CUDB: A simple document-based NoSQL DBMS

Jonathan Lam, Derek Lee, Victor Zhang

The Cooper Union for the Advancement of Science and Art

2021/12/14

What is CUDB?

Cooper **U**nion **D**ata**B**ase a.k.a. CUDA++

Document model Natural, schema-less representation

File-backed persistence Mmapv1-inspired buffer/storage engine

B-tree indexing User-declared indices speed up range queries

MongoDB-like CRUD API Familiar NoSQL CRUD operations

Value Types

```rust
pub enum Value {
    /// Special type for unique identification,
    /// similar to MongoDB's `_id`.
    Id(String),
    /// Fixed-size integer type.
    Int32(i32),
    /// Arbitrary-length strings.
    String(String),
    /// Recursive documents (hashtables).
    Dict(Document),
    /// Array types.
    Array(Vec<Value>),
}
```

Figure: `cudb::value::Value` definition

Collection management API

```
impl Collection {
    /// Create a collection from a path.
    pub fn from(path: &str) -> Collection {}

    /// Close collection and underlying file pointer.
    pub fn close(self) {}

    /// Drop collection data and indices.
    pub fn drop(self) {}
}
```

Figure: `cudb::db::Collection` management API

Collection CRUD API

```
impl Collection {
    insert_one(doc: Document) {}
    insert_many(docs: Vec<Document>) {}
    find_one(query: Query) -> Option<Document> {}
    find_many(query: Query) -> Vec<Document> {}
    find_all() -> Vec<Document> {}
    update_one(query: Query, update: Document) {}
    update_many(query: ConstraintDocument,
                update: Document) {}
    delete_one(query: Query) {}
    delete_many(query: ConstraintDocument) {}
}
```

Figure: CRUD API

Sample query syntax

```
SELECT name, dob
FROM students
WHERE gpa>3.0 AND grade<>9
ORDER BY gpa DESC;
```

Figure: SQL

Sample query syntax

```
db.students.find({
  gpa: { $gt: 3.0 },
  grade: { $ne: 9 }
}, {
    name: 1,
    dob: 1
}).sort({ gpa: -1 });
```

Figure: MQL (JS)

Sample query syntax

```
Query(
  constraints: {
    ["gpa"]: Constraint::GreaterThan(Value::Double(3.0))),
    ["grade"]: Constraint::NotEquals(Value::Int32(9)))
  },
  projection: {
    ["name"]: Projection::Include,
    ["dob"]: Projection::Include
  },
  order: Some([
    ResultOrder::Desc(FieldPath)
  ])
)
```

Figure: CUDB (RON)

Index schema and index instance

```
// Index schema
index_schema = [
  ( field_path: ["a"], default: Value::Int(0) ),
  ( field_path: ["b", "c"], default: Value::String("World") ),
  ( field_path: ["b", "e"], default: Value::String("some value") ),
]

// Document
document = {
  "a": Value::Int32(42),
  "b": Value::Dict({
    "c": Value::String("Hi"),
    "d": Value::Int32(-2)
  })
}

// Index instance for this schema and document
index_instance = [
  Value::Int32(42),
  Value::String("Hi"),
  Value::String("some value")
]
```

Figure: Index schema and index instance

Performance and memory bounds

Operation	Time	Memory
Insert	$O(I \log C)$	$O(D)$
RUD (no index, unsorted)	$O(I + C)$	$O(CD)$
RUD (index, unsorted)	$\approx O(I + \log C)$	$\approx O(\log CD)$
RUD (no index, sorted)	$O(I + C \log C)$	$O(CD)$
RUD (index, sorted)	$\approx O(I + (\log C)(\log \log C))$	$\approx O(\log CD)$

Table: Expected performance characteristics

Resources

- GitHub
- API
- Report
- Presentation