Sun Java DB,

a small, easy to use, pure Java RDBMS

Bernt Marius Johnsen
Database Technology Group
Sun Microsystems
Bernt.Johnsen@Sun.COM



Database Technology Group

- Database Technology Group
  - Responsible for database technology within Sun
  - Trondheim, California, Bangalore
- Databases
  - HADB (aka. Clustra)
  - Java DB (aka. Derby)
  - Testing & evaluation of other databases (e.g. PostgreSQL)
  - Evaluation of databases for internal Sun use
  - New DB technologies

Java DB Overview

Ready for the challenge?



What is Java DB?

- Sun's supported distribution of Apache Derby
- Open Source Database Technology
  - Pure Java
  - Easy to use
  - Small footprint
  - Standards based
  - Complete relational database engine
  - Secure

Three organizations. Three brands. One product













A short Java DB History

- 1996: Cloudscape founded
- 1997: JBMS Released
- 1999: Cloudscape acquired by Informix
- 2001: IBM acquired DB part of Informix
- 2004: IBM donated Cloudscape to Apache as Derby
- July 2005: Derby graduated from Apache Incubator
- December 2005: Sun announces Java DB
- Current stable release: 10.1.3.1

Complete Relational Engine

- Tables
- Indexes
- Views
- Triggers
- Joins
- Procedures *(Java)*
- *Functions (Java)*
- Temporary tables
- Foreign keys
- Constraints
- Cursors
- Transactions
- Isolation levels
- ACID

Complete Relational Engine contd.

- Multi-user
- Deadlock detection
- Crash recovery
- Backup & restore
- Cost based optimizer
- Data caching
- Statement caching
- Write ahead logging
- Group commit

Complete Relational Engine contd.

- Multiple databases per system
- Multiple systems per read-only database
- Embedded
  - application in same VM as engine
- Client/Server
  - application in another VM than the engine
  - somewhere else in the network

Pure Java

- Database code written in Java
- Write Once Run Anywhere
  - Requires a J2SE 1.3, J2SE 1.4, J2SE 5.0 or Java SE 6 virtual machine
  - Any hardware, any operating system, any vendor
- Single binary does run everywhere
  - Solaris, Solaris x86, Linux, FreeBSD, NetBSD, OpenBSD, MacOs, Windows, AIX, Z/OS, AS400, OS/390, …
- Database on-disk format is platform independent

Small Footprint

- Engine jar file is around 2MB
  - 600kB when compressed with Pack200
  - Optional jar files
    - Network server ~150kB
    - Tools ~200kB
- Runtime memory use
  - Dependent on application, data caching etc.
  - Can run when Java heap memory restricted to 4MB
  - Runs on machines with only 16MB physical memory

Java DB Security

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

Note: SQL users and GRANT/REVOKE will be available in 10.2

Standards Based

- SQL
  - SQL92, SQL99, SQL2003, SQL/XML, …
- Java
  - J2SE 1.3, 1.4, 5.0, Java SE 6
  - J2ME/OSGi
- JDBC
  - 2.0, 3.0, 4.0
  - J2EE – Passed CTS for J2EE 1.4 & 1.3
  - JSR 169 (JDBC for CDC)
- DRDA V3
  - OpenGroup (http://www.opengroup.org/)

Java DB Tools

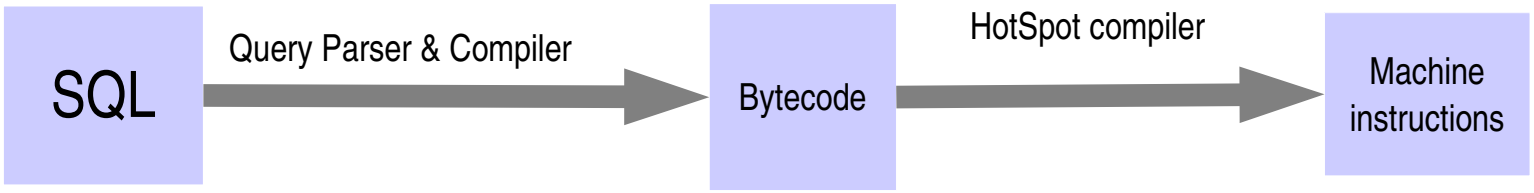
- ij – SQL scripting tool
  - JDBC neutral, can be used against other JDBC drivers
- dblook – schema extraction tool for Derby
- sysinfo – Derby version information
  - Output useful for bug reporting in Jira

