

CRUD operation

```
INSERT INTO User  
VALUES(' Jim Doe', ' doe@party. com')
```

```
# delete user entry  
dn: uid=jim dc=betraye, dc=com  
changetype: delete
```

```
db. user  
  . update( { _id: 1 },  
            { $rename: { ' Jimmy': ' Jim' } } )  
  . comment(" Renaming » Jimmy« to » Jim");
```

Query

```
SELECT *  
FROM User  
WHERE email LIKE 'd%
```

```
(&(objectClass=user)(email=d*))
```

```
db.inventory.find({  
  $and: [  
    { class: "User" },  
    { email: /^d/ }  
  ]  
})
```

```
List<User> c = qf  
  .selectFrom(user)  
  .where(user.email  
    .StartsWith("d"))  
  .fetch();
```

Schema

```
CREATE TABLE User (  
  id INTEGER PRIMARY KEY  
  , cname VARCHAR(255)  
  , email VARCHAR(255)  
)
```

```
{  
  "$schema": "http://json-schema.org/draft/2019-09/schema",  
  "title": "Product",  
  "type": "object",  
  "required": ["id", "name", "price"],  
  "properties": {  
    "id": {  
      "type": "number"  
    }  
  }  
  ...  
}
```

Procedures / triggers

```
CREATE PROCEDURE
  insert_data(a integer, b integer)
LANGUAGE SQL
AS $BODY$
  INSERT INTO tbl VALUES (a);
  INSERT INTO tbl VALUES (b);
$BODY$;

CALL insert_data(1, 2);
```

```
CREATE TRIGGER last_change
  BEFORE UPDATE
  ON tbl
  FOR EACH ROW
  EXECUTE PROCEDURE log_changes();
```


Data access control

```
GRANT SELECT, INSERT,  
      UPDATE, DELETE  
ON employees  
TO smithj;
```

```
privileges: [  
  { resource:  
    {  
      db: "products",  
      collection: "user"  
    },  
    actions: [ "find",  
              "update",  
              "insert" ]  
  },  
]
```

API support

```
conn = DriverManager.getConnection(
    (DB_URL, USER, PASS));

stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(
    "SELECT DISTINCT email FROM User");
while(rs.next()) {
    System.out.println(rs.getString("email"));
}
```

```
client = MongoClient(port=27017)
db=client.business
fivestar = db.reviews.find_one({'rating': 5})
print(fivestar)
```