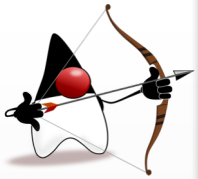
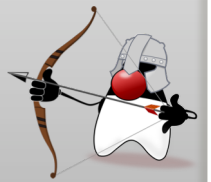


JNoSQL



Eclipse JNoSQL

One API to Many NoSQL Databases

Jonathan Vila Lopez

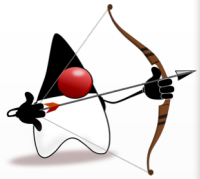
Leonardo Lima

Otávio Santana

Hillmer Chona

Patricia Uribe

#CodeOne #JNoSQL @JNoSQL @HillmerCh @leomrlima @otaviojava @patricia_uz @vilojona



JNoSQL

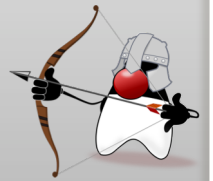
About the Speakers

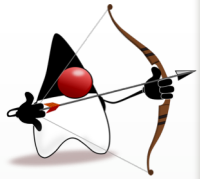
Hillmer Chona

Medellin JUG Leader

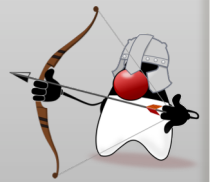
Catholic University Luis Amigó - Colombia

Oracle Groundbreaker ambassador





JNoSQL



About the Speakers

Jonathan Vila Lopez

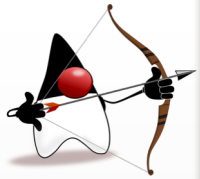
Software developer for the last 25 years in different languages

Works for RedHat

Leader of Barcelona Java Users Group and JBCNConf

Very passionate about Java

Interested in particular on OSGi, Kubernetes, VR



JNoSQL

About the Speakers

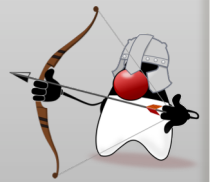
Leonardo Lima

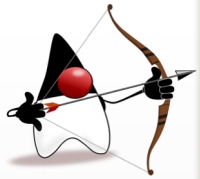
Computer engineer, server & embedded sw developer

CTO at V2COM

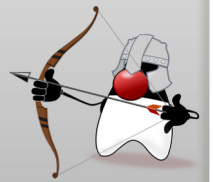
Spec Lead – JSR363 – Units of Measurement

V2COM's Representative at JCP Executive Committee





JNoSQL



About the Speakers

Otávio Santana

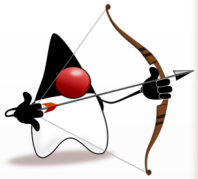
Software engineer, Tomitribe

Java Champion, SouJava JUG Leader

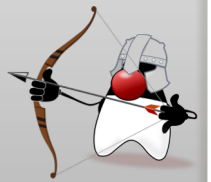
Apache, Eclipse and OpenJDK Committer

Expert Group member in many JSRs

Representative at JCP EC for SouJava



JNoSQL

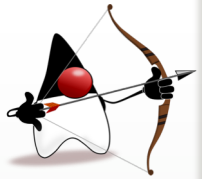


About the Speakers

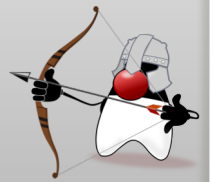
Patricia Uribe

Medellin JUG Leader

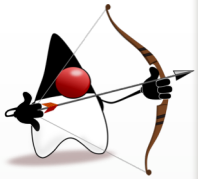
Catholic University Luis Amigó - Colombia



JNoSQL



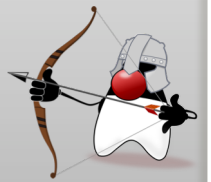
NoSQL
(Not Only SQL)

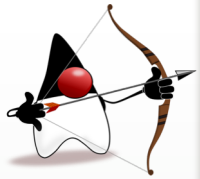


JNoSQL

What defines NoSQL databases?

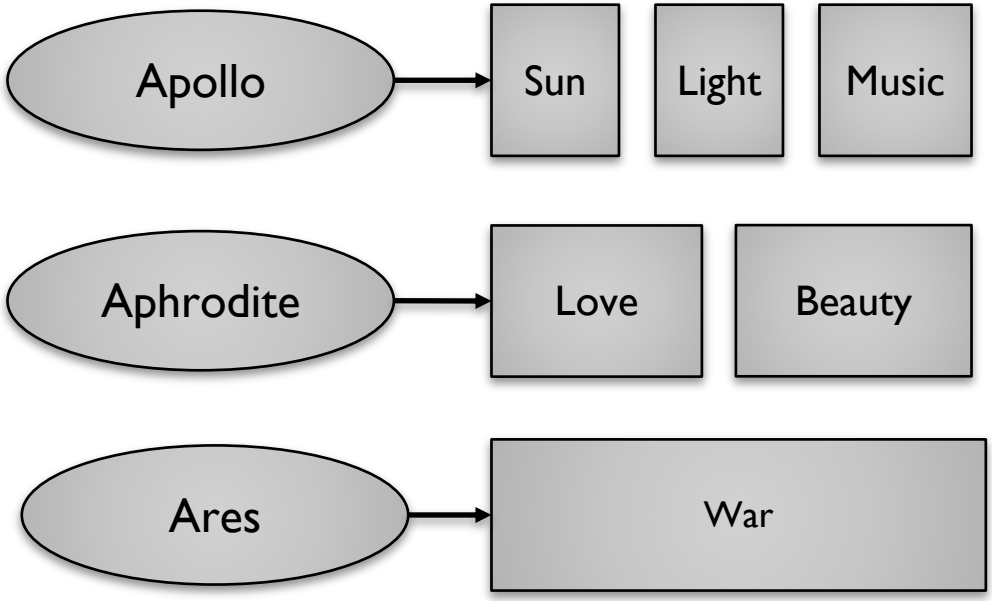
- No fixed data structure
- Not an ACID, but a BASE
- Five different types
  - Key/Value
  - Column Family
  - Document
  - Graph
  - Multi-Model



