Otávio Santana @otaviojava

Werner Keil @wernerkeil

About the Speakers

Werner Keil

Twitter @wernerkeil

- Consultant – Coach
- Open Source Evangelist
- Software Architect
- Apache, Eclipse Committer,
  Agile Alliance Member
- Expert Group member in many JSRs
- Spec Lead – JSR354, JSR385
- Jakarta EE Specification Committee Member

[www.linkedin.com/in/catmedia]

About the Speakers

Otávio Goncalves de Santana

Software engineer, Tomitribe

Java Champion, SouJava JUG Leader

Apache, Eclipse and OpenJDK Committer

Expert Group member in many JSRs

Spec Lead – JSR 354, JSR 385

JNoSQL Project Lead

Representative at JCP EC for SouJava

[www.linkedin.com/in/otaviojava]

NoSQL

Not Only SQL

01 Database

02 Doesn't use structure

03 No Transactions

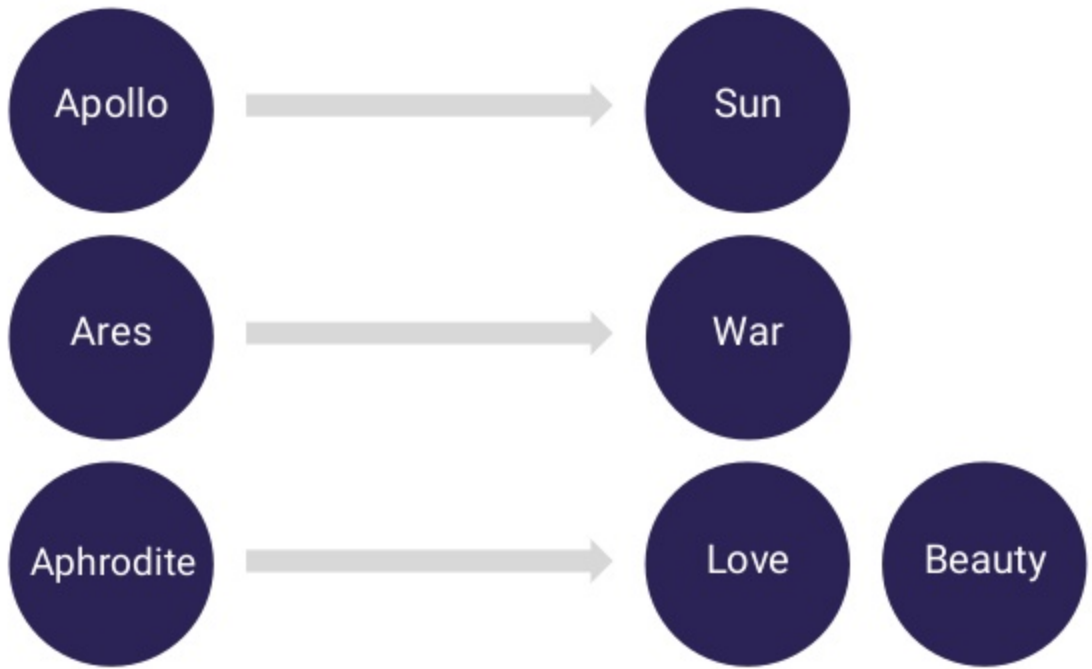
04 Base

05 Five different types

Key Value

NotOnlySQL

- AmazonDynamo
- AmazonS3
- Redis
- Hazelcast



Column Family

Not only SQL

HBase

Cassandra

Scylla

Clouddata

SimpleDb

DynamoDB

Document

NoSQL

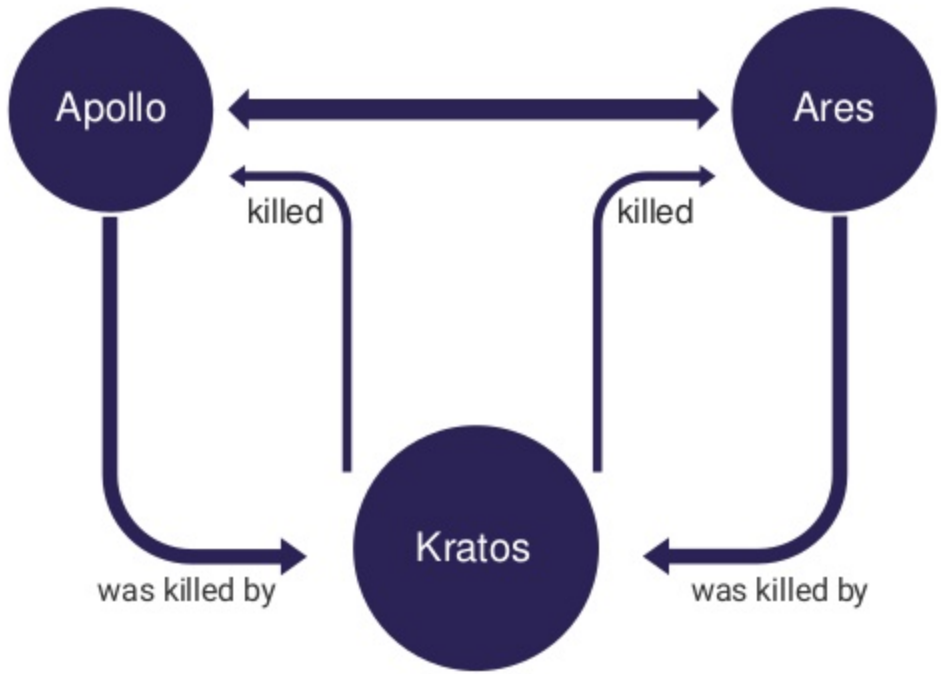
- ApacheCouchDB
- MongoDB
- Couchbase

```
{
  "name":"Diana",
  "duty":[
    "Hunting",
    "Moon",
    "Nature"
  ],
  "siblings":{
    "Apollo":"brother"
  }
}
```

