CQL-JAVA SURVEY

Create keyspace

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/create_keyspace_r.html

```
CREATE ( KEYSPACE | SCHEMA )  keyspace_name WITH REPLICATION = map AND DURAB
LE_WRITES = ( true | false )
```

```
private void createKeyspace(String key) { // create database

                session.execute("CREATE KEYSPACE IF NOT EXISTS ____ WITH replication " +
                                "= {'class':'SimpleStrategy', 'replication_factor':3};");

}
```

// put keyspace's name in blank.

Use keyspace

```
private void useKeySpace(String key) {
                session = cluster.connect(key);
}
```

Create table in keyspace

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/create_table_r.html

CREATE TABLE *keyspace_name.*table_name(column_definition*,column_definition,…) WITH pro perty AND property ...*

API Example:

```
private void createTable() {
                String sql = String
                                .format("CREATE TABLE %s.teacher (id bigint PRIMARY KEY,name
text, title text,courses list<bigint>)",
                                                keyspaceName);
                System.out.println(sql);
                CassandraConnection conn = pool.getConnection();
                try {
                                conn.execute(sql);
                } finally {
                                conn.close();
                }
        }
```

Alter table

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/alter_table_r.html

```
ALTER TABLE keyspace_name.table_name instruction
```

```
public void alterTableDemo() {
		System.out.println("---Altering table----");
		session.execute("ALTER TABLE demo.songs DROP data");
}
```

Drop table

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/drop_table_r.html

```
DROP TABLE keyspace_name.table_name
```

```
private void dropTable() {
		String sql = String.format("DROP TABLE %s.teacher", keyspaceName);
		System.out.println(sql);
		CassandraConnection conn = pool.getConnection();
		try {
			conn.execute(sql);
		} finally {
			conn.close();
		}
	}
```

Close

```
public void close() {
		session.close();
		cluster.close();
}
```

Insert item

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/insert_r.html

```
INSERT INTO keyspace_name.table_name ( column_name, column_name...) VALUES (
 value, value ... ) USING option AND option
```

```
private void insert(Teacher obj) {
		String sql = String
				.format("INSERT INTO %s.teacher(id,name,title,courses) VALUES
(?,?,?,?)",
						keyspaceName);
		System.out.println(sql);
		CassandraConnection conn = pool.getConnection();
```

```
		try {
			PreparedStatement ps = conn.prepare(sql);
			System.out.println(ps);
			BoundStatement bs = ps.bind(obj.getId(), obj.getName(), obj.getTitle(),
obj.getCourses());
			conn.execute(bs);
		} finally {
			conn.close();
		}
	}
```

Select data contains xx

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/select_r.html

```
SELECT select_expression FROM keyspace_name.table_name WHERE relation AND re
lation ... ORDER BY ( clustering_column ( ASC | DESC )...) LIMIT n ALLOW FIL
TERING
```

```
private List<Teacher> selectAll() {
		String sql = String.format("SELECT * FROM %s.teacher", keyspaceName);
		System.out.println(sql);
		CassandraConnection conn = pool.getConnection();
		List<Teacher> result = new ArrayList<Teacher>();
		try {
			ResultSet rs = conn.execute(sql);
			for (Row row : rs) {
				Teacher obj = new Teacher();
				obj.setId(row.getLong("id"));
				obj.setName(row.getString("name"));
				obj.setTitle(row.getString("title"));
				obj.setCourses(row.getList("courses", Long.class));
				result.add(obj);
			}
		} finally {
			conn.close();
		}
		return result;
	}
private List<Teacher> selectById(Long id) {
		if (null == id) {
			return null;
		}
		String sql = String.format("SELECT * FROM %s.teacher WHERE id = ?",
keyspaceName);
		System.out.println(sql);
```

```
        CassandraConnection conn = pool.getConnection();
        List<Teacher> result = new ArrayList<Teacher>();

        try {
            PreparedStatement ps = conn.prepare(sql);
            BoundStatement bs = ps.bind(id);
            ResultSet rs = conn.execute(bs);

            for (Row row : rs) {
                Teacher obj = new Teacher();
                setInfor(obj, row);
                result.add(obj);
            }
        } finally {
            conn.close();
        }
        return result;
    }
```

Select data where xx

```
ResultSet result = session.execute(QueryBuilder.select("a","b")
                .from("mykeyspace", "tablename")
                .where(QueryBuilder.gt("a", 1))
                .and(QueryBuilder.lt("a", 1)));
Iterator<Row> iterator = result.iterator();
while(iterator.hasNext())
{
        Row row = iterator.next();
        row.getInt("a");
        row.getInt("b");
}
```

or

```
BoundStatement bindStatement =
session.prepare(
"select * from mykeyspace.tablename where a=? and b=?")
.bind("1","2");
session.execute(bindStatement);
```

or

```
PreparedStatement prepareStatement =
```

```
session.prepare(
"select * from mykeyspace.tablename where a=? and b=?");
BoundStatement bindStatement =
      new BoundStatement(prepareStatement).bind("1","2");
session.execute(bindStatement);
```

Delete item

http://docs.datastax.com/en/cql/3.0/cql/cql_reference/delete_r.html

```
DELETE column_name, ... | ( column_name term )  FROM keyspace_name.table_nam
e USING TIMESTAMP integer WHERE row_specification

//TODO
```