Architecture: Fundamentally solid

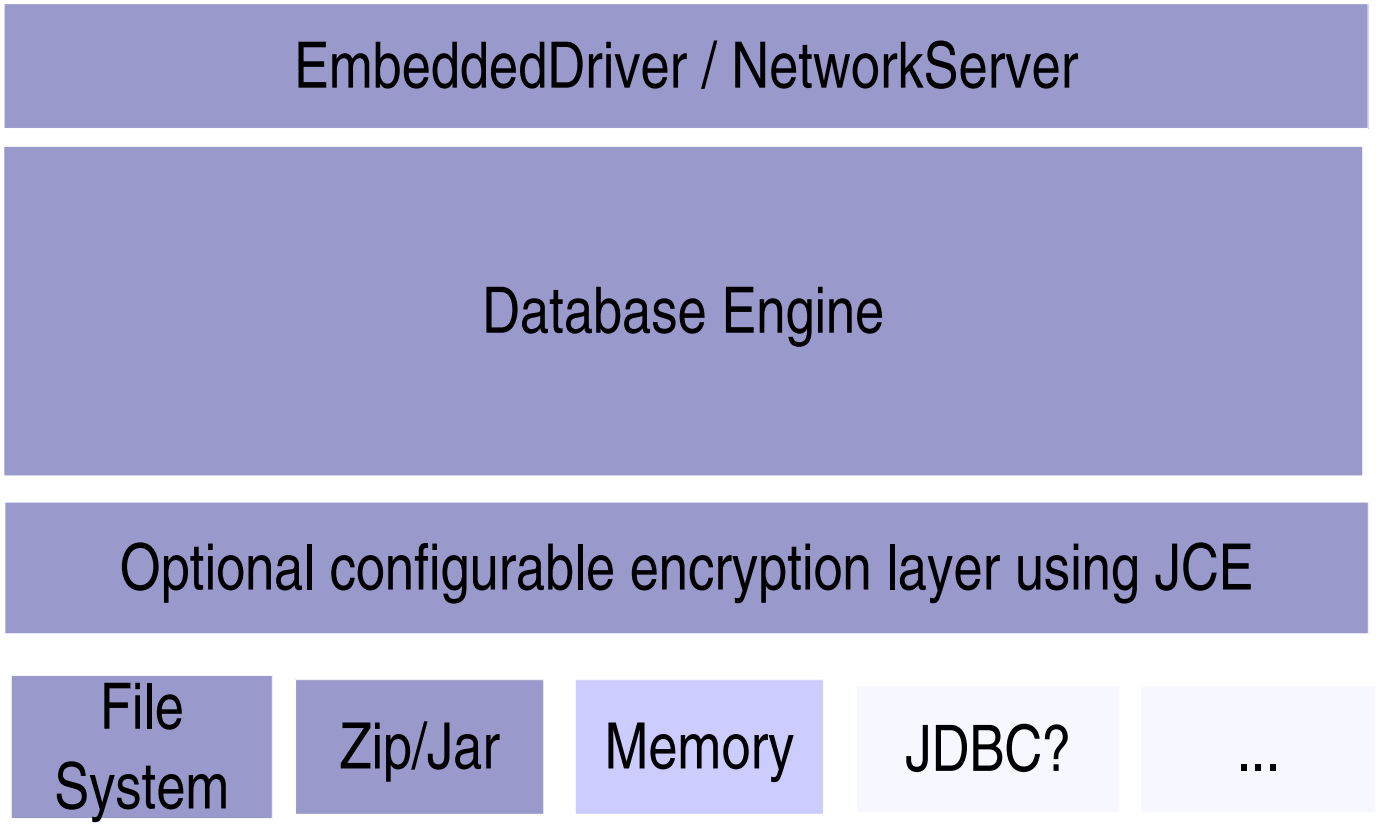
- Straight, solid state-of-the art technology
- Well proven techniques (text-book)
  - Aries
  - Indices are B-trees
- Modular architecture

No separate SQL virtual machine

- SQL compiled into Java byte-code
- Run on standard Java VM
  - Pro: Fast. Utilizes HotSpot
  - Con: Compilation + class-loading



