Embedding SQL Engine to Your Application

Iwo Panowicz
Percona
What’s an Embedded Database?

- A library embedded in the application, that implements methods to access and manipulate data.
- A database running on an embedded computer mostly adapted for embedded purposes.
Libraries embedded in the application
What’s an Embedded Database?

- ... but why not just flat flies/xml/json?
  - creating a database management system is harder than you might thought
  - saving time,
  - not reinventing the wheel.
- Mobile applications
- IoT
- Microservices
Libraries

- Usually lightweight.
  - SQLite can be fit under 300KB and be made to run in minimal stack space (~4KB) and heap space (~100KB).
- Doesn’t require any connection to a database instance.
- Widely used in IoT and mobile applications.
- Low latencies.
- Usually used with much smaller data sets that with conventional DBMS.
- Help processing data
  - In-memory databases are often used as temporary databases that require no permanent storage.
- DBA not needed.
Sample use case

- Mobile contacts manager
- E-book reader database
- Web browser cookie storage
- … anything that stores data.
  - like airplanes
MySQL Embedded, libmysqld

- The embedded MySQL server library makes it possible to run a full-featured MySQL server inside a client application.
- The main benefits are increased speed and more simple management for embedded applications.
- Written in C/C++ and available only for C/C++.
- The API is identical for the embedded MySQL version and the libmysqlclient.
- Deprecated in 5.7 and fully removed in 8.0.
MySQL Embedded, major restrictions

- No user-defined functions (UDFs).
- No networking (handled by MySQL).
  - No Replication.
- No Event Scheduler.
- No Performance Schema.
MySQL Embedded, restrictions

...  
#include "mysql.h"
MYSQL *mysql; MYSQL_RES *results; MYSQL_ROW record;
...
int main(void) {
    [1] mysql_library_init(num_elements, server_options, server_groups);
    [2] mysql = mysql_init(NULL);
    [3] mysql_options(mysql, MYSQL_READ_DEFAULT_GROUP, "libmysql_client");
    [4] mysql_options(mysql, MYSQL_OPT_USE_EMBEDDED_CONNECTION, NULL);
MySQL Embedded, restrictions

```c
[5 ] mysql_real_connect(mysql, NULL, NULL, NULL, "database1", 0, NULL, 0);
[6 ] mysql_query(mysql, "SELECT column1, column2 FROM table1");
[7 ] results = mysql_store_result(mysql);

while((record = mysql_fetch_row(results))) {
    printf("%s - %s \n", record[0], record[1]);
}

[8 ] mysql_free_result(results);
[9 ] mysql_close(mysql);
[10] mysql_library_end();

return 0;
```
SQLite

- **Small. Fast. Reliable. Choose any three.**
- Extremely popular embedded database systems. ProxySQL uses it.
- SQLite author claims that reads and writes small blobs 35% faster than the same blobs can be read from or written to individual files on disk using fread() or fwrite(). YMMV.
- SQLite database holding 10-kilobyte blobs uses about 20% less disk space than storing the blobs in individual files. YMMV.
SQLite

- Zero-Configuration apart for a schema.
- In-Memory databases
- Single Database Files
- SQL statements compile into virtual machine code
- Implements most of the SQL-92
  - basic support of triggers
  - basic support ALTER TABLE
SQLite
SQLite interface elements can be grouped into three categories:

- **List of objects**
  - list of all abstract objects/datatypes used (sqlite3_stmt).

- **List of Constants**
  - list of numeric constants (#define SQLITE_OK)

- **List of Functions**
  - List of all functions and methods (sqlite3_initialize).