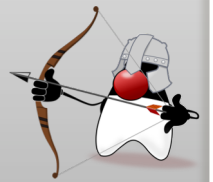


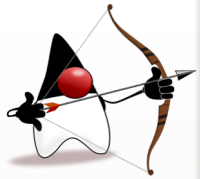
Key/Value

- AmazonDynamo
- AmazonS3
- **Redis**
- Scalaris
- Voldemort
- Couchbase
- Hazelcast



JNoSQL

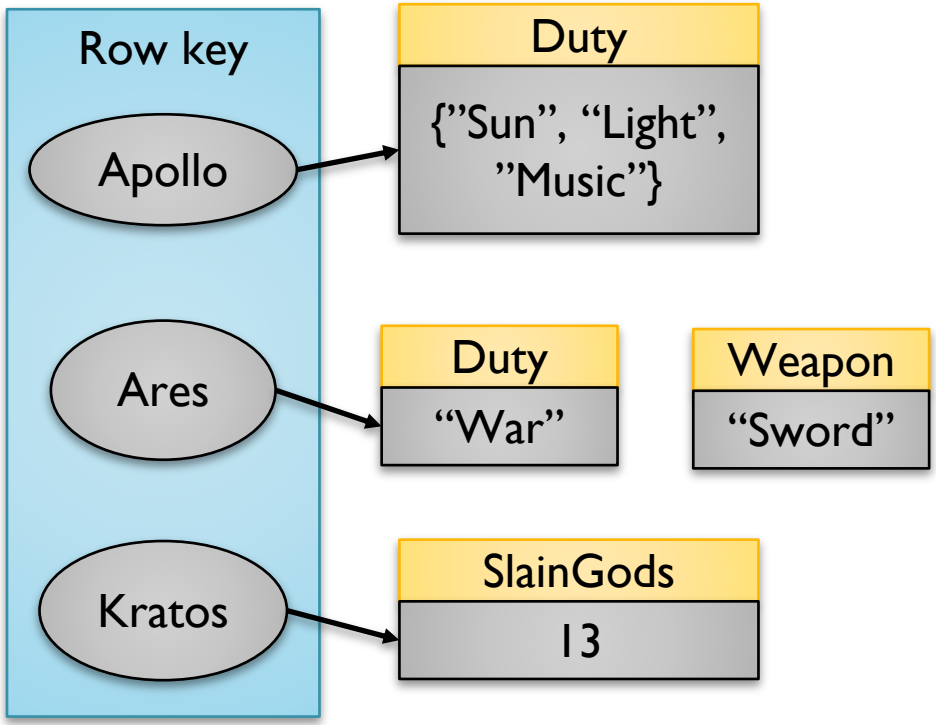


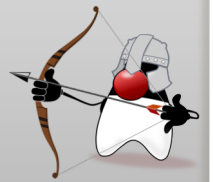


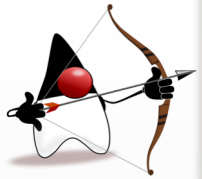
JNoSQL

Column Family

- Hbase
- **Cassandra**
- Scylla
- Clouddata
- SimpleDb
- DynamoDB





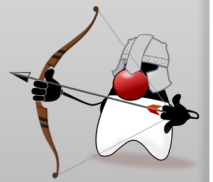


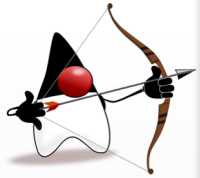
JNoSQL

Document

- AmazonSimpleD
- Apache CouchDB
- Couchbase
- **MongoDB**
- Riak

```
{
  "name":"Diana",
  "father":"Jupiter",
  "mother":"Latona",
  "siblings":{
            "name": "Apollo"
  },
  "godOf": {"Hunt"}
}
```

