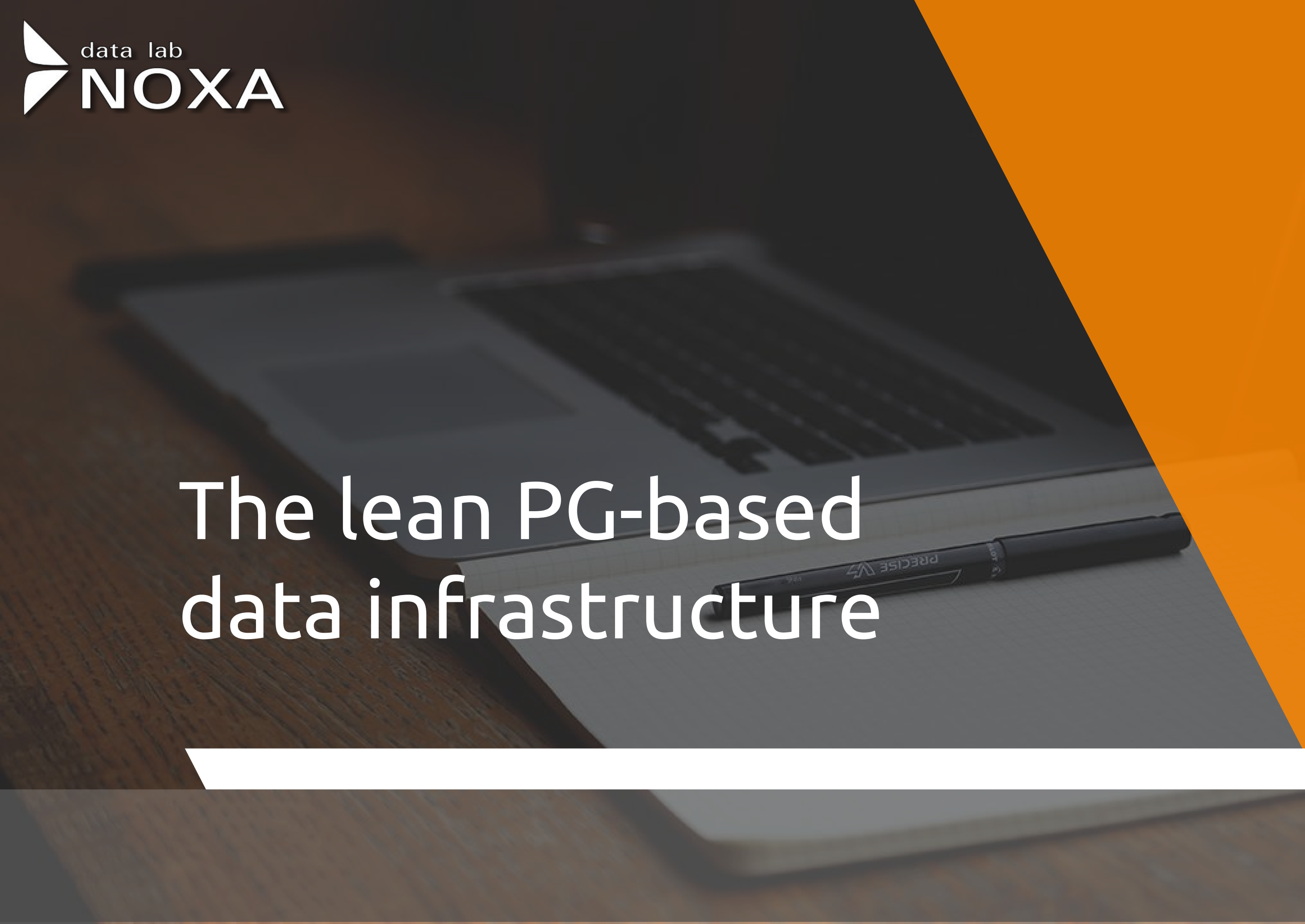
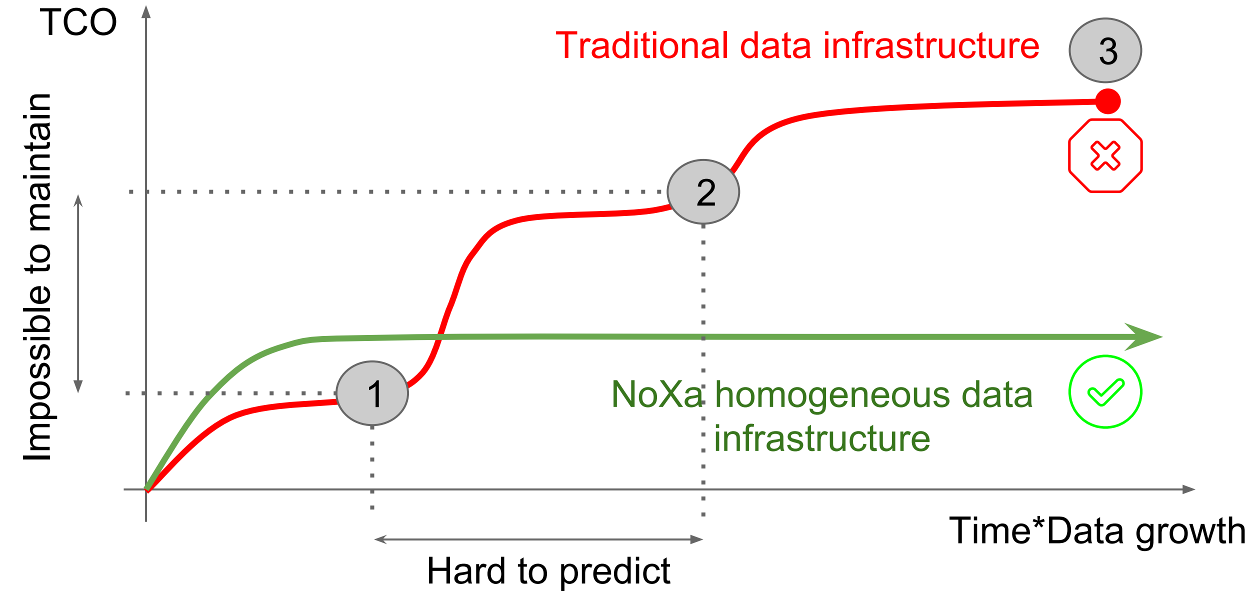
data lab NOXA