Most applications only use a handful.
Tokenizer

- When a SQL statement is to be evaluated it is first sent to tokenizer.
- The tokenizer breaks the SQL text into tokens and hands those tokens one by one to the parser.
Parser

- The parser assigns meaning to tokens based on their context.
  - Assembles tokens into a parse tree.
- SQLite uses own solutions, called Lemon.
  - … which does the same job as YACC/BISON.
- Lemon is thread-safe.
- Lemon and its grammar file define the SQL language that SQLite understands.
Code Generator

- Code Generator analyzes the parser tree and generates opcode that performs the work of the SQL statement.
- The opcode is not SQLite’s API.
  - Details about the opcode change between releases.
- Sample opcode:

```sql
sqlite> EXPLAIN SELECT 1;

addr  opcode         p1  p2  p3  p4             p5  comment
----  -------------  ----  ----  ----  -------------  --  --------------
0     Init           0    1    0                    00  Start at 1
1     Integer        1    1    0                    00  r[1]=1
2     ResultRow      1    1    0                    00  output=r[1]
3     Halt           0    0    0                    00
```
Virtual Machine

- Executes the Code Generator program.
- Some functions are implemented/inlined as bytecode directly by the code generator:
  - `typeof()`, `coalesce()`
- Some opcodes that Virtual Machine is executing are self-altering.
B-Tree

- SQLite stores
  - each table in a separate Binary Tree
  - each index in a separate Binary Tree
- All Binary Trees are stored in a single file.
- The file format is stable and is guaranteed to be compatible moving forward.
Pager

- Records are stored in fixed-sized pages.
  - Page size can be any power of 2 between 512 and 65536.
  - The default is 4096 (4KB).
- It is responsible for reading, writing and caching pages.
- It provides the rollback and atomic commit abstraction.
- It takes care of locking of the database file.
OS Interface

- It is called VFS.
- VFS is what makes SQLite portable across operating systems
  - provides methods for reading/writing data from a file
  - obtaining randomness,
  - finding current time, etc.
- Whenever any of the other modules in SQLite needs to communicate with the operating system, they invoke methods in VFS.
Accessories

- All helpers, that SQL uses:
  - memory allocation,
  - caseless string comparisons,
  - and even own `printf()` implementation.
While it works great for many use-cases, it is not recommended for cases when:

- Data is separated from the application (client/server).
  - NFS is **NOT** an option.
    - but if you really need it then check if `fnctl()` works.
    - Async I/O might be faster for NFS.
- The dataset is very large or complex.
- High concurrency is needed.
SQLite
SQLite
Berkeley DB

- Initially released in 1996.
- Key-value database.
- Database objects can use various access methods: btree, hash, heap, queue, recno.
- Currently, Berkeley DB name is given to three different products:
  - Berkeley DB (C edition)
  - Berkeley DB Java Edition
  - Berkeley DB XML
Berkeley DB

- Each major release cycle has introduced a single new major feature. Most Notable features:

<table>
<thead>
<tr>
<th>Version</th>
<th>Feature Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.85</td>
<td>Last UCB (University of California, Berkeley) release</td>
</tr>
<tr>
<td>2.0</td>
<td>Transactions, recovery</td>
</tr>
<tr>
<td>3.0</td>
<td>Queue AM, POSIX threads, subdatabases</td>
</tr>
<tr>
<td>4.0</td>
<td>Secondary Indices</td>
</tr>
<tr>
<td>4.1</td>
<td>Replication (master-slave)</td>
</tr>
<tr>
<td>4.2</td>
<td>Encryption</td>
</tr>
<tr>
<td>4.4</td>
<td>Sequence numbers</td>
</tr>
<tr>
<td>4.5</td>
<td>MVCC</td>
</tr>
<tr>
<td>5.0</td>
<td>Full Text, R-Tries, SQL Api, JDBC</td>
</tr>
</tbody>
</table>
Oracle provides four Berkeley DB products:
  - Berkeley DB Data Store
  - Berkeley DB Concurrent Data Store
  - Berkeley DB Transactional Data Store
  - Berkeley DB High Availability
Berkeley DB

- Berkeley DB Data Store
  - An embeddable, high-performance data store.
  - Supports
    - multiple concurrent threads of control
    - multiple concurrent processes of control
  - Does not support locking.
    - Concurrent writes need to be locked on an application side.
Berkeley DB

- Berkeley DB Concurrent Data Store
  - Provides built-in concurrency and locking feature.
    - Multiple-reads, single-writer at time.
    - Deadlock-free
  - Application is unaware of which is happening.
Berkeley DB