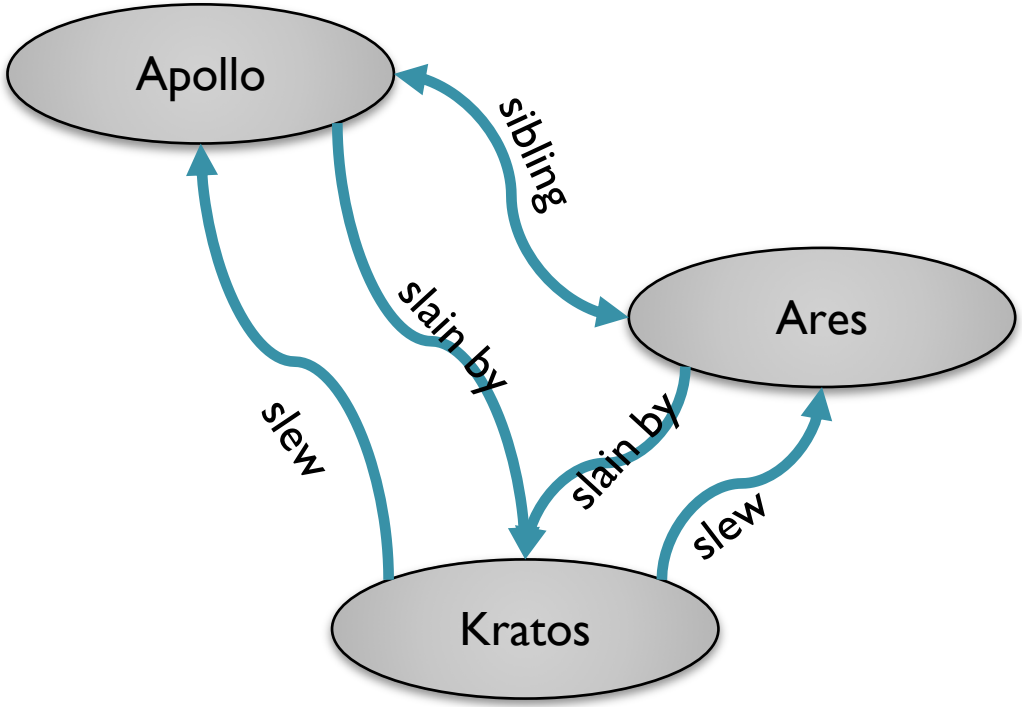


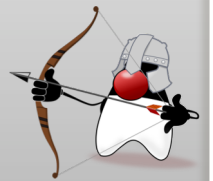


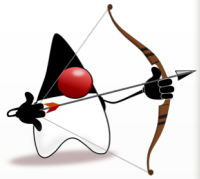
JNoSQL

Graph

- **Neo4J**
- InfoGrid
- Sones
- HyperGraphDB



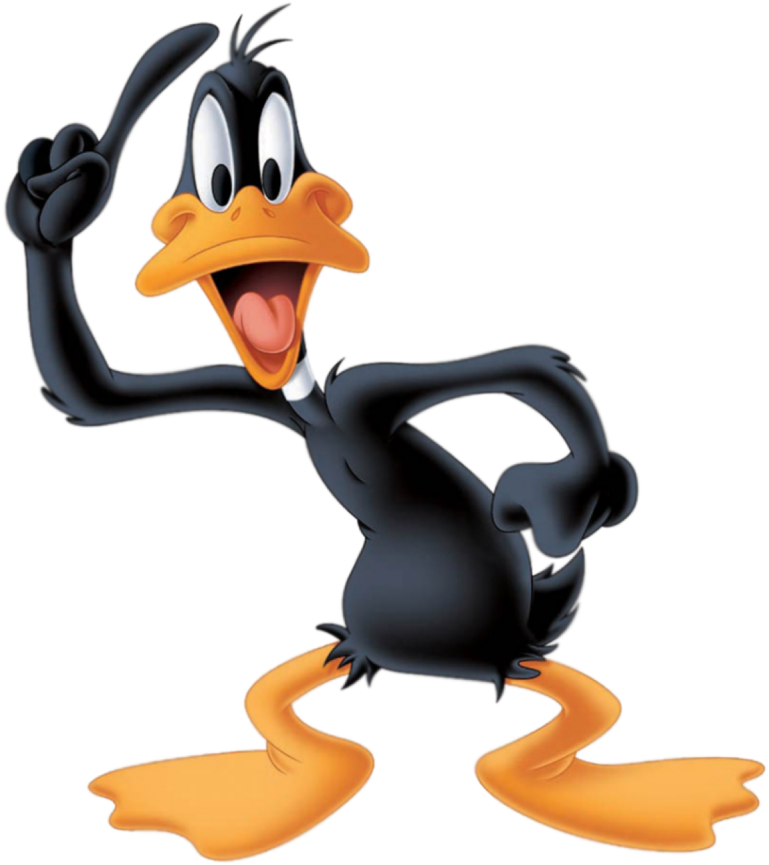


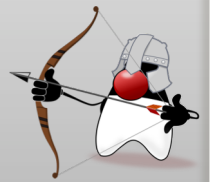


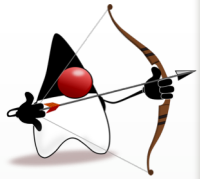
JNoSQL

Multi-model

- OrientDB
  - graph, document
- Couchbase
  - key-value, document
- Elasticsearch
  - document, graph
- ArangoDB
  - column family, graph, key-value