Graph

Not only SQL

- Neo4j
- InfoGrid
- Sones
- HyperGraphDB



Multi-Model

Not Only SQL

01 OrientDB (graph, document)

02 Couchbase (key value, document)

03 ArangoDB (document, graph, key-value)

04 Elasticsearch (document, graph)

SQL vs NoSQL

Not only SQL

SQL	KEY-VALUE	COLUMN	DOCUMENTS	GRAPH
Table	Bucket	Column family	Collection	
Row	Key/value pair	column	Documents	Vertex
Column		Key/value pair	Key/value pair	Vertex and Edge property
Relationship			Link	Edge

BASE vs ACID

NoSQL

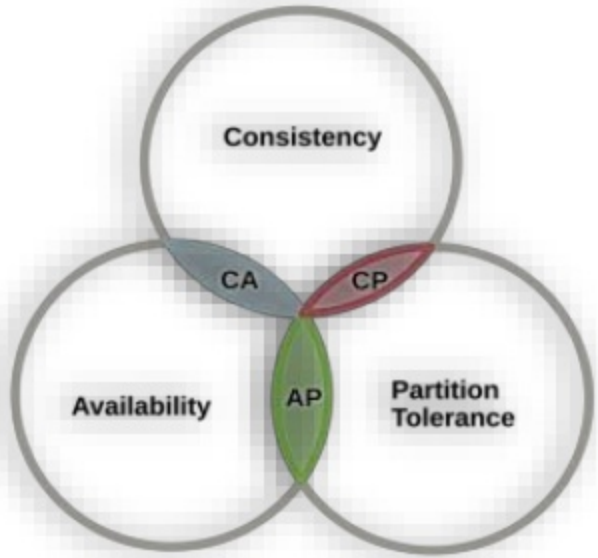
- Basically Available
- Soft state
- Eventual consistency