# The lean PG-based data infrastructure

# Business challenges while data is growing exponentially

- NoXa simply works



# Structure of risks

1 OLAP\OLTP contradiction - the data ecosystem must be divided in at least 2 approaches of modelling and design. TX purity loss. Availability loss due the index and partition management.

RDBMS still works at least in shared clusters

2 Single source of truth management requires expensive MDM solutions, different data domains requires different resources and competences. Complex multi layered design (TX layer, Integration layer, Data Mart layer) is hard to maintain.

Data Quality, MDM, ETL\Replication tools and Database Appliance are required

3 Data generates data. Single point of truth management is not possible due the lack of ACID support. Efficient data modelling is not possible. ETL \ Replication generates redundant entities. Business spend more and more money to maintain, evaluate BIG DATA approach and loses control on data infrastructure.

Big Data is required, Data Infrastructure becomes a Data Lake with dirty water

***NoXa stands for "NO distributed transactions" in homogeneous shared nothing clusters***

# 1.

# We introduce Smart Data Concept

“

- *All data must be relational*
- *Data must be virtually centralized*
- *Data must be normalized*
- *Data must durable*
- *Data must be transactional*
- *The performance, availability must be under control of business in arbitrary expense*
- *The data volume does not matter*
- *The data must be integration friendly*

”

# 7 normal form is the fundamental of Data Quality

- **Highest theoretically possible normalization level**

Allows to create high quality design of logical data schema without anomalies.

