COSC 416
NoSQL Databases

Berkeley DB

James, Cody, Justin
University of British Columbia Okanagan

Berkeley DB

Berkeley DB (BDB) is a software library that provides a high-performance embedded database for key/value data.

It is written in C with API bindings for C++, C#, PHP, Java, Perl, Python, Ruby, Tcl, Smalltalk, and most other programming languages.

Key components:

- Stores arbitrary key/data pairs as byte arrays
- Supports multiple data items for a single key
- It is not a relational database but can be accessed as one with an SQLite wrapper

Programs that Use Berkeley DB

Berkeley DB provides the storage and retrieval system of several servers, database systems, and many other free/open source applications. Notable software that use Berkeley DB for data storage include:

- Bitcoin
- Memcachedb
- MySQL
- Red Dwarf

Berkeley DB

Using Berkeley DB you can:

- Reduce time to market
- Reduce development costs
- Simplify data storage on mobile devices
- Lower the cost of deployment
- Eliminate costly re-writes as your system grows
- Eliminate costly administrative overhead
- Eliminate data loss and corruption
- Provide internet-scale, highly-available services

Berkeley DB Essentials

Berkeley DB is also used as the common name for three distinct products; Oracle Berkeley DB, Berkeley DB Java Edition, and Berkeley DB XML. These three products all share a common ancestry and are currently under active development at Oracle Corporation.

Berkeley DB Essentials

Berkeley DB:

- Key-value store programmed in C. (Berkeley DB official documentation uses the term key-data in place of key-value.)

Berkeley DB Java Edition (JE):

- Key-value store re-written in Java. Can easily be incorporated into a Java stack.

Berkeley DB XML:

- Written in C++, this version wraps the key-value store to behave as an indexed and optimized XML storage system.

Interacting With Berkeley DB

- Data Persistence Layer (DPL)
- Base API
- Berkeley DB SQL (BDB SQL)

DPL

Store Java objects in an underlying database

Objects can then be accessed via EntityStore

Objects must be decorated with Java annotations

- `@Entity`
- `Public class Employee {`
  - ⇨ `...}`

Key must be unique, declared with annotations

- `@PrimaryKey`
- `int id`

Can also index on secondary keys

Better for applications with relatively static schema

Base API

Record keys and data stored as byte arrays

Passed to and from database using DatabaseEntity instance

Data stored internally with B-tree

Primary key constraint relaxed

- Gets will return first record
- Cursors can be used to iterate over all results
- Updates/deletes require cursor to find desired record

Handles dynamic schemas better than DPL

BDB SQL

Your interaction with the BDB SQL interface is almost identical to SQLite. You use the same APIs, the same command shell environment, the same SQL statements, and the same PRAGMAs to work with the database created by the BDB SQL interface as you would if you were using SQLite.

Features Not Supported in SQLite

- RIGHT and FULL OUTER JOIN
  - LEFT OUTER JOIN is implemented, but not RIGHT OUTER JOIN or FULL OUTER JOIN
- Complete ALTER TABLE support
  - Only the RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported
- Complete trigger support
  - FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers
- Writing to VIEWs
  - VIEWs in SQLite are read-only
- GRANT and REVOKE
  - GRANT and REVOKE commands are meaningless for an embedded database engine

Accessing BDB SQL Databases

BDB SQL databases can be accessed using a number of different drivers, applications and APIs. Only some of these are supported by all major platforms, as identified in the following table.

	UNIX/POSIX	Windows	Windows Mobile/CE	Android	iOS
DBSQL Library	x	x	x	x	x
DBSQL Shell	x	x	x	x	x
ODBC	x	x			
JDBC	x	x			
ADO.NET		x	x		

The Journal Directory

When you create a database using the BDB SQL interface, a directory is created alongside of it. This directory has the same name as your database file, but with a -journal suffix.

This directory contains files that are very important for the proper functioning of the BDB SQL interface. Do not delete this directory or any of its files unless you know what you are doing.

In Berkeley DB terms, the journal directory contains the environment files that are required to provide access to databases across multiple processes.

Pros and Cons

Pros:

- Good for long-term storage
- Fast lookups with memory cache
- Portable across platforms
- Size capabilities (256TB/DB and 4GB/record)

Cons:

- Does not support constraints
- Portable across differing endian-ness but performance worse with little endian-ness

Resources

Berkeley DB Documentation

http://docs.oracle.com/cd/E17076_02/html/index.html

SQLite Tutorial

http://souptonuts.sourceforge.net/readme_sqlite_tutorial.html

Berkeley DB SQL: The Absolute Basics

http://docs.oracle.com/cd/E17076_02/html/bdb-sql/dbsqlbasics.html