Pluggable Storage Architecture

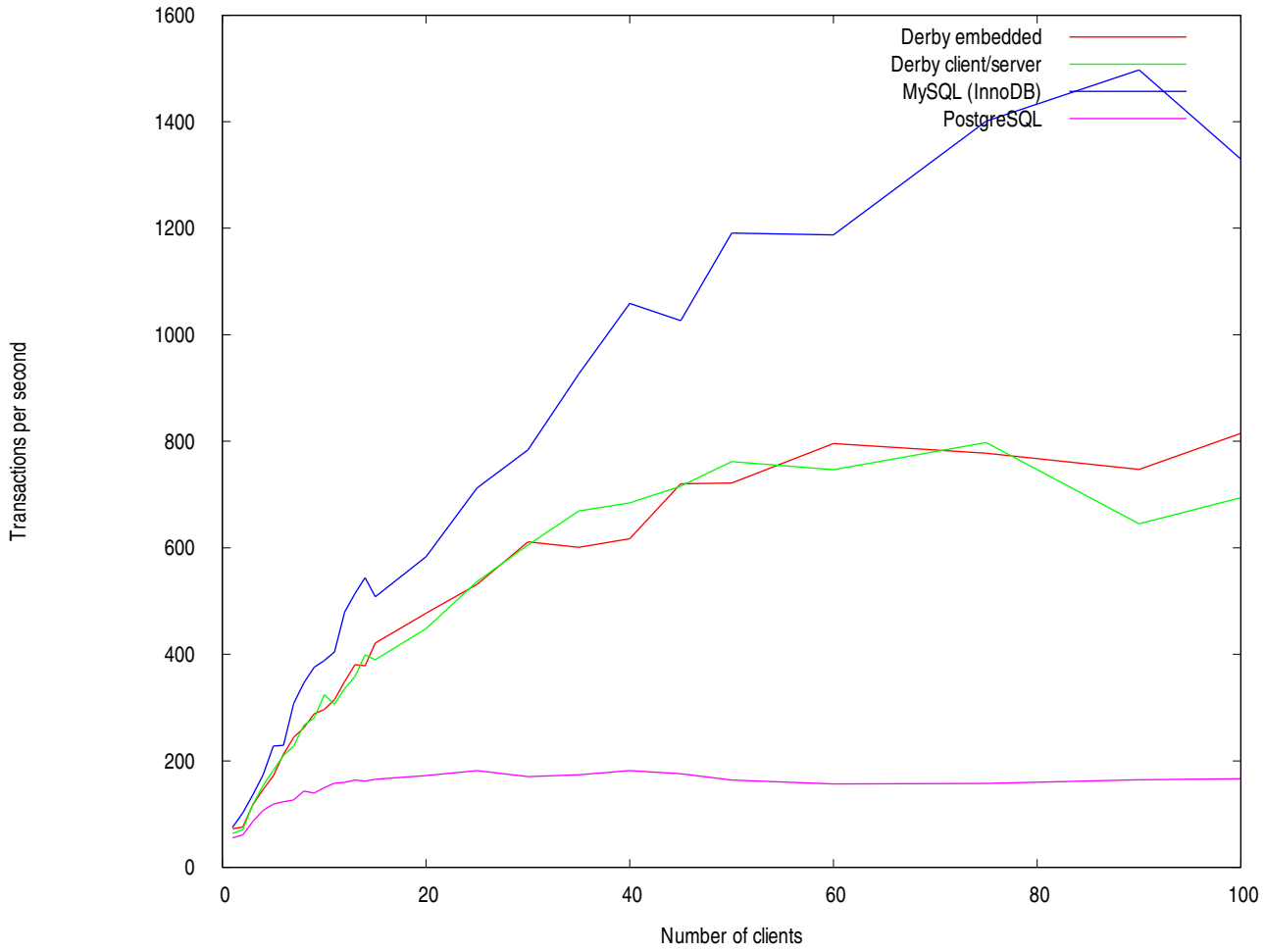


Java DB performs!

- Java DB performs well
  - Comparable to competition
- Java is not slow!
  - Just In Time (JIT) compilers compile interpreted byte code to native machine code
- We have studied and compared the performance of
  - Java DB
  - MySQL
  - PostgreSQL