- Atomicity
- Consistency
- Isolation
- Durability

CAP

NoSQL


MySQL
MariaDB

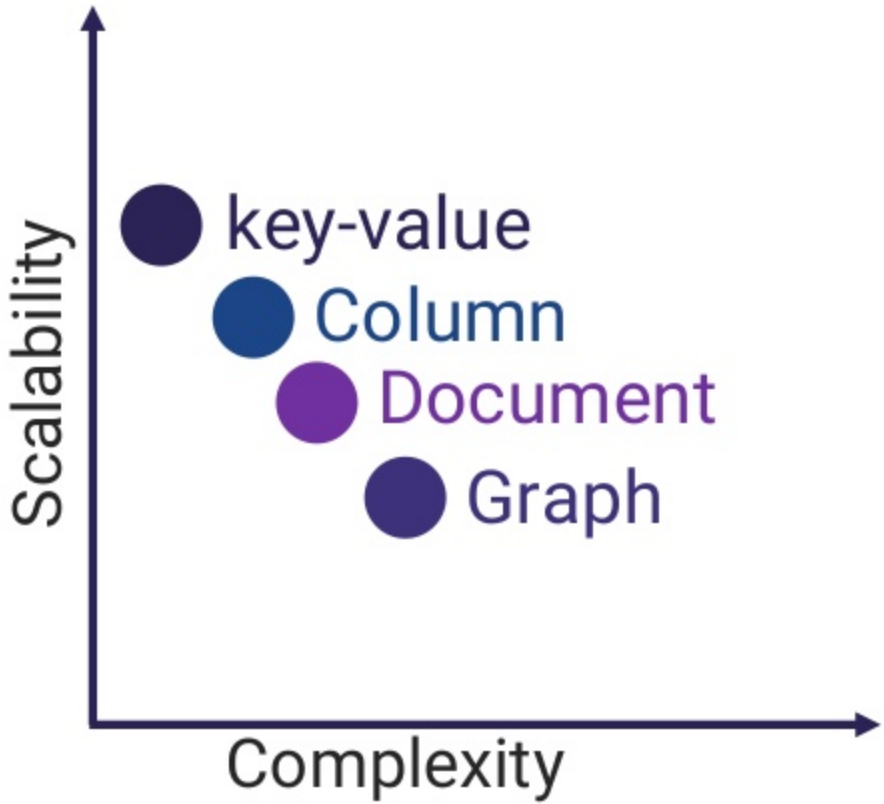

Consistency
CA
CP
Availability
AP
Partition
Tolerance


mongoDB.
redis

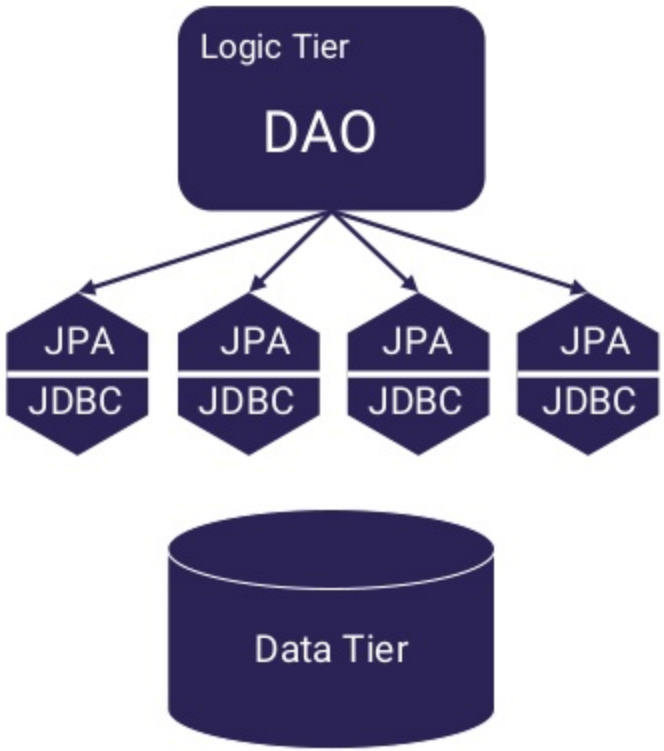

cassandra
riak

Scalability vs Complexity

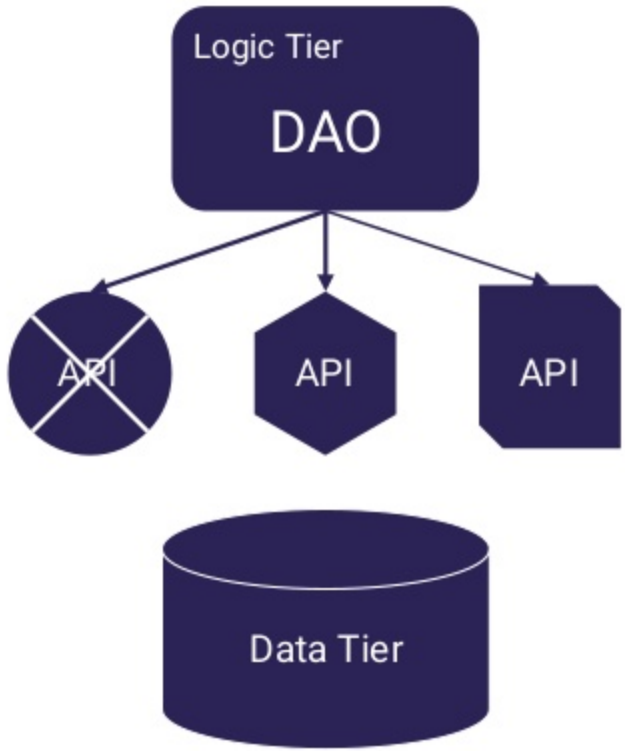
