Let's talk about NoSQL Standard

Otávio Santana
@otaviojava
NoSQL

- Database
- Doesn't use relationship
- BASE
- Five types

[Logos of CouchDB, Redis, Cassandra, MongoDB, membase, and Riak]
Key Value

- AmazonDynamo
- AmazonS3
- Redis
- Scalaris
- Voldemort
- Couchbase
### Key Value

- American Express
- AT&T
- Bank of America

<table>
<thead>
<tr>
<th>Table</th>
<th>Bucket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>Key/value pair</td>
</tr>
<tr>
<td>Column</td>
<td>---</td>
</tr>
<tr>
<td>Relationship</td>
<td>---</td>
</tr>
</tbody>
</table>
Document

- AmazonSimpleD
- ApacheCouchdb
- MongoDB
- Riak
- Couchbase

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ]
}
```
Document

- MetLife
- BuzzFeed
- Ebay

<table>
<thead>
<tr>
<th>Table</th>
<th>Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
<tr>
<td>Column</td>
<td>Key/value pair</td>
</tr>
<tr>
<td>Relationship</td>
<td>Link</td>
</tr>
</tbody>
</table>
- Hbase
- Cassandra
- Scylla
- Clouddata
- SimpleDb
- DynamoDB

```plaintext
Column

<table>
<thead>
<tr>
<th>Row-key</th>
<th>Columns...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo</td>
<td>Duty</td>
</tr>
<tr>
<td></td>
<td>Sun</td>
</tr>
<tr>
<td>Aphrodi te</td>
<td>Duty</td>
</tr>
<tr>
<td></td>
<td>{Love, happy}</td>
</tr>
<tr>
<td>Ares</td>
<td>Weapon</td>
</tr>
<tr>
<td></td>
<td>Sword</td>
</tr>
<tr>
<td>Kratos</td>
<td>Dead Gods</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
</tbody>
</table>
```


- Github
- Netflix
- Reddit

<table>
<thead>
<tr>
<th>Table</th>
<th>Column Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>Column</td>
</tr>
<tr>
<td>Column</td>
<td>Key/value pair</td>
</tr>
<tr>
<td>Relationship</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
Graph

- Neo4j
- InfoGrid
- Sones
- HyperGraphDB

Apollo → Is brother → Ares
Ares → killed → Kratos
Kratos → killed → Ares
Ares → was dead → Kratos
Kratos → was dead → Apollo
Graph

- Walmart
- InfoJobs
- Linkedin

<table>
<thead>
<tr>
<th>Table</th>
<th>Vertex and Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row</td>
<td>Vertex</td>
</tr>
<tr>
<td>Column</td>
<td>Vertex and Edge property</td>
</tr>
<tr>
<td>Relationship</td>
<td>Edge</td>
</tr>
</tbody>
</table>
Multi-model

- OrientDB
- Couchbase
- Elasticsearch
- Cassandra
The Problem
The Problem
The Current solution

- Spring Data
- Hibernate OGM
- TopLink
JPA problem

- Saves Async
- Async Callback
- TTL
- Consistency Level
- SQL based
- Diversity in NoSQL
The Challenge

- 4 Types
- Differents behaviors
- Differents goals
The Solution

- SPI
- Communication
- No lock-in
- Split problems
JNoSQL

- Several tools to make easy an integration between the Java Application with the NoSQL.
- Communication API
- Abstraction API
Diana

- API Communication layer
- Document, key-value, Column, Graph
- Universal Adapter
- Standard
What Diana is not

- A new API to replace JPA
- A new API to abstraction layer
- Just one API communication to solve all kind of NoSQL database
- Be responsible to do integrations with other technologies such as CDI, EJB, Bean Validation, Spring, etc.
Diana Project

- Commons API
- Document API
- Graph API
- Key-value API
- Column API
- Four TCKs
Nomenclature

- Configuration
- Factory
- Manager
- Entity
- Value
Diversity

ColumnEntity entity = ColumnEntity.of("COLUMN_FAMILY");
Column id = Column.of("id", 10L);
entity.add(id);
entity.add(Column.of("version", 0.001));
entity.add(Column.of("name", "Diana"));
entity.add(Column.of("options", Arrays.asList(1, 2, 3)));

//both implementation
columnEntityManager.save(entity);