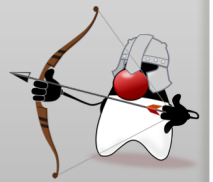


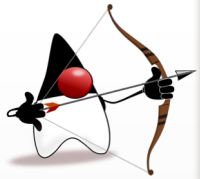


JNoSQL

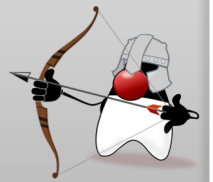
SQL -> NoSQL

SQL	Key-Value	Column Family	Document	Graph
Table	Bucket	Column Family	Collection	
Row	Key/Value pair	Column	Document	Vertex
Column		Key/Value Pair	Key/Value Pair	Vertex/Edge properties
Relationship			Link	Edge

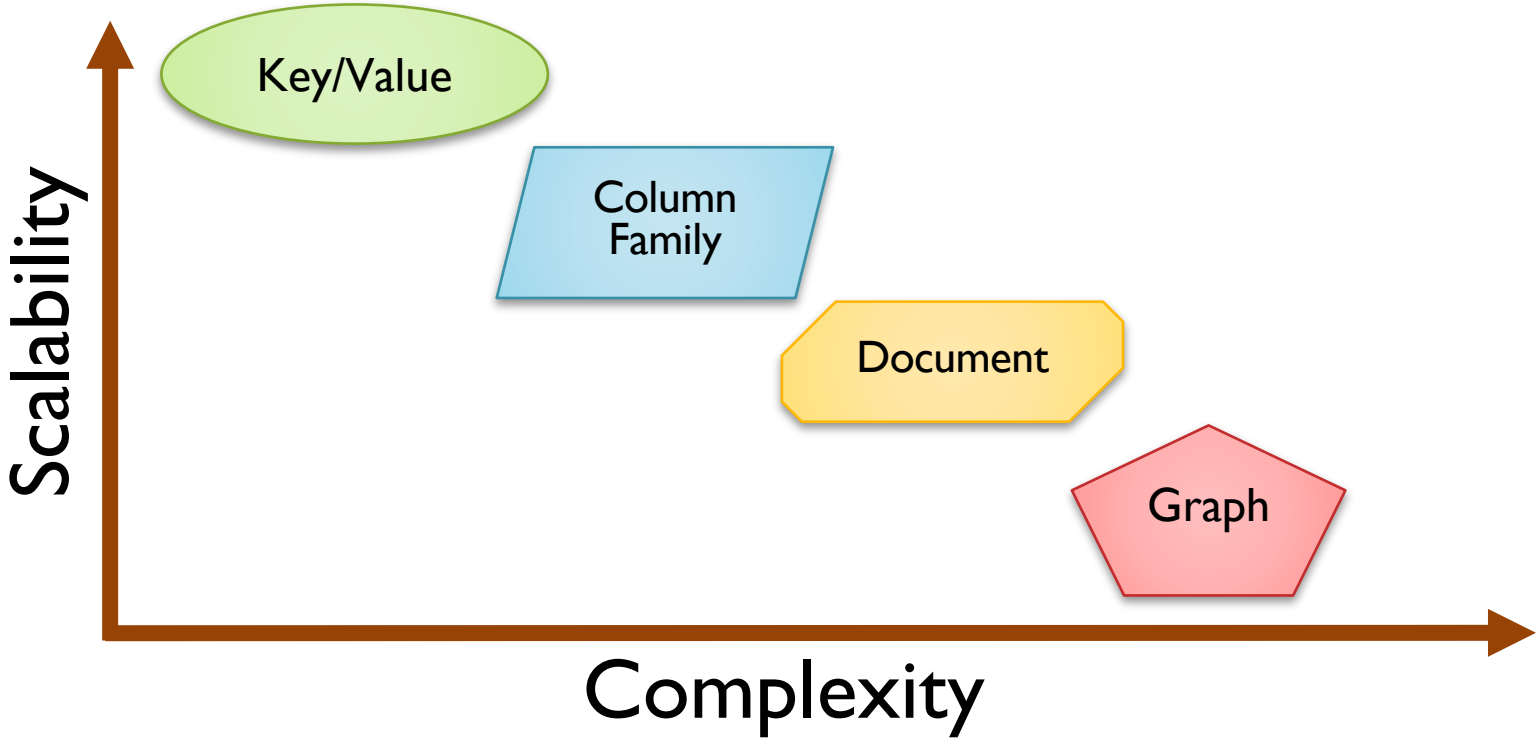


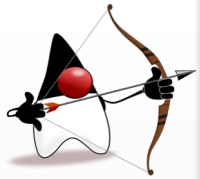


JNoSQL