NoSQL



Relational Application



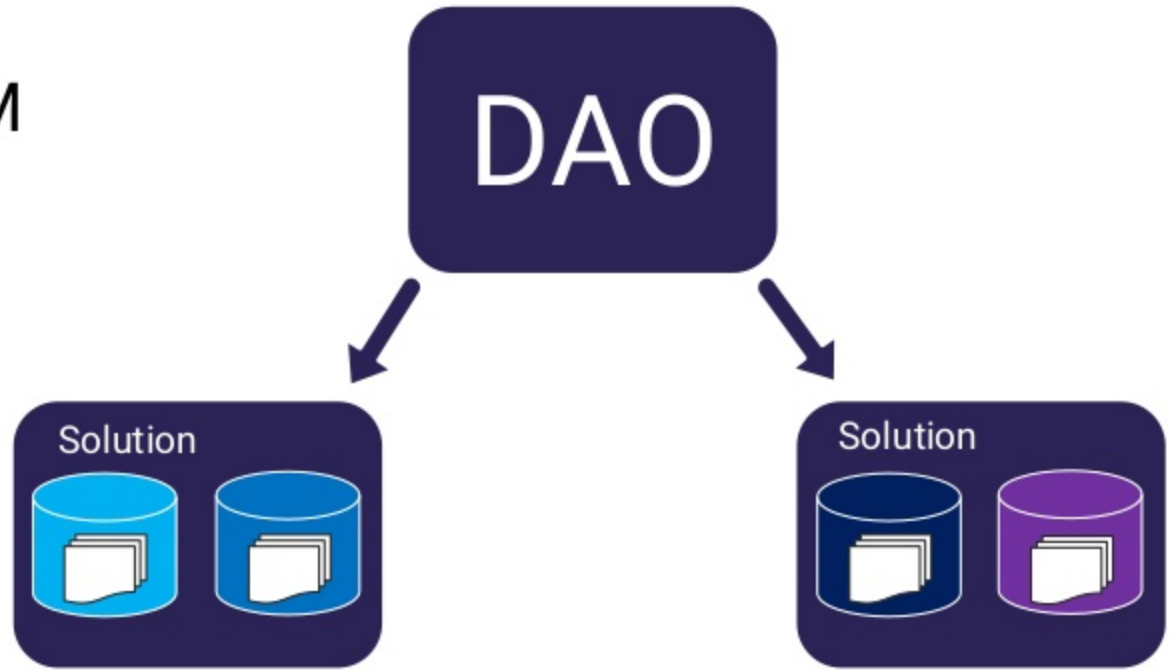
NoSQL Application



The Current Solution

Hibernate OGM

TopLink



JPA problem for NoSQL

Not Only SQL

01 Saves Async

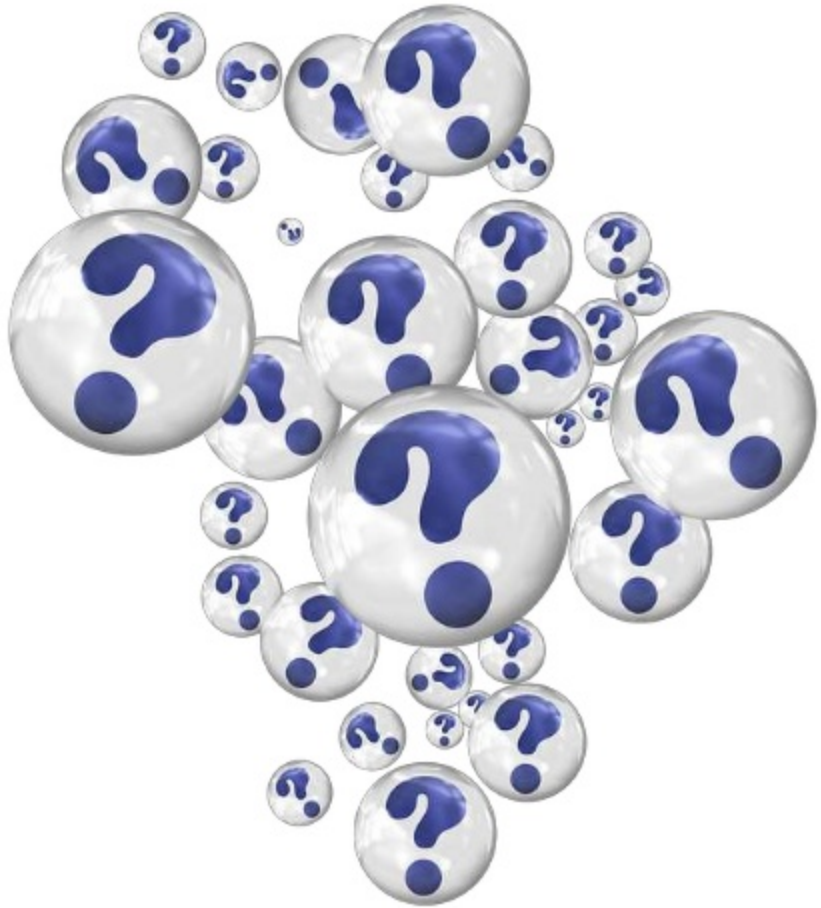
02 Async Callback

03 Time to Live (TTL)

04 Consistency Level

05 SQL based

