Apache Derby – a 100% Java Open Source RDBMS

Using Apache Derby in the real world

Victorian AJUG, Australia
28th August 2008

Chris Dance

Introduction

- Chris Dance
- Director and Found of PaperCut Software
- PaperCut Software:
  - Print control software
  - Based in Melbourne, Australia
  - Commercial software deployed to tens-of-thousands of schools, universities and businesses in 60+ countries
  - A commercial “download and install” application mostly written in Java

Objective

- Provide a summary of Apache Derby and its key features.
- Arm you with the knowledge to help you make the decision to use Apache Derby.
- Not a HOWTO guide. Derby already has great documentation.
- The experiences with using Derby in a real world application.

Derby Project History

- Started as Cloudscape in 1996
- Acquired by Informix… then IBM…
- IBM contributed code to Apache project in 2004
- An active Apache project with conservative dev.
- DB2 influence. Many of the same limits/features
- Has Sun's stamp of approval – Java DB and in JDK 6

What is Apache Derby

- A 100% Java
- Small disk footprint (JAR size ~2Mb) (Memory varies)
- JDBC 4 support
- Very simple to embed in an application
- Zero administration (explain caveats later!)
- Can support network client/server topology
- Not a toy! Its packed full of goodies you would never expect.

Apache Derby

Networked or Embedded

- Embedded
  - Run in-process and in-thread
  - Low overhead
  - Conservative defaults (minimal memory use)
  - Like SQLite (this is the worlds most deployed database)
- Networked
  - TCP based network server
  - Accepts connections from other processes or hosts
  - Lightweight client driver
  - Multiple concurrent connections/applications/hosts
  - Like Oracle, MySQL, or Postgres

Apache Derby

Why use Derby?

- Databases are now used everywhere (browsers, calendaring apps, iPhone, most websites, etc.)
- Every database has different features and target different (but often overlapping) database domains.

Right tool for the job:

"Oracle sucks. I tried to use it to back my new contacts management application and it took 10x more RAM that my own little application used!"

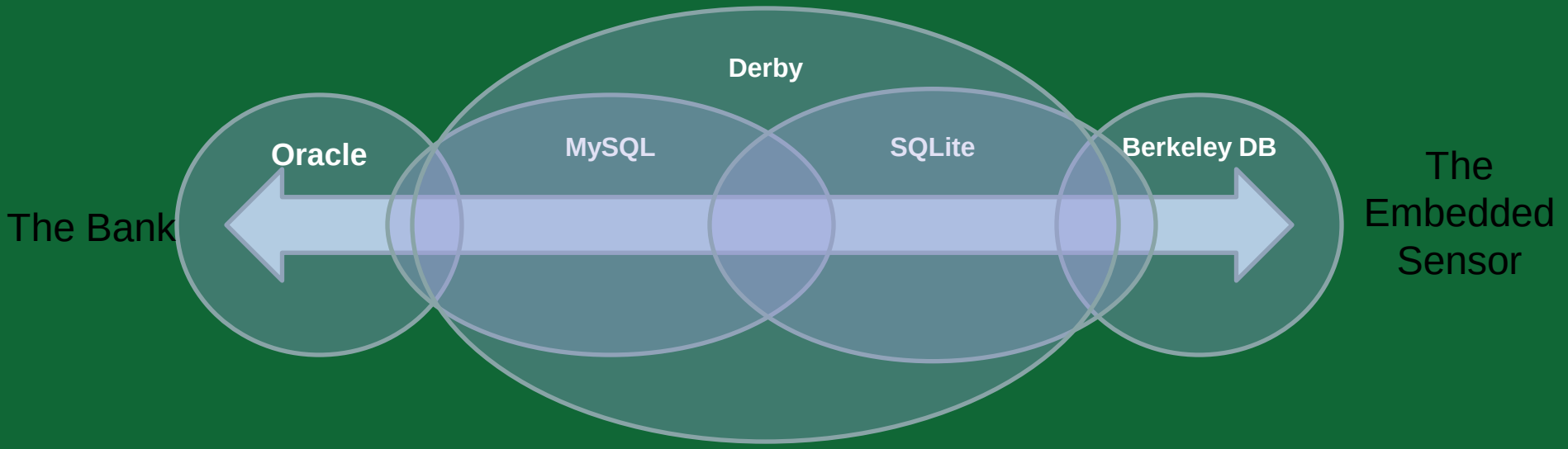
"MySQL Rules. I needed a simple database for my PHP app and it was already sitting there on my ISP's server all set up and ready to go."

