required



Derby Tooling

Basic Tools

- **Derby provides three command line Java tools**
- **ij – SQL scripting tool**
  - JDBC neutral, can be used against other JDBC drivers
- **dblook – schema extraction tool for Derby**
- **sysinfo – Derby version information**
  - Output useful for bug reporting in Jira

Example ij script

```
driver 'org.apache.derby.jdbc.EmbeddedDriver';
connect 'jdbc:derby:db1;create=true';

-- comments supported
-- statements can be spread over multiple lines
CREATE TABLE ADDRESS (ID INT, STREET VARCHAR(60),
CITY VARCHAR(60), STATE CHAR(2), ZIP CHAR(10));

CREATE INDEX AD_CITY ON ADDRESS(CITY,STATE);
```

- Note use of semi-colons to terminate statements, handled by ij, not Derby

ij extras

- **Also used to execute test-cases, thus supports more commands to allow JDBC interaction**
- **COMMIT/ROLLBACK**
- **AUTOCOMMIT {ON|OFF}**
- **GET CURSOR**
- **PREPARE name AS 'SQL-statement'**
  - EXECUTE name USING 'SQL-QUERY'
- **RUN 'script'**
- **Multiple named connections**
  - Connect 'jdbc:derby:sales' as SALES
- **Note, these are ij commands and not SQL statements for Derby**

dblook

- **java org.apache.derby.tools.dblook -d jdbc:derby:cs1**

```
-- Timestamp: 2004-10-15 10:59:33.553
  -- Source database is: cs1
  -- Connection URL is: jdbc:derby:cs
  -- appendLogs: false
  -- ----------------------------------------------
  -- DDL Statements for schemas
  -- ----------------------------------------------
  CREATE SCHEMA "SR";
  -- DDL Statements for tables
  CREATE TABLE "SR"."ADDRESS" ("ID" INTEGER NOT NULL, "STREET"
  VARCHAR(60), "CITY" VARCHAR(60), "STATE" CHAR(2), "ZIP" INTEGER);
  -- DDL Statements for indexes
  CREATE INDEX "SR"."AD_CITY" ON "SR"."ADDRESS" ("CITY", "STATE");
  -- DDL Statements for keys
  -- primary/unique
  ALTER TABLE "SR"."ADDRESS" ADD CONSTRAINT "SQL041015105817410"
  PRIMARY KEY ("ID");
```

sysinfo

- **java org.apache.derby.tools.sysinfo**

```
------------------ Java Information ------------------
  Java Version:    1.4.1_07
  Java Vendor:     Sun Microsystems Inc.
  Java home:       c:\work\p4\mi8\jdk141\jre
  Java classpath:  jars/insane/derby.jar
  OS name:         Windows 2000
  OS architecture: x86
  OS version:      5.0
  Java user name:  djd
  Java user home:  C:\Documents and Settings\djd
  Java user dir:   C:\work\p4\mi8\opensource
  --------- Derby Information --------
  [C:\work\derby\jars\insane\derby.jar] 10.0.2.0 - (47118:47119M)
  ------------------------------------------------------
  ----------------- Locale Information -----------------
  ------------------------------------------------------
```

- **Useful info to put in Jira reports**

Derby - Summary

Derby Summary

- **Easy to use, embedded, run anywhere database**
- **Standards based**
- **Security options for dispersed data**
- **Network server for network connectivity**
- **Easy integration with J2EE application servers**
- **Pure Java and complete database engine**

Community

- **Existing Cloudscape developers all new to open source**
- **All enjoying the experience**
- **Seems to be going well**
  - *I would like to express my joy with this community. Every questions and/or suggestion that I have made has actually led to conversation that moved the item forward. I have been following all of the threads, not just mine, and this seems to be the general state of affairs. The team are real doers and it is a pleasure a watch the progress as this project comes together. Kudos and thanks for the great efforts.*
  - Joel Rosi-Schwartz