06 Diversity in NoSQL



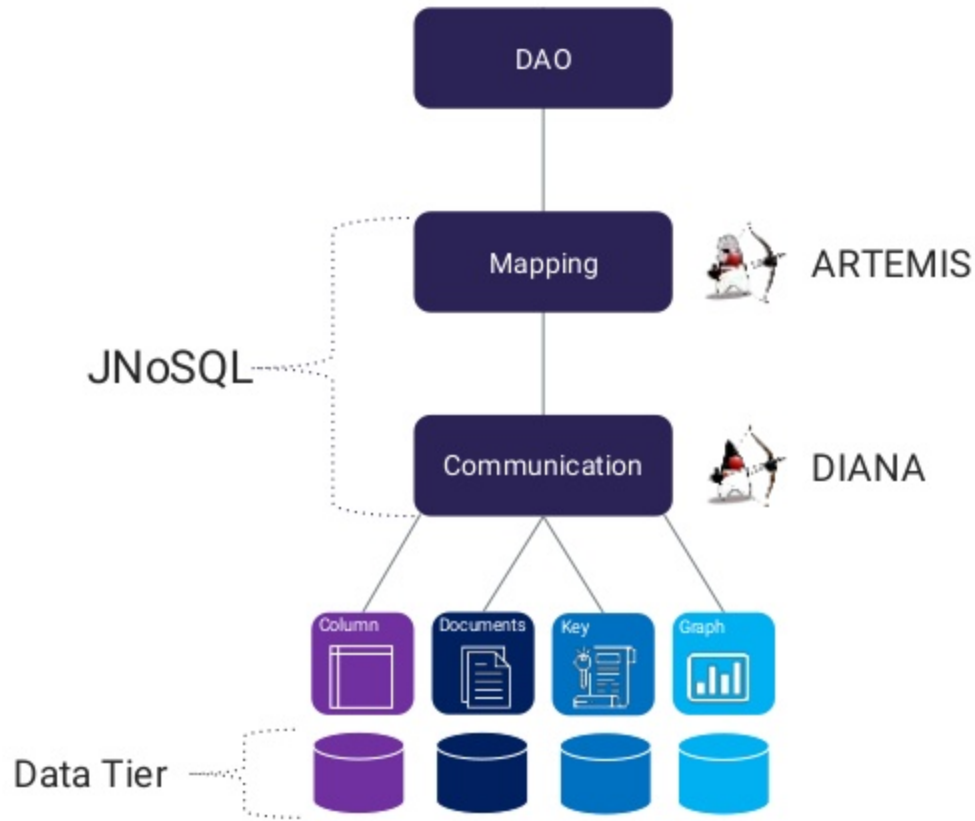
The Eclipse JNoSQL Solution

01 Mapping API

02 Communication API

03 No lock-in

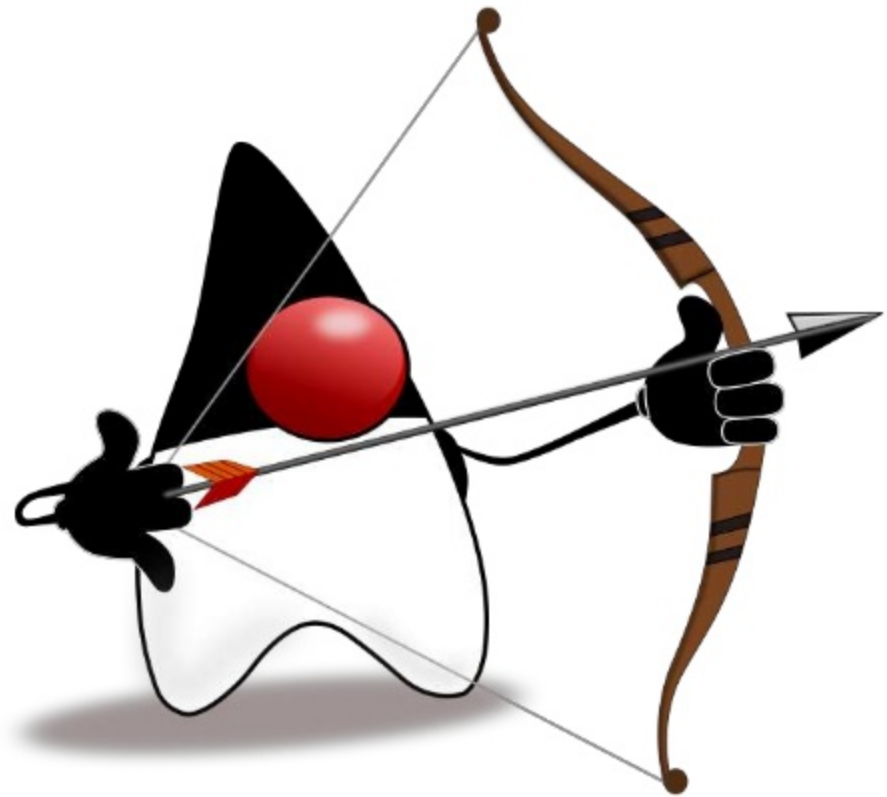
04 Divide and Conquer



Diana

Not Only SQL

- API Communication layer
- Document, key-value, Column, Graph (TinkerPop)