ColumnQuery query = ColumnQuery.of("COLUMN_FAMILY");
query.addCondition(ColumnCondition.eq(id));
Optional<ColumnEntity> result = columnEntityManager.singleResult(query);

//cassandra implementation
List<ColumnEntity> entities = columnEntityManagerCassandra.cql("select * from newKeySpace.newColumnFamily where id=10;");
columnEntityManagerManagerCassandra.save(entity, ConsistencyLevel.ALL);
NoSQL Providers

- ArangoDB
- cassandra
- Couchbase
- mongoDB
- riak
- Scylla
- Hazelcast
- Apache HBase
- neo4j
- OrientDB
- Redis
JUGs/Communities
Graph API

- Apache TinkerPop
- Apache Project
- 12 Graph databases
- Most stable code
- Adapter layer
- CDI Based
- Diana Based
- Annotation Based
- Events to insert, delete, update
- Supports to Bean Validation
- Configurable and extensible

Artemis
@Entity("movie")
public class Movie {
  @Column
  private String name;

  @Column
  private long year;

  @Column
  private Set<String> actors;
Events

Movie

firePreEntity
firePreAPI
firePostAPI
firePostEntity

Interceptor
Configurable and extensible

- EventManager
- CrudOperation
- ClassRepresentations
- Converter
- Decorate
- Replace
Why Diana?

- Goddess of the hunt, nature and moon
- Fought in Troy
- brave warrior and hunter
Road Map

- Draft and code Proposal
- Community Feedback
- Involve NoSQL Vendors
- Involve Solution Vendors
- Eclipse Project
- Development
- JSR
Site

Apache Diana is a flexible and extensible API to connect NoSQL databases. It brings an easy interface to support key-value, column family, document oriented and graph databases as JDBC does for SQL databases.

http://jnosql.org/
Diana
A flexible and extensible API to connect NoSQL databases
https://github.com/JNOSQL

diana
Diana is a flexible and extensible API to connect NoSQL databases. It brings an easy interface to support key-value, column family, document oriented and graph databases as JDBC is for SQL databases.
Updated 2 hours ago

diana-site
Diana is a flexible and extensible API to connect NoSQL databases. It brings an easy interface to support key-value, column family, document oriented and graph databases as JDBC is for SQL databases.
Updated 5 days ago

diana-book
Diana is a flexible and extensible API to connect NoSQL databases. It brings an easy interface to support key-value, column family, document oriented and graph databases as JDBC is for SQL databases.
Updated 5 days ago

diana-demos
Diana examples code
Updated 7 days ago

https://github.com/JNOSQL
Mailing list: **jnosql-dev**

**jnosql developer discussions**

**About jnosql-dev**

**jnosql developer discussions**

**Using jnosql-dev**

To post a message to all the list members, send email to [jnosql-dev@eclipse.org](mailto:jnosql-dev@eclipse.org). You must be subscribed to the list before you can post. To access a web archive of this list, visit the **jnosql-dev Archives** or subscribe to this list’s **RSS feed**.

**Subscribing to jnosql-dev**

All contributions you make to our web site are governed by our **Terms Of Use**. Your interactions with the Eclipse Foundation web properties and any information you may provide us about yourself are governed by our **Privacy Policy**.

Subscribe to jnosql-dev by filling out the following form. You will be sent email requesting confirmation, to prevent others from gratuitously subscribing you. This is a hidden list, which means that the list of members is available only to the list administrator.

https://dev.eclipse.org/mailman/listinfo/jnosql-dev
Mailing List

https://gitter.im/JNOSQL/developers
Thank you

Otávio Santana
@otaviojava
osantana@tomitribe.com
otaviojava@apache.org