- **Insert only data storage footprint**

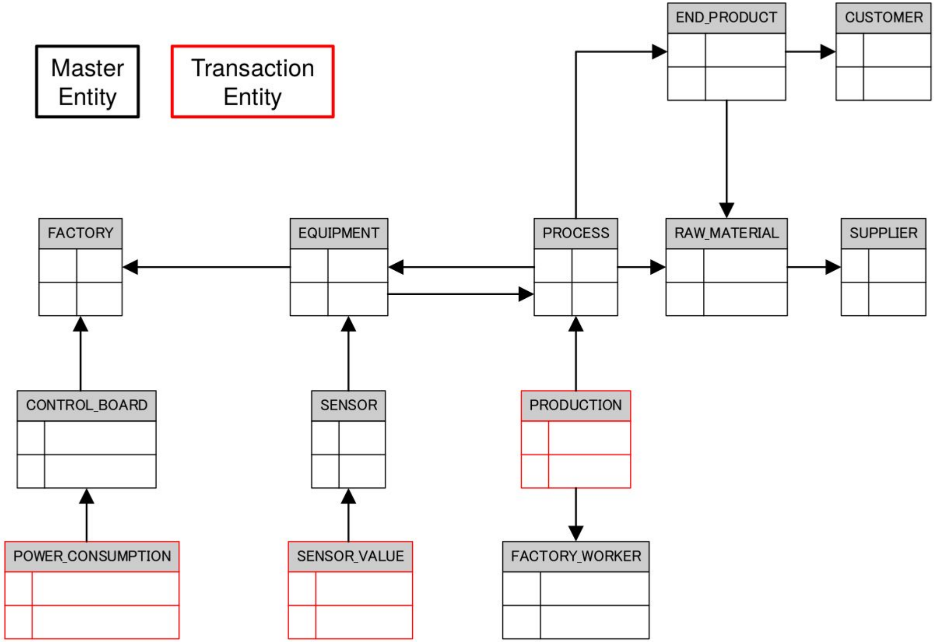
Allows to use HPC&MPP approaches in a block free manner even on system IO level.

- **All entities are replicated and distributed among shared nothing cluster**

High performance distributed partial “SELECT”s from indexed storage with on node “JOIN” and “FORK JOIN” in memory aggregation.

- **Single point of truth is a guaranty**

Master data is the PART of the SCHEMA, no need to use expensive tools to maintain it.



# 7 normal form is the key of productivity

- **Modeling methodology**

Defines rules and best practices as well as Software Development Process

- **NO WAL\REDO LOGGING**

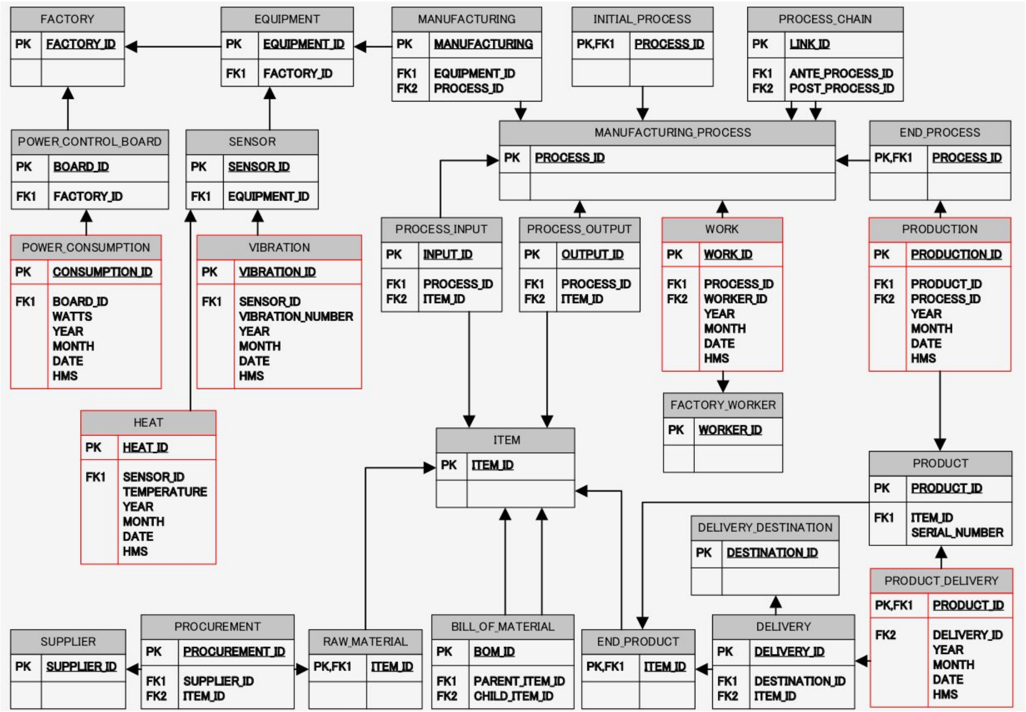
Allows to use HPC&MPP approaches in a block free manner even on system IO level.