Communication Issue


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

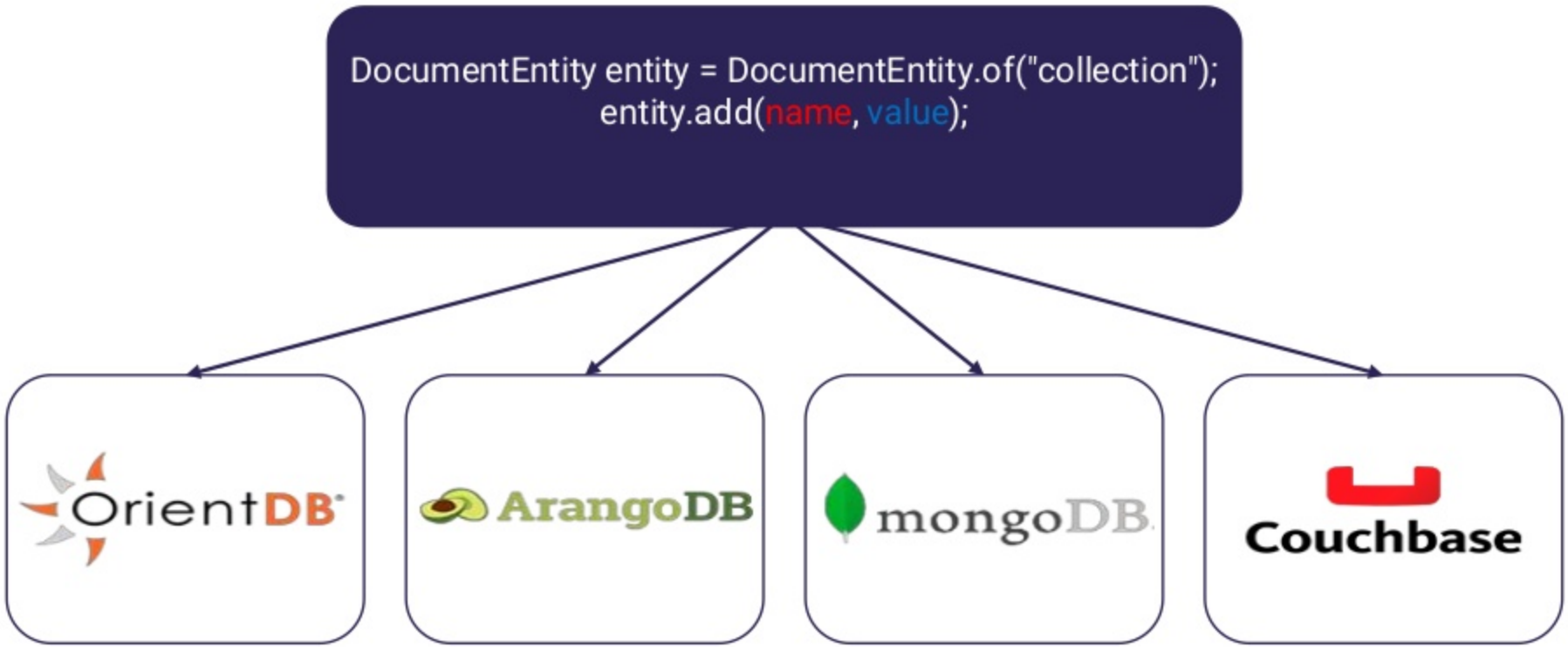

```
Document document = new Document();
document.append(name, value);
```


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


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

Eclipse JNoSQL

```
DocumentEntity entity = DocumentEntity.of("collection");
entity.add(name, value);
```



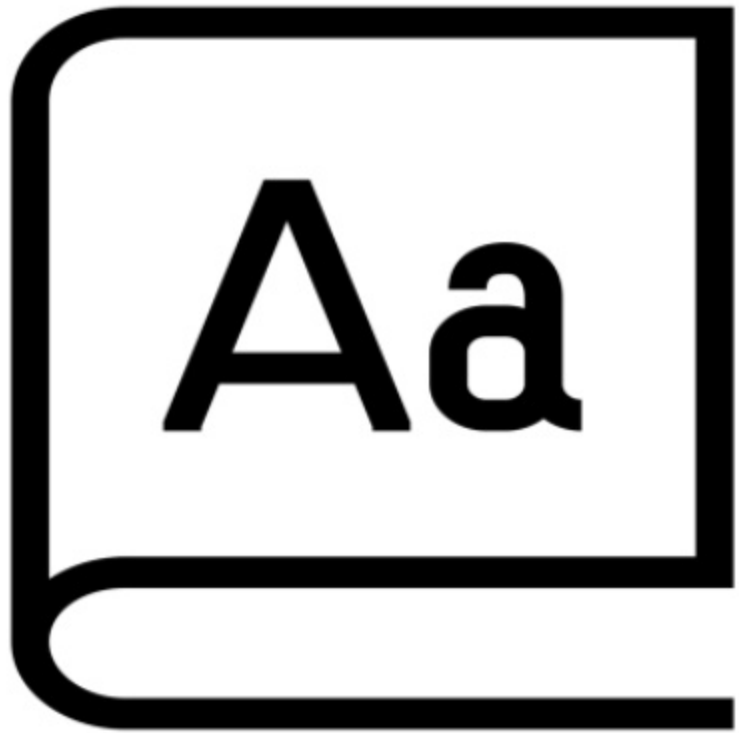
Names & Definitions

01 Configuration

02 Factory

03 Manager

04 Entity



Names & Definitions

```
ColumnConfiguration<?> configuration = new DriverConfiguration();

try(ColumnFamilyManagerFactory managerFactory = configuration.get()) {

    ColumnFamilyManager entityManager =
managerFactory.get(KEY_SPACE);

    entityManager.insert(entity);

    ColumnQuery select = select().from(COLUMN_FAMILY)
                                 .where("id").eq("Ada").build();

    ColumnDeleteQuery delete = delete().from(COLUMN_FAMILY)
                                       .where("id").eq("Ada").build();

   Optional<ColumnEntity> result = entityManager.singleResult(query);

   entityManager.delete(delete);
```