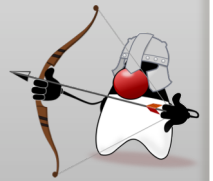


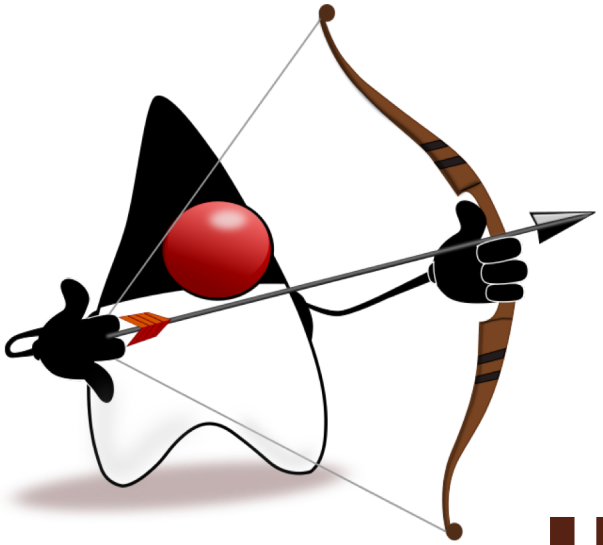
Scalability x Complexity



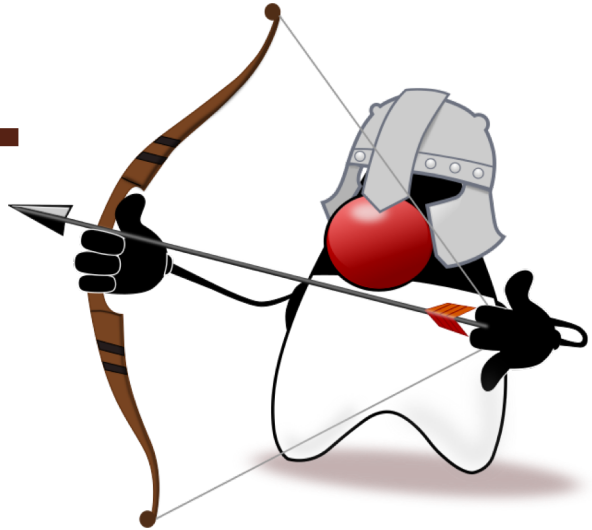


JNoSQL

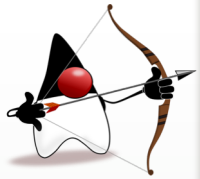




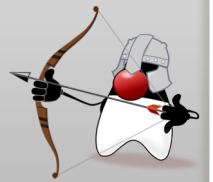
JNoSQL



#CodeOne #JNoSQL @JNoSQL @HillmerCh @leomrlima @otaviojava @patricia_uz @vilojona

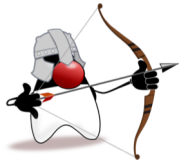


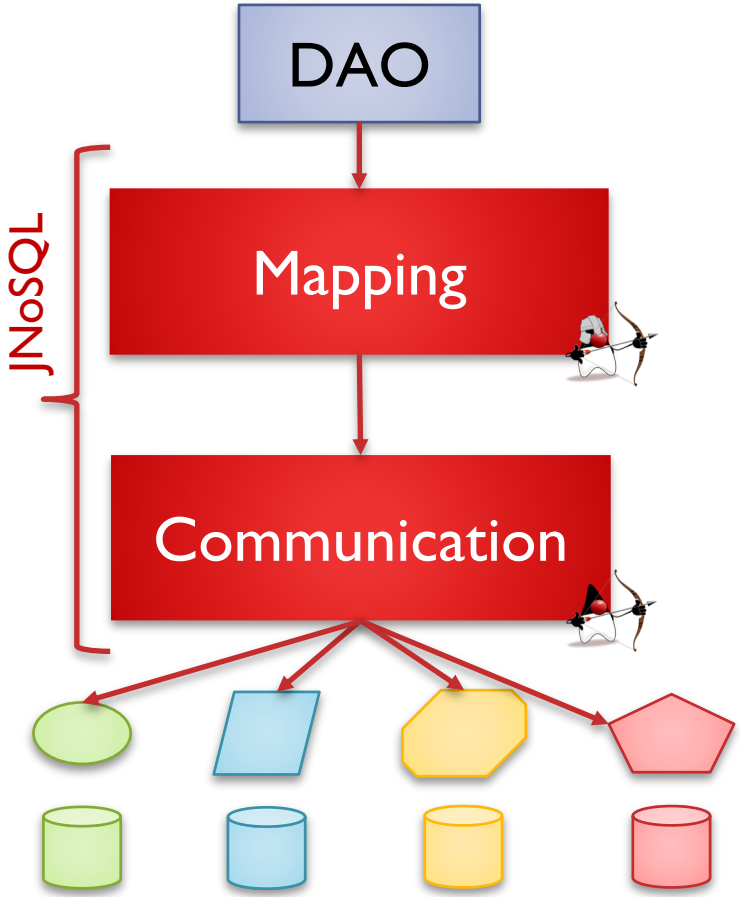
JNoSQL

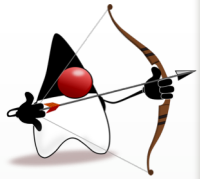


What is JNoSQL?