- **Simple and elegant**

Key/Value storage with Core and Attribute reference

- **Logical Updates and Delete**

Also implemented as insert for separate tables



| FACTORY | |
|---|---|
| PK | FACTORY_ID |

| EQUIPMENT | |
|---|---|
| PK | EQUIPMENT_ID |
| FK1 | FACTORY_ID |

| MANUFACTURING | |
|---|---|
| PK | MANUFACTURING |
| FK1<br>FK2 | EQUIPMENT_ID<br>PROCESS_ID |

| INITIAL_PROCESS | |
|---|---|
| PK,FK1 | PROCESS_ID |

| PROCESS_CHAIN | |
|---|---|
| PK | LINK_ID |
| FK1<br>FK2 | ANTE_PROCESS_ID<br>POST_PROCESS_ID |

| POWER_CONTROL_BOARD | |
|---|---|
| PK | BOARD_ID |
| FK1 | FACTORY_ID |

| SENSOR | |
|---|---|
| PK | SENSOR_ID |
| FK1 | EQUIPMENT_ID |

| MANUFACTURING_PROCESS | |
|---|---|
| PK | PROCESS_ID |

| END_PROCESS | |
|---|---|
| PK,FK1 | PROCESS_ID |

| POWER_CONSUMPTION | |
|---|---|
| PK | CONSUMPTION_ID |
| FK1 | BOARD_ID<br>WATTS<br>YEAR<br>MONTH<br>DATE<br>HMS |

| VIBRATION | |
|---|---|
| PK | VIBRATION_ID |
| FK1 | SENSOR_ID<br>VIBRATION_NUMBER<br>YEAR<br>MONTH<br>DATE<br>HMS |

| PROCESS_INPUT | |
|---|---|
| PK | INPUT_ID |
| FK1<br>FK2 | PROCESS_ID<br>ITEM_ID |

| PROCESS_OUTPUT | |
|---|---|
| PK | OUTPUT_ID |
| FK1<br>FK2 | PROCESS_ID<br>ITEM_ID |

| WORK | |
|---|---|
| PK | WORK_ID |
| FK1<br>FK2 | PROCESS_ID<br>WORKER_ID<br>YEAR<br>MONTH<br>DATE<br>HMS |

| PRODUCTION | |
|---|---|
| PK | PRODUCTION_ID |
| FK1<br>FK2 | PRODUCT_ID<br>PROCESS_ID<br>YEAR<br>MONTH<br>DATE<br>HMS |

| FACTORY_WORKER | |
|---|---|
| PK | WORKER_ID |

| HEAT | |
|---|---|
| PK | HEAT_ID |
| FK1 | SENSOR_ID<br>TEMPERATURE<br>YEAR<br>MONTH<br>DATE<br>HMS |

| ITEM | |
|---|---|
| PK | ITEM_ID |

| PRODUCT | |
|---|---|
| PK | PRODUCT_ID |
| FK1 | ITEM_ID<br>SERIAL_NUMBER |

| DELIVERY_DESTINATION | |
|---|---|
| PK | DESTINATION_ID |

| PRODUCT_DELIVERY | |
|---|---|
| PK,FK1 | PRODUCT_ID |
| FK2 | DELIVERY_ID<br>YEAR<br>MONTH<br>DATE<br>HMS |

| SUPPLIER | |
|---|---|
| PK | SUPPLIER_ID |

| PROCUREMENT | |
|---|---|
| PK | PROCUREMENT_ID |
| FK1<br>FK2 | SUPPLIER_ID<br>ITEM_ID |

| RAW_MATERIAL | |
|---|---|
| PK,FK1 | ITEM_ID |

| BILL_OF_MATERIAL | |
|---|---|
| PK | BOM_ID |
| FK1<br>FK2 | PARENT_ITEM_ID<br>CHILD_ITEM_ID |

| END_PRODUCT | |
|---|---|
| PK,FK1 | ITEM_ID |

| DELIVERY | |
|---|---|
| PK | DELIVERY_ID |
| FK1<br>FK2 | DESTINATION_ID<br>ITEM_ID |

# Redundancy is the key of Durability and Performance

- **Logical changes are decoupled and broadcasted as micro transactions among virtual cluster and multiple data centers**

