OctoDB - Simple Data Storage

Štěpán Davidovič

Czech Technical University in Prague, Faculty of Electrical Engineering

Introduction

Databases offer robust solution to storing large volume of diverse data and allow application developers to focus on the business logic, instead of designing and implementing physical storage.
OctoDB is a bachelor thesis with a goal to design and implement an embedded data storage allowing transactional processing, concurrent access and complex queries.

Data model and queries

OctoDB implements a simple data model, storing data in a set of flat tables. Each column has an associated data domain (data type), and order of rows in the tables is of no significance.
Each cell may contain even non-atomic values, therefore even more complex data such as arrays, structures, trees or even e.g. XML documents and other persistent objects may be stored. Current implementation offers base homogeneous array.
Data types have associated predicates. A query has this query predicated for all possibly matching rows. Query illustration:

column eq 100

This query will execute predicate *eq* with argument *100* for each cell in *column*, and return rows for which this predicate evaluated as true. Predicate calls can be combined and negated without limits, and are passed in conjunctive normal form. Data types which contain values of other data types (e.g. array) can call their predicates. This may result in the following example, where every item of an array must be equal to 100:

column every eq 100

Actual query is passed using API, no intermediate query language is used.

Transactions

A transaction is a group of commands, which can end either by committing all commands into the database, or rolling the commands back to previous state.
Transactions conform to four basic properties, known as ACID:

- *Atomicity* - a transaction either completes entirely, or not at all.
- *Consistency* - transaction begins at a consistent database state, and when finishes, it must return database to a consistent state.
- *Isolation* - one transaction does not affect another transaction.
- *Durability* - once a transaction commits data, the data must endure potential subsequent system failures.

Transactions implemented in OctoDB conform to ACID, offering "read committed" transaction isolation.

Concurrency control

Concurrency control is a mechanism used to ensure that data integrity is not compromised due to concurrent access. The work describes several significant concurrency control mechanisms:

- Global exclusive lock
- Two-phase locking
- Optimistic concurrency control
- Multi-version concurrency control (MVCC)

MVCC offers best performance, as conflicts are comparatively easy to resolve and there are no limits to the number of concurrent reads and writes. It is currently used in many database products, such as Oracle.
Because MVCC offers high performance during parallel access, it has been implemented as concurrency control mechanism in OctoDB.

Performance

In order to asses the efficiency of the implementation, several benchmarks were implemented. OctoDB was then compared to two contemporary database products: SQLite, which is also an embedded database system, and PostgreSQL, which represents a full-featured client/server database.

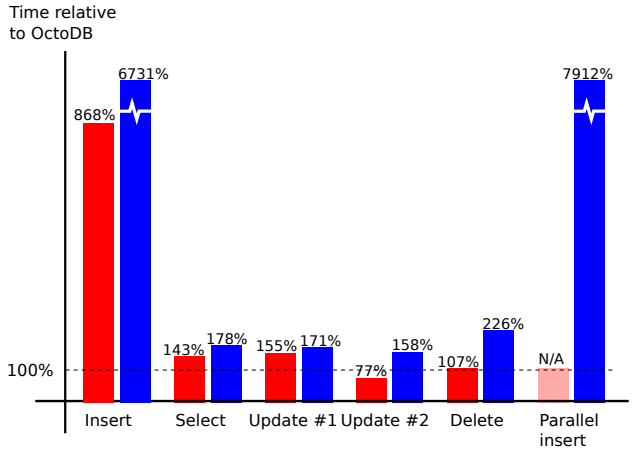

Figure: Benchmark results, relative to OctoDB - lower is better. Red column represents SQLite, blue represents PostgreSQL, OctoDB results are 100%.

The benchmarks perform most common operations on a data set of ca. 100 000 rows.

Conclusion

OctoDB is a successful implementation of a database, storing both simple and complex values and querying them. The database allows concurrent access to the data by both readers and writers and supports transactions conforming to ACID.