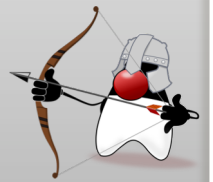
- Mapping API - **Artemis**
- Communication API - **Diana**
- No lock-in
- Divide and conquer







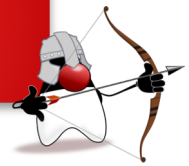
JNoSQL



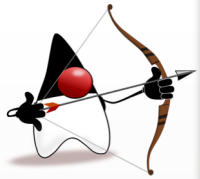
Eclipse JNoSQL

- Eclipse Foundation
- Apache v2 + EPL 1.0
- API to each NoSQL type
- Configurable
- Extensible

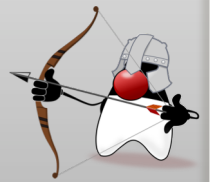
Mapping



Communication



JNoSQL



For example: for a Document DB…

ArangoDB

```
BaseDocument baseDocument = new BaseDocument();
baseDocument.addAttribute(name, value);
```

mongoDB

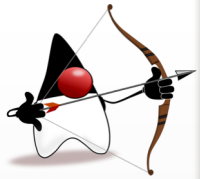
```
Document document = new Document();
document.append(name, value);
```

Couchbase

```
JsonObject jsonObject = JsonObject.create();
jsonObject.put(name, value);
```

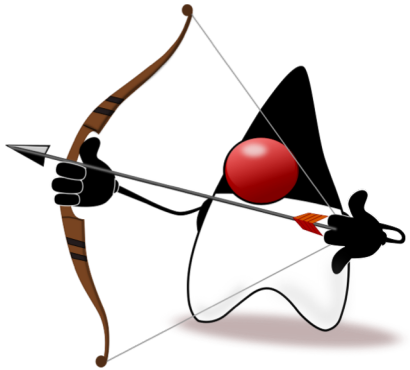
OrientDB

```
ODocument document = new ODocument("collection");
document.field(name, value);
```



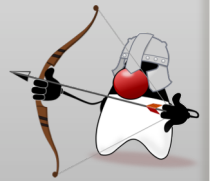
JNoSQL

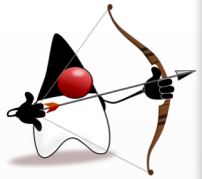
… now with JNoSQL



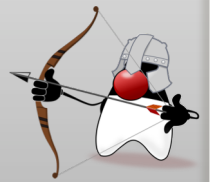
```
DocumentEntity entity = DocumentEntity.of("documentCollection");
Document document = Document.of(name, value);
entity.add(document);
```

OrientDB Couchbase mongoDB ArangoDB



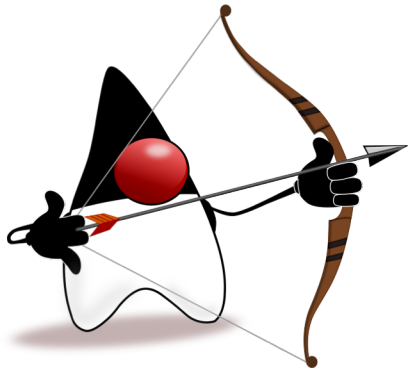


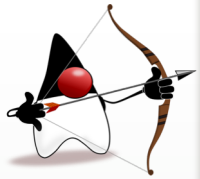
JNoSQL



Names & Definitions

- Configuration
- Factory
- Manager
- Entity

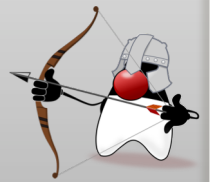


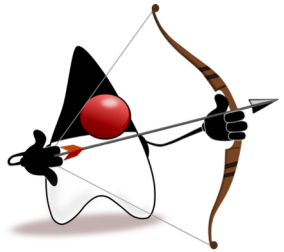


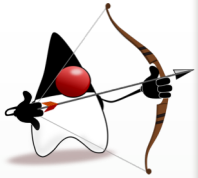
JNoSQL

Names & Definitions

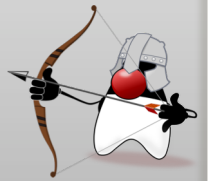
```
Column<b>Configuration</b><?> condition = new DriverConfiguration();
try(Column<b>Family</b><b>ManagerFactory</b> managerFactory = condition.get()) {
    Column<b>Family</b><b>Manager</b> entityManager = managerFactory.get(KEY_SPACE);
    entityManager.<b>insert</b>(entity);

    ColumnQuery select = select().from(COLUMN_FAMILY).where(eq(id)).build();
    ColumnDeleteQuery delete = delete().from(COLUMN_FAMILY)
                                       .where(<b>eq</b>(id)).build();

   Optional<Column<b>Entity</b>> result = entityManager.singleResult(query);
   entityManager.delete(delete);
}
```