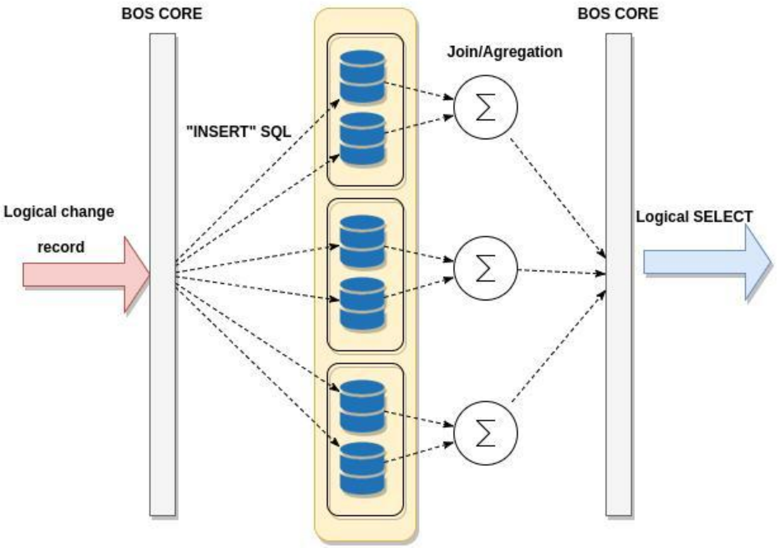
There is no synchronized log based replication, just small Inserts in underlying databases. No FSYNC, No BLOCKING, Real time.

- **Out of the box 6X redundancy per one logical node**

Provide almost 6X faster selects and processing due the MULTITHREADED lock free NUMA optimized computational nodes.

- **Redundancy means durability**

At the same time eliminates the Zero Point of failure in 6x ratio.



# ACID and MVCC is defined by application on demand during SELECT

- **Business applications operate views**

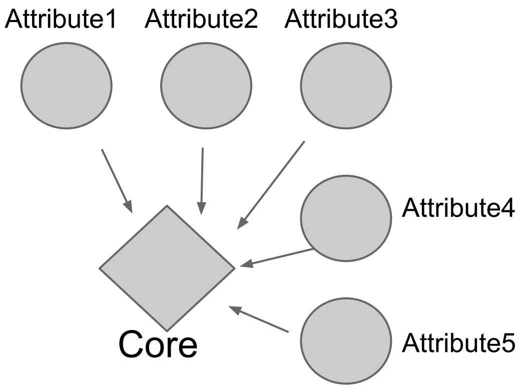
BOS application joines data internally and provides data consumers with SQL like tables - so called views.

- **ACID is implemented by referencing the CORE ID, which plays the role of XID in RDMS**

*Insert algorithm:*

1. *Prepare CORE ID with definite timestamp.*
2. *Insert In Attribute 1 tables.*
3. *Insert In Attribute 2 tables.*
4. *Insert In Attribute 3 tables.*
5. *Insert In Attribute 4 tables.*
6. *Insert In Attribute 5 tables.*
7. *Insert in CORE ID tables.*

If something goes wrong here, CORE ID is never inserted, and the result tuple is never formed during SELECT referencing this CORE ID. That means that logical ROLLBACK is performed.
If everything goes fine, the CORE ID is inserted and is available for select. This corresponds to logical COMMIT.



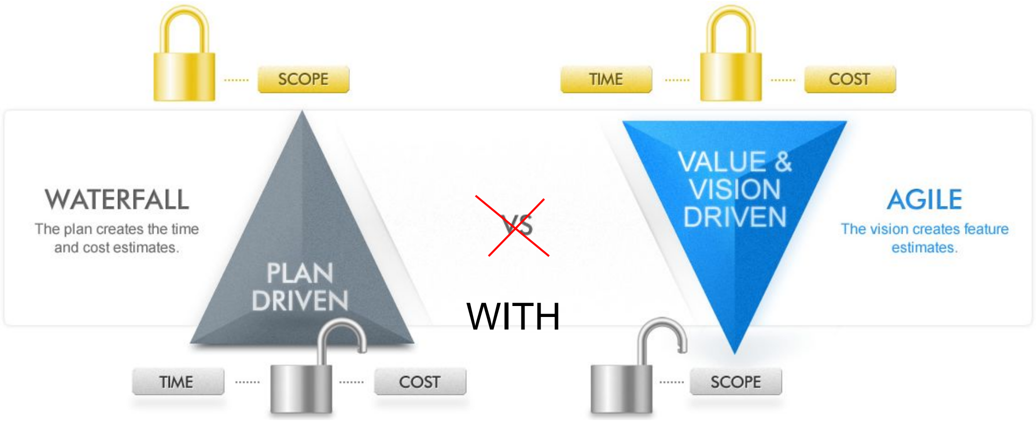
# NoXA

First in the word, brings the state of the art data infrastructure where ACID is guaranteed among 10k> distributed tables during SELECT rather than TX Logging, Distributed XA TX, and multiphase commit.