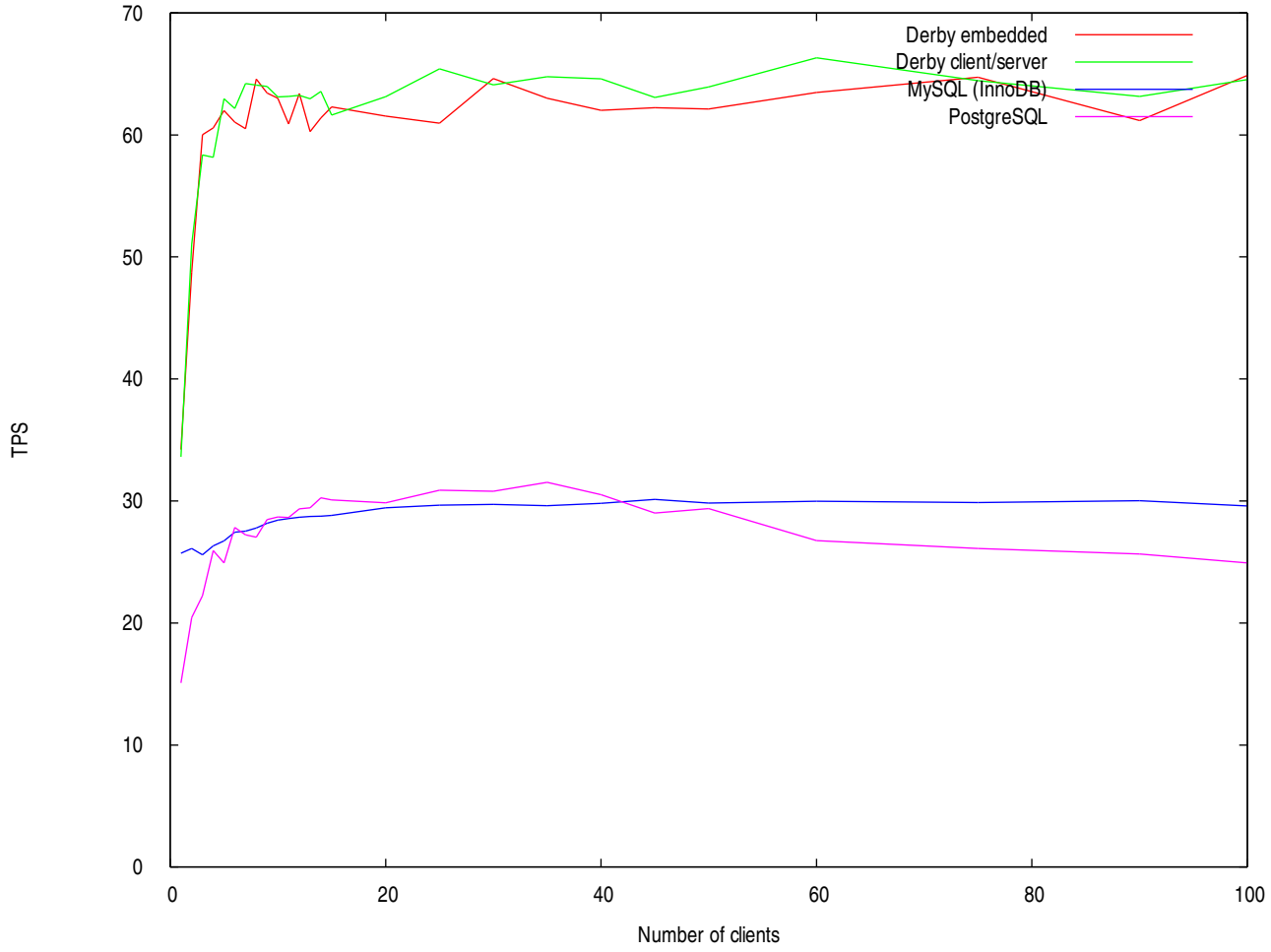
TPC/B-like load, In-memory DB

DB 10MB, buffer 50MB, 400 branches



TPC/B-like load, Disk-bound DB

DB 10GB, buffer 64MB, 400 branches



Performance hints

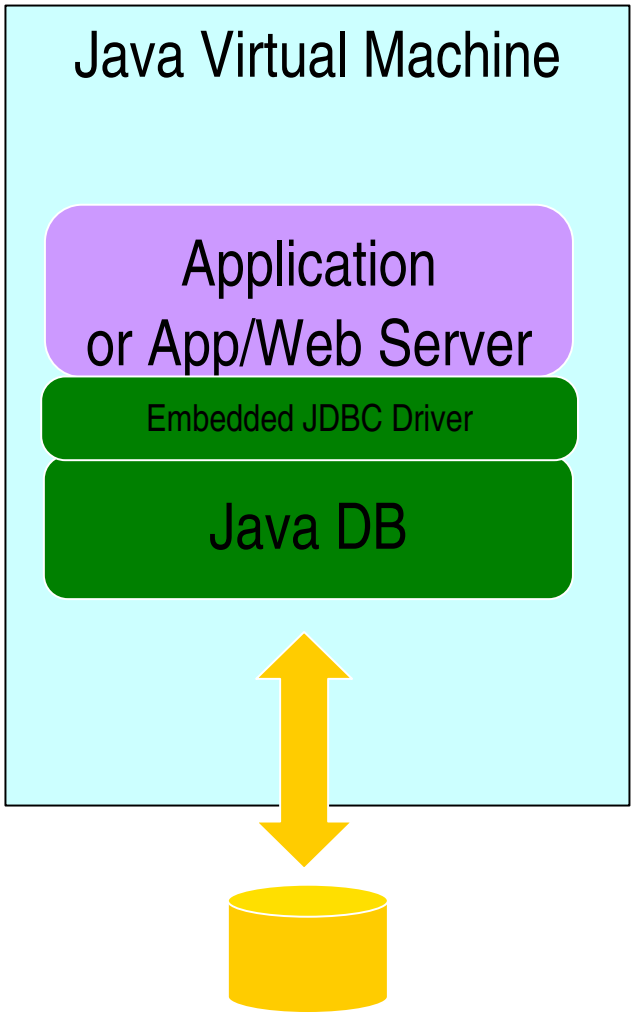
- Use (and reuse) prepare statements
  - Not: `"SELECT a FROM t WHERE b=4" + val`
  - But: `"SELECT a FROM t WHERE b=?"`
- Put DB log and data on separate disks
  - Specify the `logDevice` attribute when creating DB
- Tune page cache size (default 4 MB)
  - `derby.storage.pageCacheSize`
- Use indices to avoid table scans
  - Check query plans (`derby.language.logQueryPlan=true`)