- Berkeley DB Transactional Data Store
  - Adds support for transactions and database recovery.
    - Commits and rollbacks.
    - High concurrency read/write operations.
  - By default, the serializable isolation level is used (degree 3 isolation).
    - READ-COMMITTED (degree 2 isolation) and READ-UNCOMMITTED (degree 1 isolation) are also possible.
  - Deadlocks can happen.
Berkeley DB

- Berkeley DB High Availability
  - Providing support for replication.
  - A single master system handles all updates.
  - A distributes these updates to multiple replicas.
  - All replicas can handle read operations.
  - Automatic failover:
    - If the master system fails for any reason, one of the replicas takes over as the new master system and distributes updates to the remaining replicas.
Berkeley DB

- While BerkeleyDB is a really advanced system, it is not:
  - a relational database,
  - network, client-server database
    - Berkeley DB Server is a Java application
      - ... and has a limited functionality.
Berkeley DB
Berkeley DB

- Access Methods
  - General purpose support for creating and accessing Berkeley DB database files
  - Useful in the absence of transactions
  - Usually for very simple applications only.
Berkeley DB

- Transactions
  - Allows a group of database changes to be treated as an atomic unit so that either all of the changes are done, or none of the changes are done.
  - This module is useful outside of the Berkeley DB package for processes that want to transaction-protect their own data modifications.
Berkeley DB

- Lock
  - General purpose lock manager.
  - As Transactions, is useful outside of the Berkeley DB package for processes that require a portable, fast, configurable lock manager.
Berkeley DB

- Buffer pool
  - Shared memory buffer pool implementation.
  - Cache allows multiple processes and threads within processes to share access to databases.
  - Buffer pool is page-oriented, and can be reused for different purposes.
Berkeley DB

- Logging
  - Write-ahead redo logs.
  - It’s rather unlikely this module can be reused for different applications.
Berkeley DB

- SQLite API
  - Drop-in replacement. *Almost*.
  - There are a few subtleties:
    - `BEGIN IMMEDIATE` or `BEGIN EXCLUSIVE` depending on what you are doing. If you do not, you can end up with deadlocks.
    - `SQLITE_BUSY` will never happen in BerkeleyDB
      - as there’s no such concept in BerkeleyDB.
Berkeley DB
Berkeley DB

rpm

exim

Gluster

Apache SpamAssassin

amazon

Oracle NoSQL Database

Zimbra

Percona Live
RocksDB

- Facebook’s fork of Google’s LevelDB.
- Key-value storage.
- Optimized to use many CPUs and make efficient use of fast storage, such as solid-state drives.
- Based on Log-Structured-Merge-Tree idea.
RocksDB

- The main idea is to minimize amount of random writes.
- **Random writes are slow. Sequential writes are fine.**
- RockDB introduces two major concepts:
  - Memtable
  - SStable
RocksDB

- Memtable:
  - A buffer that can temporarily host the incoming writes in memory.
    - memtable usually are sorted (in RocksDB, the default memtable implementation is SkipList),
      - but that is not required.
RocksDB

- SSTable:
  - when Memtable is full, the data is saved to the storage,
  - All data is saved in Sorted Strings Table.
  - SSTables are immutable.
RocksDB

- SSTable:
  - when MemTable is full, the data is saved to the storage.
    - RocksDB saves data also in redo-logs.
  - All data is saved in Sorted Strings Table.
  - SSTables are immutable.
RocksDB

- **SSTable:**
  - when MemTable is full, the data is saved to the storage.
    - RocksDB saves data also in redo-logs.
  - All data is saved in Sorted Strings Table.
  - SSTables are immutable.
  - But what if an application constantly updates the same key?
But what if an application constantly updates the same key?

- RocksDB will not check if the records exists, but will process like the key never existed before.
- When an application will try to get the data RocksDB will search all SSTables for the most recent version of data.
  - The above statement simplifies the real behaviour of SST, which is more subtle.
RocksDB comes with a background thread that compacts SSTable. Compaction is:
- merging two or more SSTables into one, larger SSTable,
- removing duplicates and deleted items,
- compaction is customizable, thus it can be extended for features like TTL.
Rate My Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations

Rate & Review

Tap a star to rate
Feedback (optional)

SUBMIT
Q&A