# 2. Software development process

# We take the best from two worlds

- Small iteration cycles, and scrum alike backlog maintenance.
- Accurate time to market planning and continuous integration.
- Close business integration
- Sustainable agility of architecture, design and data model.
- Best software engineering practices.


SCOPE
TIME
COST
WATERFALL
The plan creates the time and cost estimates.
PLAN DRIVEN
TIME
COST
~~VS~~
WITH
VALUE & VISION DRIVEN
AGILE
The vision creates feature estimates.
SCOPE

# Java is the CORE, but BOS is more than Java

- **JPA alike Java 1.8 SDK**

All entities are implemented as distinct parameterized types (generics).

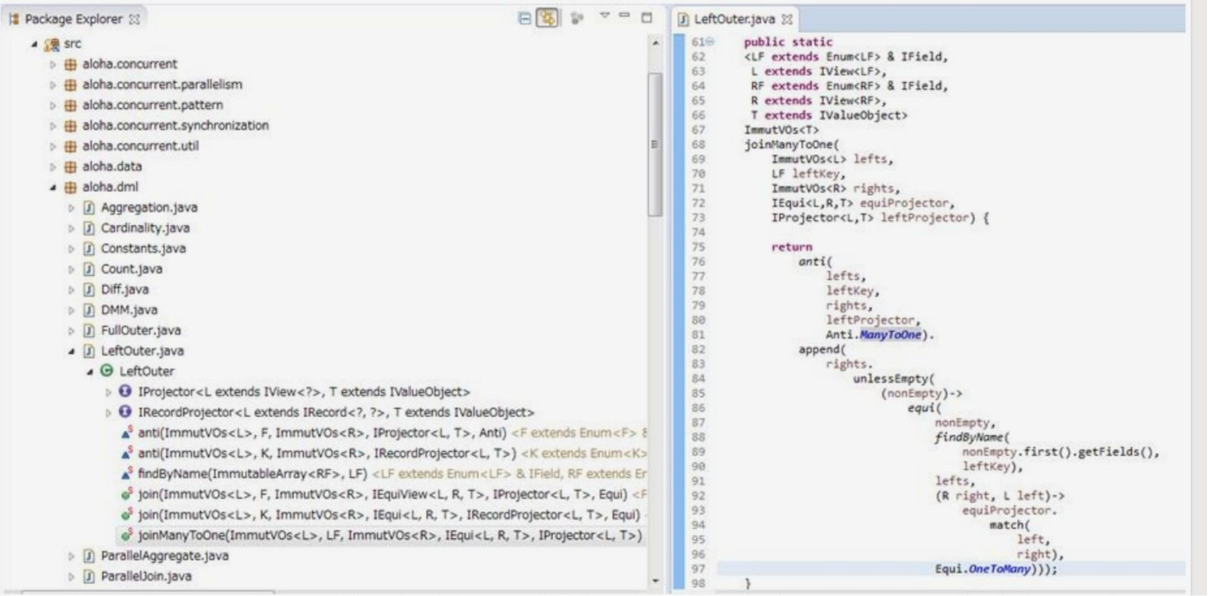
Relational model implemented and checked in a compile time verifying the boundaries of parameter types before type erasure.

- **Unit test free**

The BOS SDK design is focused on efficiency of programming, predefined rules and quality control keep code coverage rate high even without unit testing.