Java DB Usage

A powerful tool



Embedded Java DB



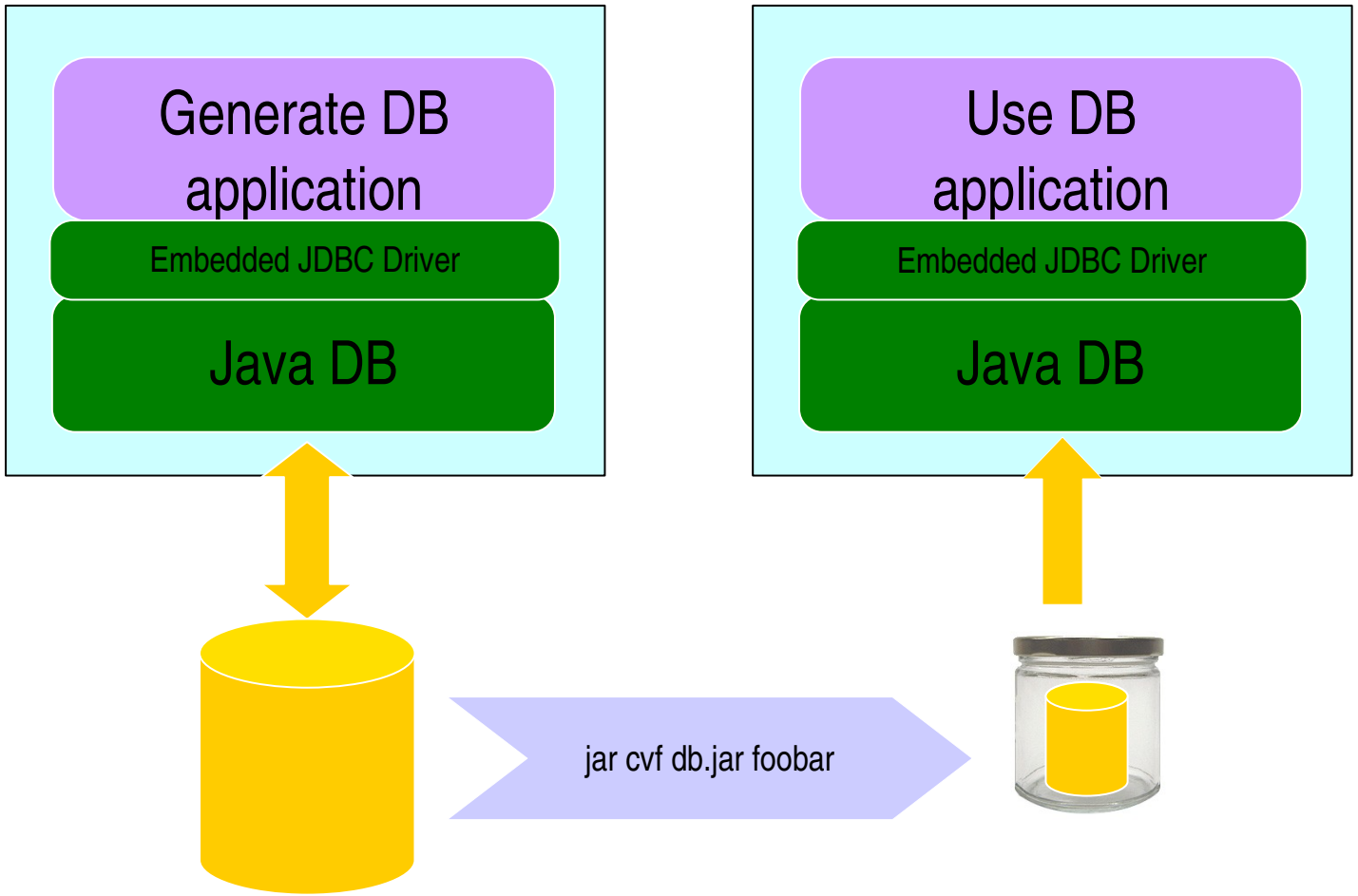
- Easy to use
- Fast
- Zero administration

Embedded: Zero Administration

```
java -classpath derby.jar ...
```

```
Class.forName
  (“org.apache.derby.jdbc.EmbeddedDriver”);
DriverManager.getConnection
  (“jdbc:derby:foobar;create=true”);
```

Read only DB-in-a-jar (or zip)