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)));
```

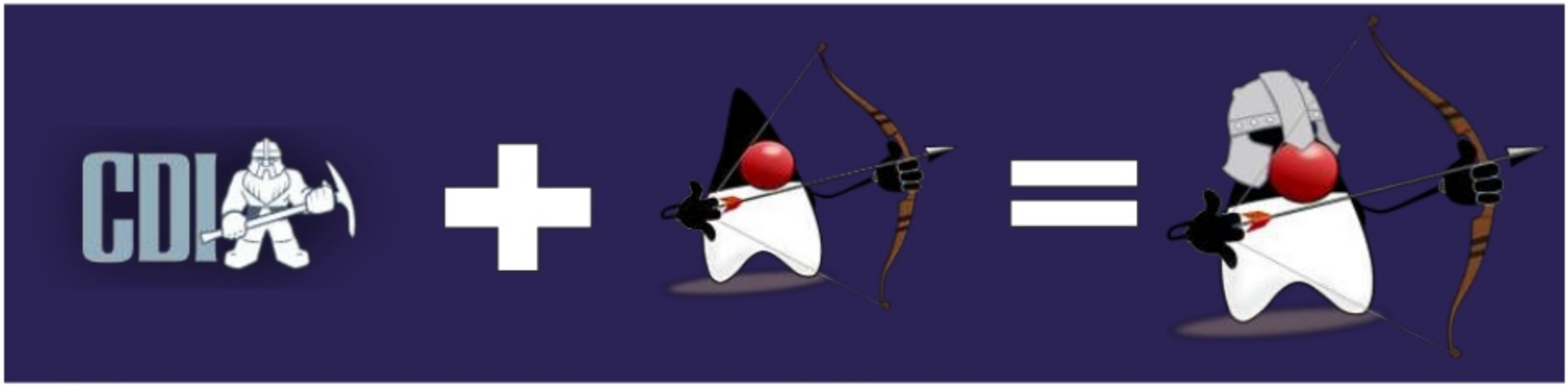
cassandra APACHE HBASE

```
//mutiple implementation
entityManager.insert(entity);
ColumnQuery query =
select().from(COLUMN_FAMILY).where("id").eq(10L).build();
Optional<ColumnEntity> result = entityManager.singleResult(query);
```

cassandra

```
//cassandra only
List<ColumnEntity> entities = entityManagerCassandra
.cql("select * from newKeySpace.newColumnFamily where id=10;");
entityManagerCassandra.insert(entity, ConsistencyLevel.ALL);
```

Artemis



01 CDI Based

02 Diana Based

03 Annotation Based

07 Query Method

04 Events to insert, delete, update

05 Supports to Bean Validation

06 Configurable and Extensible

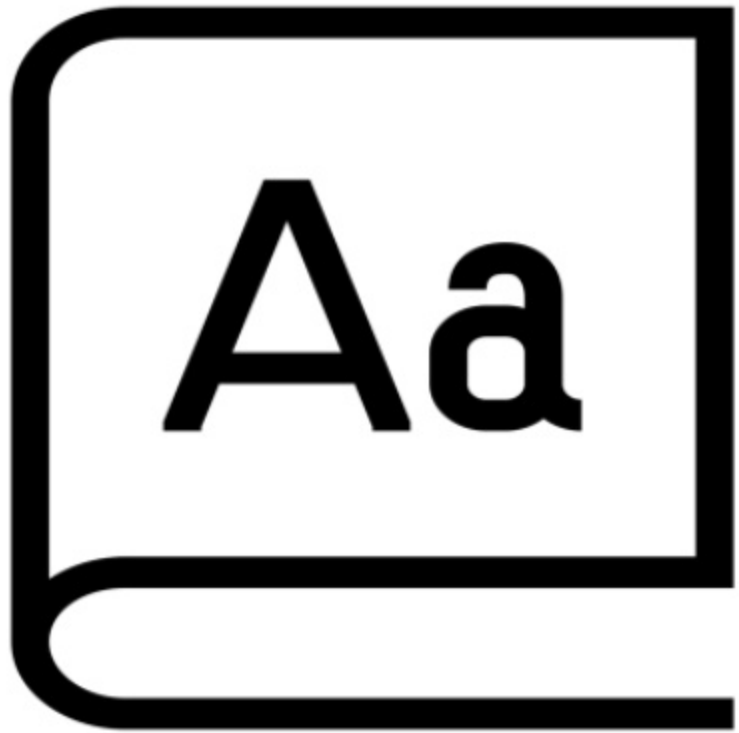
Names & Definitions

01 Annotated Entities

02 Template

03 Repository

04 Configuration



Annotated Entities

01 Mapped Superclass

02 Entity

03 Column

```
@Entity("god")
public class God {

  @Column
  private String name;

  @Column
  private long age;

  @Column
  private Set<String> powers;
}
```

Template

```
God artemis = ...;

DocumentTemplate template = …

template.insert(artemis);

template.update(artemis);

DocumentQuery query = ...