- **No local variables, everything is IMMUTABLE**

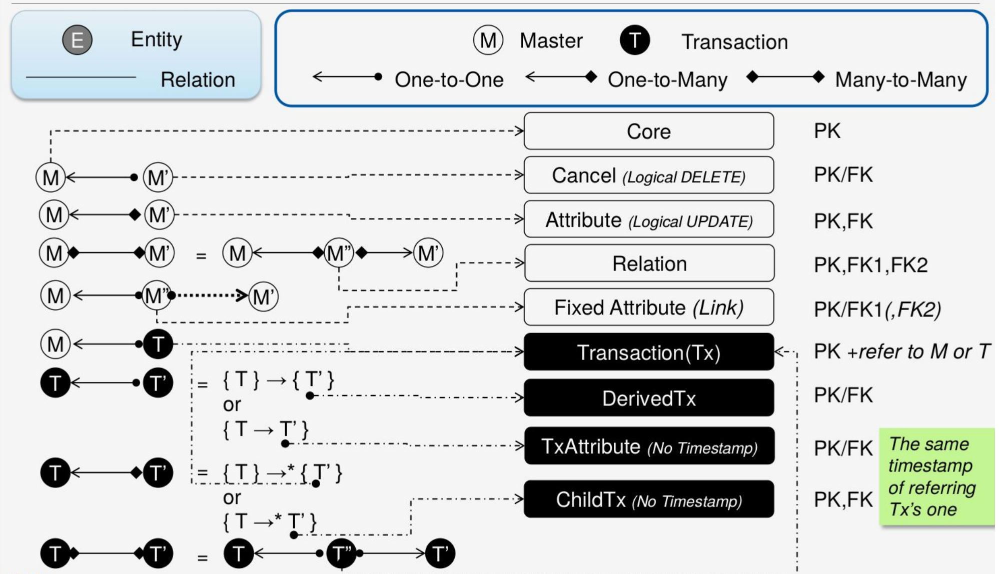
All data in BOS is immutable. But System and Domain Layer is mutually exclusive



```
src
  aloha.concurrent
  aloha.concurrent.parallelism
  aloha.concurrent.pattern
  aloha.concurrent.synchronization
  aloha.concurrent.util
  aloha.data
  aloha.dml
    Aggregation.java
    Cardinality.java
    Constants.java
    Count.java
    Diff.java
    DMM.java
    FullOuter.java
    LeftOuter.java
      LeftOuter
        IProjector<L extends IView<?>, T extends IValueObject>
        IRecordProjector<L extends IRecord<?, ?>, T extends IValueObject>
        anti(ImmutVOs<L>, F, ImmutVOs<R>, IProjector<L, T>, Anti) <F extends Enum<F> &
        anti(ImmutVOs<L>, K, ImmutVOs<R>, IRecordProjector<L, T>) <K extends Enum<K>
        findByName(ImmutableArray<RF>, LF) <LF extends Enum<LF> & IField, RF extends Er
        join(ImmutVOs<L>, F, ImmutVOs<R>, IEquiView<L, R, T>, IProjector<L, T>, Equi) <F
        join(ImmutVOs<L>, K, ImmutVOs<R>, IEqui<L, R, T>, IRecordProjector<L, T>, Equi)
        joinManyToOne(ImmutVOs<L>, LF, ImmutVOs<R>, IEqui<L, R, T>, IProjector<L, T>)
    ParallelAggregate.java
    ParallelJoin.java
```

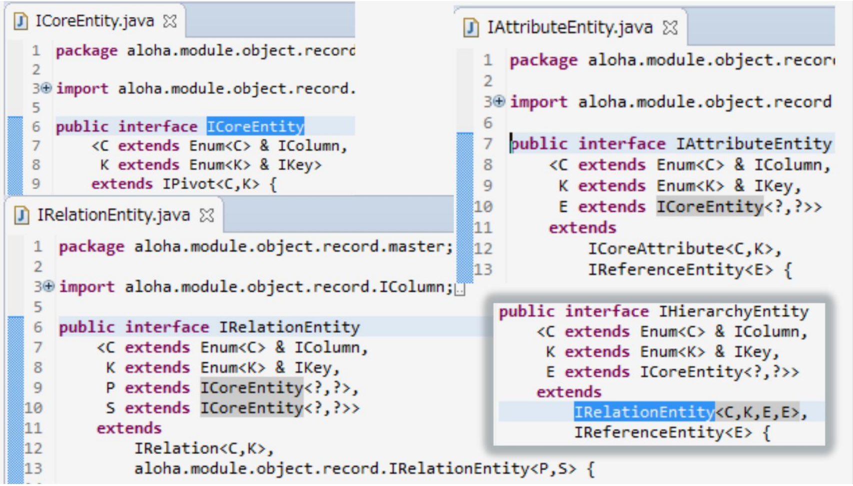
```java
    public static
    <LF extends Enum<LF> & IField,
     L extends IView<LF>,
     RF extends Enum<RF> & IField,
     R extends IView<RF>,
     T extends IValueObject>
    ImmutVOs<T>
    joinManyToOne(
        ImmutVOs<L> lefts,
        LF leftKey,
        ImmutVOs<R> rights,
        IEqui<L,R,T> equiProjector,
        IProjector<L,T> leftProjector) {

        return
            anti(
                lefts,
                leftKey,
                rights,
                leftProjector,
                Anti.ManyToOne).
            append(
                rights.
                    unlessEmpty(
                        (nonEmpty)->
                            equi(
                                nonEmpty,
                                findByName(
                                    nonEmpty.first().getFields(),
                                    leftKey),
                                lefts,
                                (R right, L left)->
                                    equiProjector.
                                        match(
                                            left,
                                            right),
                                Equi.OneToMany)));
    }
```