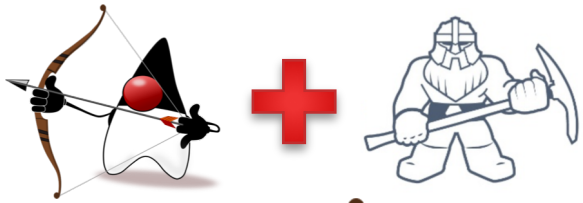


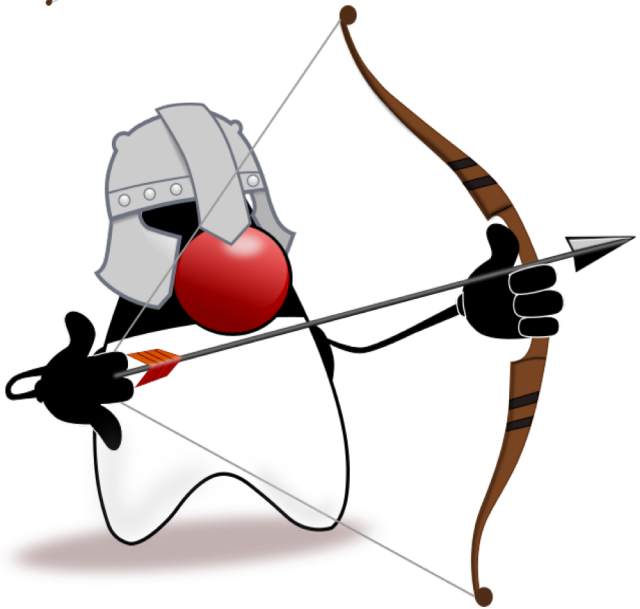
JNoSQL

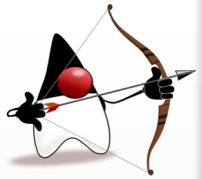


Eclipse JNoSQL - Artemis

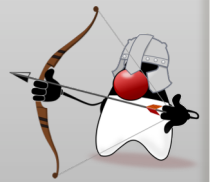
- Based on CDI and Diana
- Heavy use of Annotations
- Events on Create, Update, Delete
- Bean Validation Support
- Configurable and Extensible
- “Query by Methods”





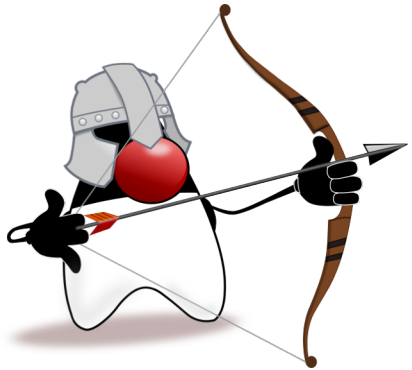


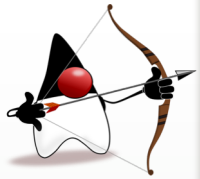
JNoSQL



Names & Definitions

- Annotated Entities
- Template
- Repository
- Configuration

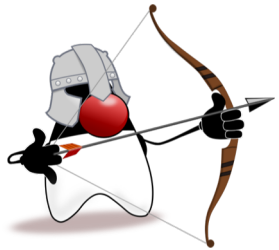


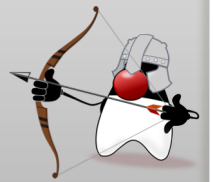


Annotaded Entities

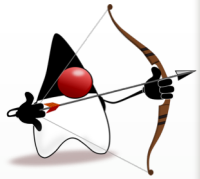
- MappedSuperclass
- Entity
- Column

```
@Entity(”god")
public class God {

  @Column
  private String name;

  @Column
  private Set<God> siblings;

…
}
```

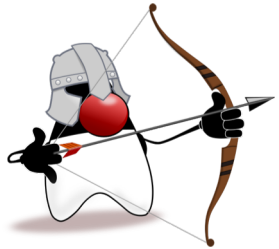




JNoSQL



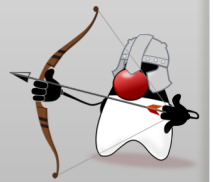
Templates

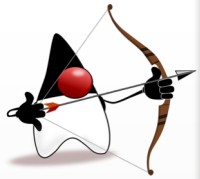


JNoSQL

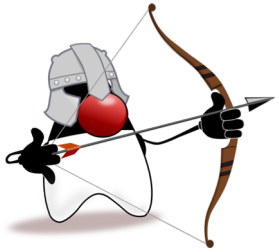
import static DocumentQuery.select;
@Inject DocumentTemplate template;

```
God diana = God.builder().withName("Diana");
template.insert(diana);
template.update(diana);

DocumentQuery query =
select().from("god").where("name").equals("Diana").build();
List<God> gods = template.select(query);
```



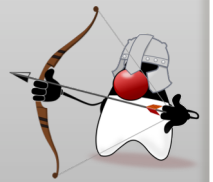


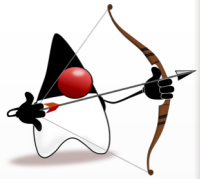
Repository



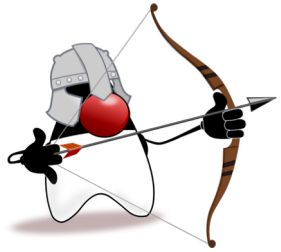
JNoSQL

```
@Inject
@Database(DatabaseType.COLUMN)
private GodRepository godRepository;

@Inject
@Database(DatabaseType.KEY_VALUE)
private GodRepository godRepository;
```