Use of DB-in-a-jar

```
java -classpath derby.jar ...
```

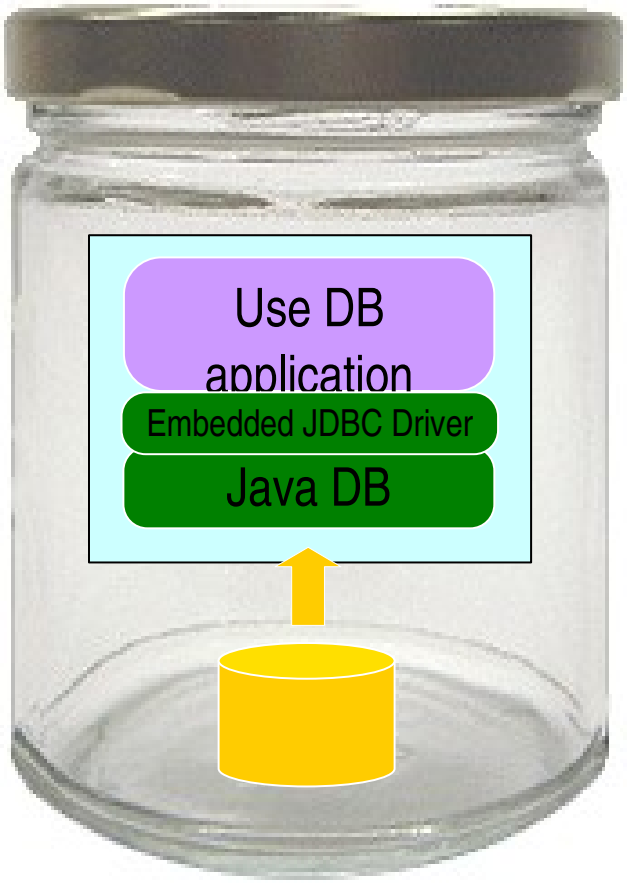
```
DriverManager.getConnection
  ("jdbc:derby:(db.jar)foobar");
```

... or ...

```
java -classpath derby.jar:db.jar ...
```

```
DriverManager.getConnection
  ("jdbc:derby:classpath:foobar");
```

All-in-one-jar



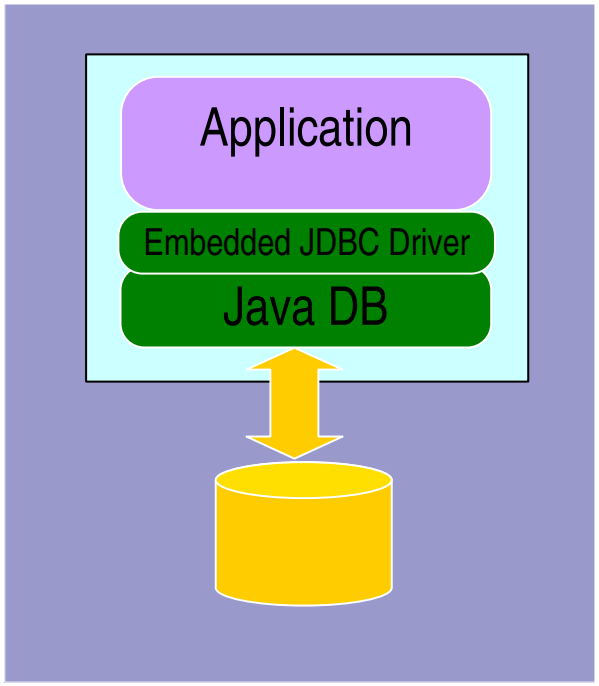
Use of All-in-a-jar

```
jar xf derby.jar
jar cf all-in-one.jar org myapp foobar
```

```
java -classpath all-in-one.jar ...
```

```
DriverManager.getConnection
  (“jdbc:derby:classpath:foobar”);
```