# Data Oriented Approach is the set of modeling building blocks



| | |
|---|---|
| Core | PK |
| Cancel *(Logical DELETE)* | PK/FK |
| Attribute *(Logical UPDATE)* | PK,FK |
| Relation | PK,FK1,FK2 |
| Fixed Attribute *(Link)* | PK/FK1*(,FK2)* |
| Transaction(Tx) | PK *+refer to M or T* |
| DerivedTx | PK/FK |
| TxAttribute *(No Timestamp)* | PK/FK |
| ChildTx *(No Timestamp)* | PK,FK |

*The same timestamp of referring Tx's one*



ICoreEntity.java

```java
package aloha.module.object.record

import aloha.module.object.record.

public interface ICoreEntity
    <C extends Enum<C> & IColumn,
     K extends Enum<K> & IKey>
    extends IPivot<C,K> {
```

IRelationEntity.java

```java
package aloha.module.object.record.master;

import aloha.module.object.record.IColumn;

public interface IRelationEntity
    <C extends Enum<C> & IColumn,
     K extends Enum<K> & IKey,
     P extends ICoreEntity<?,?>,
     S extends ICoreEntity<?,?>>
    extends
        IRelation<C,K>,
        aloha.module.object.record.IRelationEntity<P,S> {
```

IAttributeEntity.java

```java
package aloha.module.object.record

import aloha.module.object.record

public interface IAttributeEntity
    <C extends Enum<C> & IColumn,
     K extends Enum<K> & IKey,
     E extends ICoreEntity<?,?>>
    extends
        ICoreAttribute<C,K>,
        IReferenceEntity<E> {
```

```java
public interface IHierarchyEntity
    <C extends Enum<C> & IColumn,
     K extends Enum<K> & IKey,
     E extends ICoreEntity<?,?>>
    extends
        IRelationEntity<C,K,E,E>,
        IReferenceEntity<E> {
```

# BOS&DOA expose the concept of multilayered design

- **Business logic is implemented in Java using the core interfaces**

IConceptual - defines the data domain,

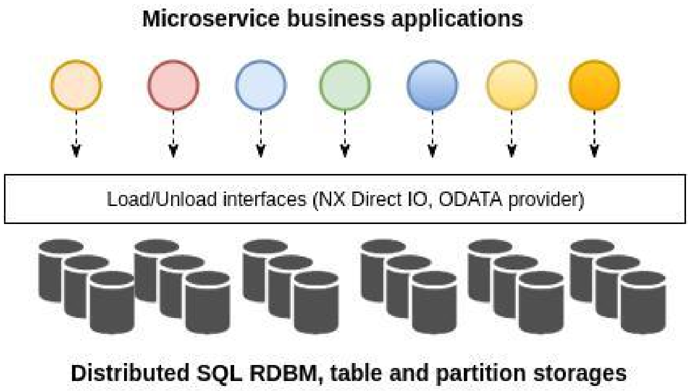
IGateWay - defines the IO and operation

IActor - define the data consumers and providers

IMDMlookUp - define the MDM

- **Business applications are small and deployed as maven artifacts without hard restart of service containers**

Hybrid OLAP\OLTP, data marts, real time lookup, and intelligent processing is processed in Java, using immutable objects and distributed closures (lambdas).


Microservice business applications
Load/Unload interfaces (NX Direct IO, ODATA provider)
Distributed SQL RDBM, table and partition storages

# 3. By the way, who is NoXA?

- Noxa is the technology enabler on eastern europe market.
- Noxa is the developer of proprietary software that makes BOS ecosystem better.
- Noxa is one of the major contributors in BOS ecosystem.

# Products and services

- **Noxa Data Management Console**

Graphical User Interface that gives the control of all aspects of infrastructure.

- **Noxa Direct IO**

The parallel load, unload utility and drivers.

- **Noxa ODATA integration**

- **Project supervision**

Consulting and PM services

- **R&D services**

We build and start brand new systems

- **Training**

# THANKS!

**Any questions?**

You can find me amergasov@noxa-datalab.com

www.noxa-datalab.com