List<God> gods = template.select(query);
```

Repository

```
interface GodRepository extends Repository<God, String> {

    Optional<God> findByName(String name);

    Stream<God> findByNameAndAgeOrderByName(String name, Integer age);

}
```

Repository

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

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

Support for

JSON

XML

YAML

Configuration

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

Diversity

```
@Entity("god")
public class God {

@Column
private String name;

@UDT("weapon")
@Column
private Weapon weapon;

}
```

```
interface GodRepository extends
CassandraRepository<God, String> {

  @CQL("select * from God where name = ?")
  List<God> findByName(String name);

}
```

cassandra

We have Query by Text

```
DocumentTemplate documentTemplate = ...;
ColumnTemplate columnTemplate = ...;
KeyValueTempalte keyValueTemplate =...;
GraphTemplate graphTemplate =...;
List<Movie> movies = documentTemplate.query("select * from Movie where year > 2012");
List<Person> people = columnTemplate.query("select * from Person where age = 25");
Optional<God> god = keyValueTemplate.query("get \"Diana\"");
List<City> cities = graphTemplate.query("g.V().hasLabel('City')");
```

We have Query by Text

```
PreparedStatement preparedStatement = template.prepare("select * from
Person where name = @name");
preparedStatement.bind("name", "Ada");
List<Person> adas = preparedStatement.getResultList();
//to graph just keep using gremlin
PreparedStatement prepare =
graphTemplate().prepare("g.V().hasLabel(param)");
prepare.bind("param", "Person");
List<Person> people = preparedStatement.getResultList();
```

We have Query by Text

```
interface PersonRepository extends Repository<Person, Long> {
    @Query("select * from Person")
    Optional<Person> findByQuery();
    @Query("select * from Person where id = @id")
   Optional<Person> findByQuery(@Param("id") String id);
}
```

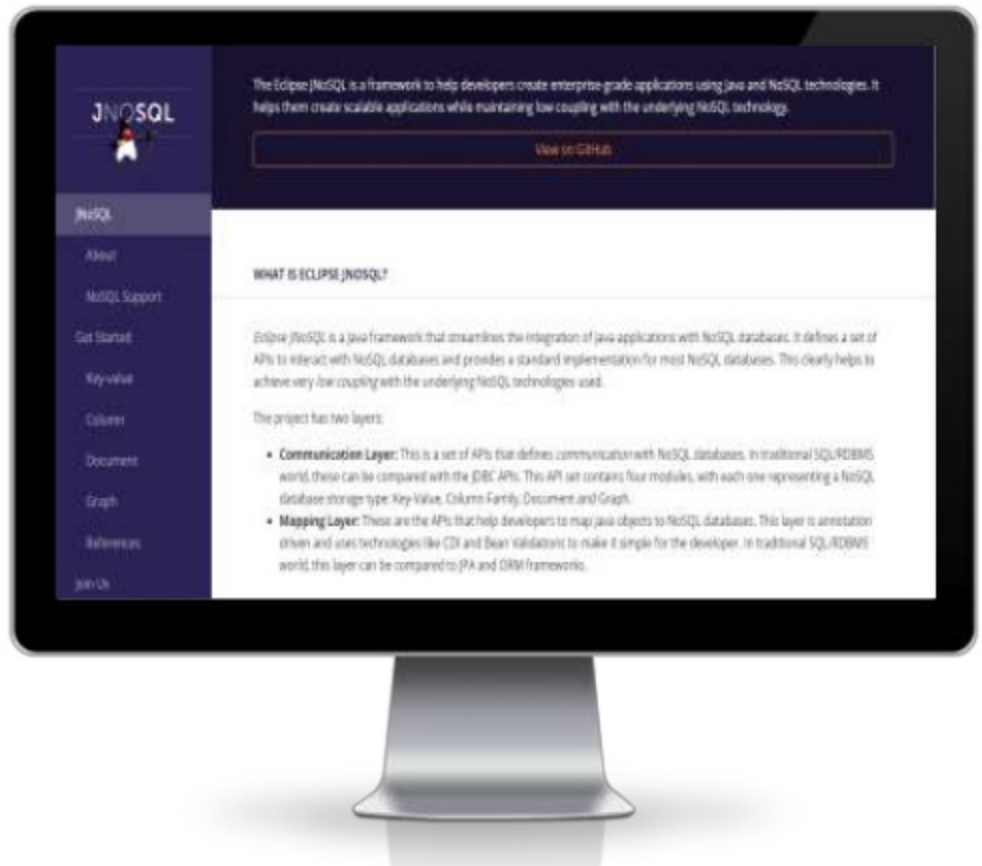
Demo

Configuration

JNoSQL

CDI 2.0 with Java SE

Hazelcast
MongoDB
Neo4J



NoSQL Providers

Road Map

Draft and code proposal

Community Feedback

Involve NoSQL Vendors

Involve Solution Vendors

Eclipse Project

Development

Specification Process

- Java EE now contributed to Eclipse Foundation
- Jakarta EE
- Code First

  Jakarta EE forwards with new specifications
- Hopefully a new spec and namespace confirmed around Oracle Code One / ECE

  “jakarta.nosql”

  See https://www.tomitribe.com/blog/jnosql-and-jakarta-ee/

JUGs/Communities

SOUJava
sociedade de usuários java

UAI JUG

DFJUG

Javace

JavaBahia

LJC

GUJavaSC

Redfoot JUG

References

Communication API
Support to Async operations
APIs

Mapping API
Bean Validation
Events
Repository
Template

Query by text
Prepared Statement

http://jnosql.org/
https://github.com/eclipse?q=Jnosql
https://dev.eclipse.org/mailman/listinfo/jnosql-dev
https://gitter.im/JNOSQL/developers
https://wiki.eclipse.org/JNoSQL

JNOSQL

Thank you

Otávio Santana @otaviojava

Werner Keil @wernerkeil