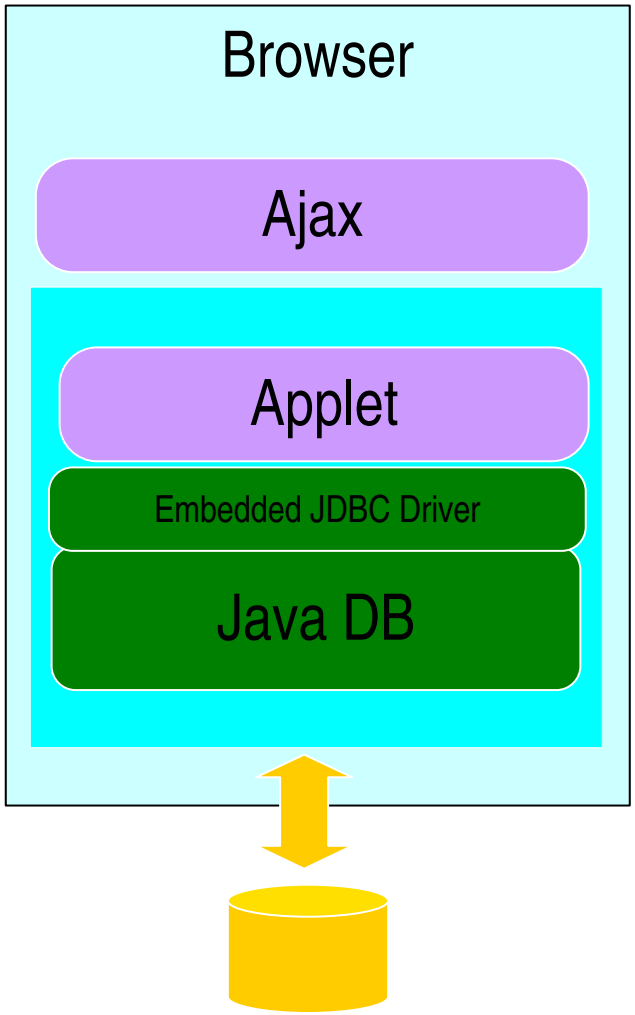
DB-on-a-stick


Application
Embedded JDBC Driver
Java DB


SanDisk
1.0GB
micro SD

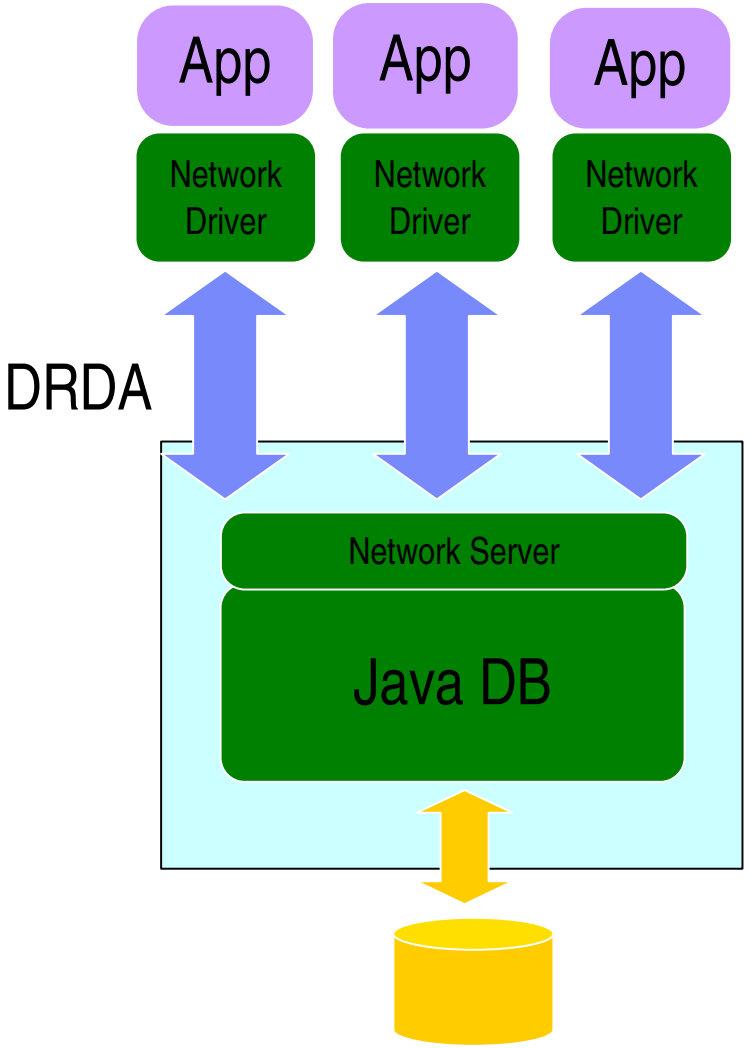


DB in a Browser



- Small enough to fit in a browser
- Fast enough for interactive use
- See demo at http://developers.sun.com/prodtech/javadb/

Client/Server



- Nearly zero administration
- Standard protocol (DRDA)
- Many apps use one DB

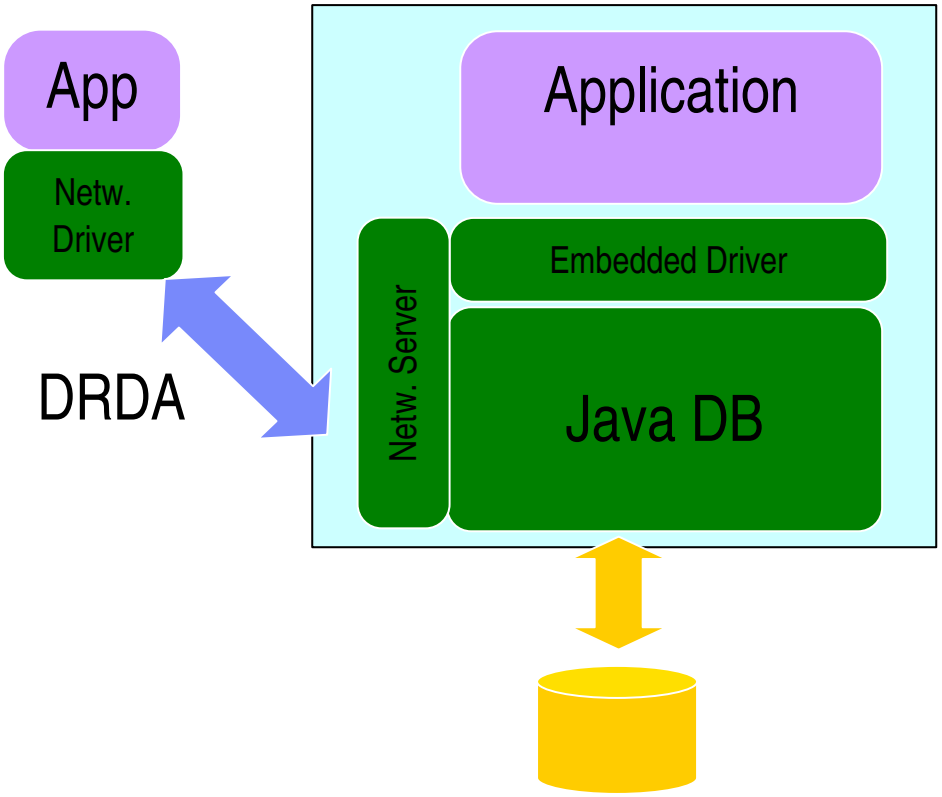
C/S: Nearly Zero Administration

```
java -jar derbynet.jar start -p 12345 -h
  ahost
```

```
java -classpath derbyclient.jar ...
```

```
Class.forName
  (“org.apache.derby.jdbc.ClientDriver”);
DriverManager.getConnection
  (“jdbc:derby://ahost:12345/foobar”);
```

