
TARANTOOL

In memory database

Includes application server

Key-value/document store with secondary indexes

Also can work as a relational database

ACID compliant with transaction journal (WAL) and snapshots for durability

Data stored in spaces (tables)

Spaces made up of tuples (rows) whose data values are called fields

Fields don’t have names and can be made of composite structures (hashmaps, etc.)

The tuples in a space can have a defined format

Standard language is Lua

Bindings for other languages exists (Python, PHP, Javascript, ...)

Basic CRUD operations which are combined in Lua scripts

We can use Tarantool as a simple key-value store similiar to Redis

Tuples are structured and can contain composite structures (including other tuples) → We can use Tarantool as a document store

Also supports SQL queries and complies with most of the standard

- Configure database

```
box.cfg{}
```

- Create space (table):

```
s = box.schema.space.create('heights')
```

- Format space:

```
s:format({
        > {name = 'id', type = 'unsigned'},
        > {name = 'firstname', type = 'string'},
        > {name = 'height', type = 'unsigned'}
})
```

- Create index called `primary`

```
s:create_index('primary', {
        > type = 'tree',
        > parts = {'id'}
})
```

- Add secondary index

```
s:create_index('secondary', {
        > type = 'tree',
        > parts = {'firstname'},
        > unique = false
})
```

- Drop index

```
s.index.secondary:drop()
```

- Drop space

```
s:drop()
```

- Select tuples using the secondary index

```
s.index.secondary:select{'Michal'}
```

- More complex query

```
s:select({0},{iterator='GT',offset=1,limit=2})
```

- Inserting tuples (rows):

```
s:insert{1, 'Michal', 183}
s:insert{2, 'Pavel', 197}
s:insert{3, 'Veronika', 168}
s:insert{4, 'Michal', 176}
```

- Deleting tuples

```
s:delete{3}
```

- Update

```
s:update({2}, {{'=', 'height', 198}})
```

- Create table (space)

```
CREATE TABLE modules (name STRING, size INTEGER, purpose STRING, PRIMARY KEY (name));
```

- Create Index

```
CREATE INDEX size ON modules (size);
CREATE UNIQUE INDEX purpose ON modules (purpose);
```

- CRUD

```
INSERT INTO modules VALUES ('json', 14, 'format functions for JSON');
UPDATE modules SET size = 15 WHERE name = 'json';
DELETE FROM modules WHERE name = 'json';
```

- Foreign keys

```
CREATE TABLE modules (name STRING,
                      size INTEGER,
                       purpose STRING,
                       PRIMARY KEY (name),
                       CHECK (size > 0));

CREATE TABLE submodules (name STRING,
                         module_name STRING,
                         size INTEGER,
                         purpose STRING,
                         PRIMARY KEY (name),
                         FOREIGN KEY (module_name) REFERENCES
                         modules (name));
```

- Subqueries

```
SELECT name FROM submodules
WHERE module_name =
    (SELECT name FROM modules WHERE purpose LIKE '%Database%');
```

- Join

```
SELECT * FROM modules JOIN submodules;
```

- Order by

```
SELECT * FROM modules ORDER BY name DESC LIMIT 2 OFFSET 2;
```

- Group by

```
SELECT module_name, count(*) FROM submodules GROUP BY module_name HAVING count(*) > 0;
```

Not fully SQL compatible:

- Does not support authorization – NoSQL requests instead
- Possible to bypass defined integrity (e.g. violate a foreign-key constraint through NoSQL request)
- Tarantool’s views are not updatable

Pros:

- Fast read and write operations
- Lua application server included

Cons:

- Need large amount of RAM
- Not as popular