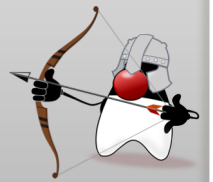




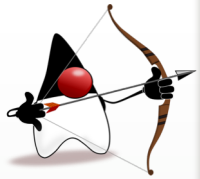
Configuration



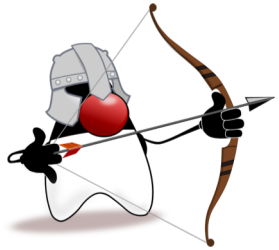
JNoSQL



```
[ {
  "description": "The couchbase document configuration",
  "name": "document",
  "provider":
"org.jnosql.diana.couchbase.document.CouchbaseDocumentConfiguration",
  "settings": {
    "couchbase-host-1": "localhost",
    "couchbase-user": "root",
    "couchbase-password": "123456"
  }
} ]
```

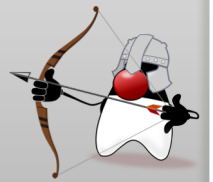


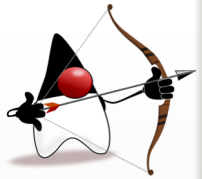
Configuration



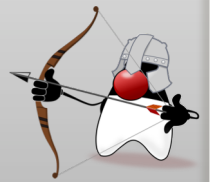
```
@Inject
@ConfigurationUnit
private DocumentCollectionManagerFactory<?> entityManager;
```

JNoSQL



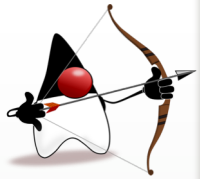


JNoSQL

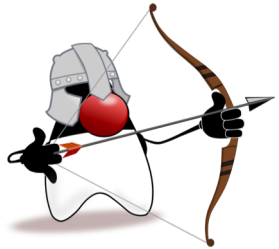


Hands-On!

https://github.com/JNOSQL/oc1-hands-on-2018



Data model for the use cases



JNoSQL

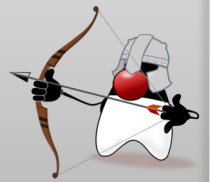
JUGs and JUG Members!

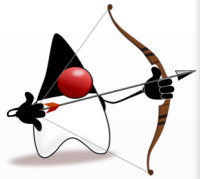
JUG

- Name
- City
- Programming Languages
- Country

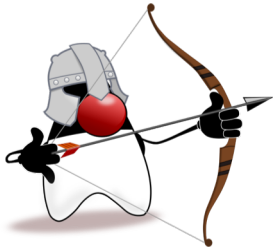
JUG Member

- Name
- City
- Programming Languages (name and skill level)

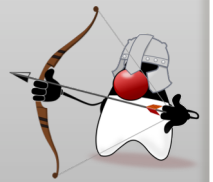




Use case #1 - Key/Value

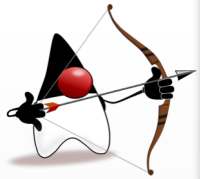


JNoSQL

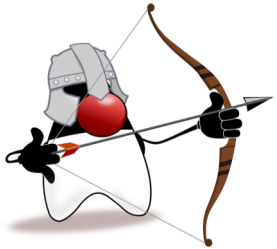


Create a database to handle JUG information using Redis

- Create, Retrieve and Update JUG information
- Model once and reuse the model with different database



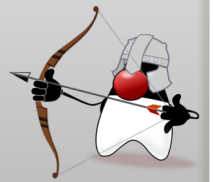
Use case #2 - Document

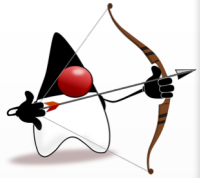


JNoSQL

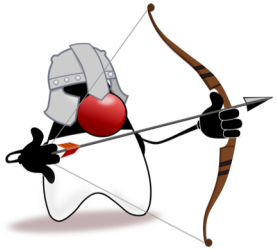
Implement some searches against a MongoDB

- Search JUG members in a given city
- Search JUG members of legal drinking age
- Search JUGs in a region

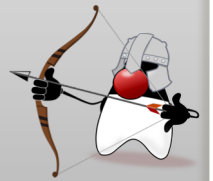




Use case #3 - Graph

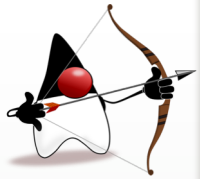


JNoSQL

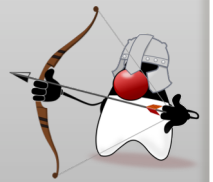


Implement recommendation searches against Neo4J

- Find Beginner Java Users that know Advanced Java User(s).
- Find Java Users in a given City
- Recommend Advanced Java Users in the same City as a given User.



JNoSQL



Thanks!

https://www.tomitribe.com/codeone/hol5998/

#CodeOne #JNoSQL @JNoSQL @HillmerCh @leomrlima @otaviojava @patricia_uz @vilojona