Combined Embedded/Server



- Nearly Zero Administration
- No application change

Combined Embedded/Server: Server

```
java -Dderby.drda.startNetworkServer=true
  -Dderby.drda.host=ahost
  -Dderby.drda.port=12345
  -classpath derbynet.jar ...
```

```
Class.forName
  (“org.apache.derby.jdbc.EmbeddedDriver”);
DriverManager.getConnection
  (“jdbc:derby:foobar;create=true”);
```

Combined Embedded/Server: Server #2

```
java -classpath derbynet.jar ...
```

```
Class.forName (“org.apache.derby.jdbc.EmbeddedDriver”);

DriverManager.getConnection
  (“jdbc:derby:foobar;create=true”);

.......

NetworkServerControl server = new NetworkServerControl
          (InetAddress.getByName(“ahost”), 12345);

server.start(null);

.......

server.shutdown();

.......
```

Combined Embedded/Server: Client

```
java -classpath derbyclient.jar ...
```

```
Class.forName
  (“org.apache.derby.jdbc.ClientDriver”);


DriverManager.getConnection
  (“jdbc:derby://ahost:12345/foobar”);
```

Java DB Scaling

- In principle unlimited: No architectural constraints
- We have tested
  - Databases up to 300GB
  - Up to 100 active connections

Java DB Horizontal Scaling and HA

- No built-in support
- May be achieved somewhat with Sequoia (C-JDBC)
  - Not like “the real thing” (e.g HADB and other “hardcore” HA solutions)
  - Low performance
  - Limited JDBC/SQL support
  - Immature
- “Building Highly Available Database Applications for Apache Derby”, ApacheCon 2005.
  - http://c-jdbc.objectweb.org

In-memory DB

- What do you mean?
  - Java DB caches data in memory.
- Durability comes at a cost
  - Persist to disk as you go: Performance penalty
  - Persist at shutdown: Risky
- Possible to run Java DB with less durability:
  - `-Dderby.system.durability=test`
- SQL RDBMS
  - B-trees designed for efficient disk storage & retrieval
  - ACID (Atomicity, Consistency, Integrity, Durability)

The Future: Java DB 10.2

- Scrollable updatable result sets
- Grant/Revoke
- Online backup
- Stronger Network Authentication
- XML support for XPath
- ... plus... unary +/- in parameters, FOR UPDATE, Client timeout, Optimizer directives, trigonometric functions, internationalizing, invoke procedures from triggers, GROUP BY supports expressions, identity value restart, etc...

The Future: JDK 6 and JDBC4

- We are bundling Java DB in JDK 6
- We will support JDBC4 and have implemented
  - Ease-of-development
  - Wrapper patterns
  - SQLException subclasses
  - Statement events
  - Lengthless streaming

Issues we are working with

- Performance
  - Already 40% improvement single row select client/server
- Standards compliance
- Standards completeness
- In-memory store
- SSL/TLS
- JMX

Java DB Usage

- Support in tools
  - e.g. Netbeans, Eclipse, Java Studio Creator
- Persistence
  - e.g. Apache JDO, Apache OJB, Hibernate, JPOX JDO
- Supported by Application Servers
  - e.g. Sun's J2EE SDK, JBOSS, Geronimo, Tomcat
- Supported by frameworks
  - e.g. Apache Cocoon

See extensive list at http://wiki.apache.org/db-derby/UsesOfDerby

Community

- http://db.apache.org/derby
  - Download, read docs
- JIRA http://issues.apache.org/jira/browse/DERBY
  - Report bugs, submit patches
- derby-user@apache.org
  - Discuss experience, get help
- derby-dev@apache.org
  - Discuss developer issues

Q&A

Bernt Marius Johnsen

Bernt.Johnsen@Sun.COM

bernt@apache.org



What about the much faster and pure Java xxxxx in-memory DB

A C I D !