… use a database in its target domain and it will excel!

Apache Derby

Domain of Applicability

If we stop comparing features but look at the database's domain as a "feature" itself, Derby is a winner.



Features

The Small End (Like SQLite):

- Embeddable with small footprint
- Close to zero administration
- Simple to use: Just drop the JAR into your application

The Big End (Like MySQL/Postgres):

- Fully ACID, real SQL, transactions, unicode, and good performance with large datasets
- Concurrency: row-level locking, exclusive, shared and update level locks
- Option for Client/Server Topology
- Network and disk level encryption
- Support for triggers and functions (functions in Java)
- Even live replication, failover, and ability to natively store JAR files!

Differences

The Small End (Like SQLite):

- Java (but this is an AJUG so we'll skip this ☺)
- Not quite zero admin

The Big End (Like MySQL/Postgres):

- Less fiddly bits to keep the a DBA busy.
- Basic tools support (ij, dblook, basic query plan)
- Minimal GUI admin tools
- Design for packaging rather than a standalone double-click installer

The Code...

- Put derby.jar on your ClassPath
- The code:

```
// Load the derby driver
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

// Establish the connection
Connection conn = DriverManager.getConnection(
                          “jdbc:derby:mydatabase”);

// Use the connection as usual
```

Apache Derby

Java Integration

- Will work just like any other database:
  - JDBC 4
  - Hibernate Support
  - Spring Transactions
  - Connection Pools (C3P0)
  - XA Support – two phase commit

Apache Derby

In the real world...

- Used in PaperCut
  - A print control application (e.g. quota student printing in schools/universities)
  - Cross platform (Mac, Linux, Novell and Windows)
  - In use in tens-of-thousands or organizations
- Derby first used in 2005
- 90% stick with Derby (7% SQL Server, 2% Postgres, 1% Oracle)
- Non-trivial system supporting transactions, live backups, reports, concurrent access.
- Datasets in the millions of rows supporting hundreds of concurrent connections.
- Very few corruptions (there was a corruption bug in one derby release)
- Zero DBA administration – most have no idea they are running Derby

Apache Derby

Real advantage...

- Derby faster than clustered Microsoft SQL server! **
- Scenario:
  - Miami Dade College, the largest college in the US with 350,000 students & staff
  - Student card numbers hosted in external system
  - Card numbers need to be periodically batch imported/updated
  - Very “chatty” SQL operations (check current card number and update if required)
  - Most time spent in network round trip – network latency and data marshaling
- Same data and scenario with embedded Derby was twice as fast
- Logic needed to be moved down into the database layer to improve SQL Server performance

Apache Derby

Real problems...

- Performance
  - Directional indexes
  - Comparable to Postgres and MySQL with smaller data sets.
  - Sudden performance problems observed as datasets grew
    - Optimizer switching from hash joins to loop joins as data hit a threshold
  - Non-default configuration required for larger datasets:
    - Increased page size to 32k (derby.storage.pageSize to 32k)
    - Increase page cache size (derby.storage.pageCacheSize)
    - Increased memory available for joins (derby.language.maxMemoryPerTable to JVM max ~15% of RAM)
  - Derby community/developers were very helpful and responsive
- Not quite zero administration
  - Statistics not updating automatically (Added explicit calls to update stats)
  - Fragmentation after deletion (Implemented monthly offline table compresses)

Apache Derby

What's missing?

- This is my chance to put my ideas to the Derby developers!
- The out of the box "domain of applicability" could be expanded with profiles (or even simply more documentation!).
    - Add a config and/or JDBC connection string setting that sets different defaults to fit different environments:

      ```
      jdbc:derby:MyDatabase;profile=largedataset;create=true
      ```

    - Profiles set logical performance defaults

      ```
      largedataset, smalldataset, minimizemem, auto
      ```

- Aim for zero administration:
  - Auto generate database statistics.
  - Automatic compression / cleanup of tables (like Postgres auto vacuum).

Apache Derby

Why use Derby for your next project?

"Think of SQLite not as a replacement for Oracle but as a replacement for fopen()."

- D. Richard Hipp, creator of SQLite

- Database are used everywhere today
- Derby is the best embedded or “prepackaged” database option for Java
- Some ideas:
  - A pre-configured demo database for your existing application
  - A testing database and a way of verifying database independence
  - Replace your existing file based storage

Resources

Derby Website:

http://db.apache.org/derby/

Derby Mailing List:

http://db.apache.org/derby/derby_mail.html

PaperCut website and download (how it's used in our print quota application):

http://www.papercut.com/