DALi

A database abstraction layer for multi-user web applications

Johan Brichau, Andy Kellens

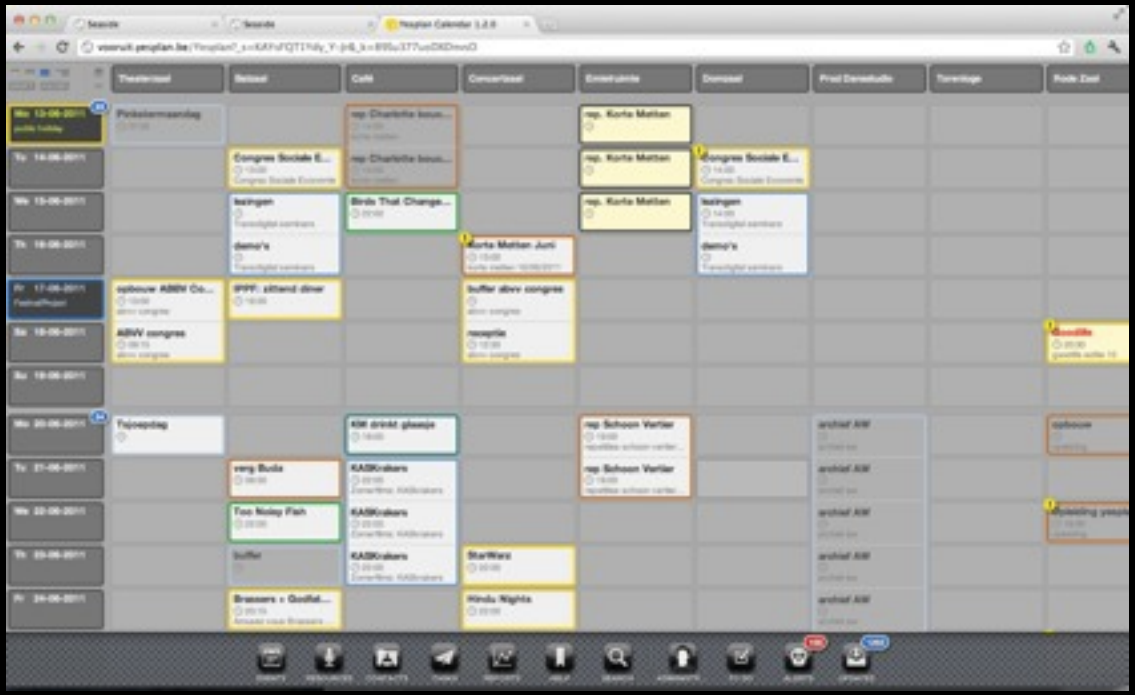
INCEPTIVE.BE
INNOVATIVE SOFTWARE CONCEPTS

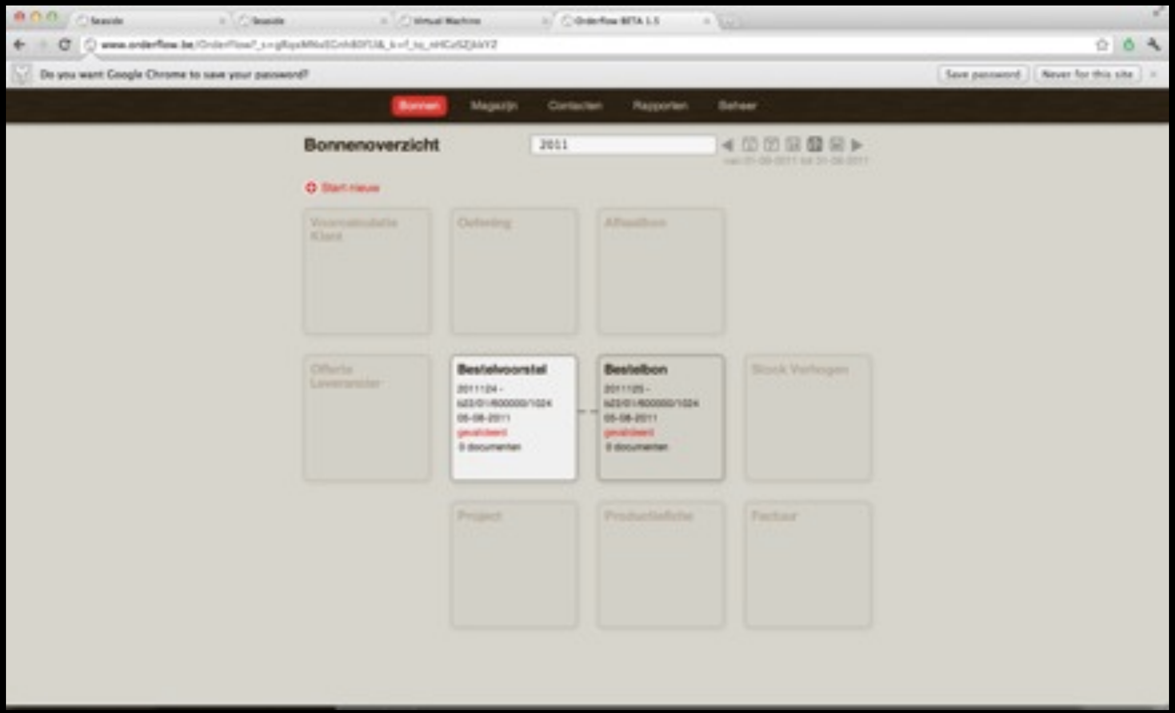
Yesplan
Let's make it happen

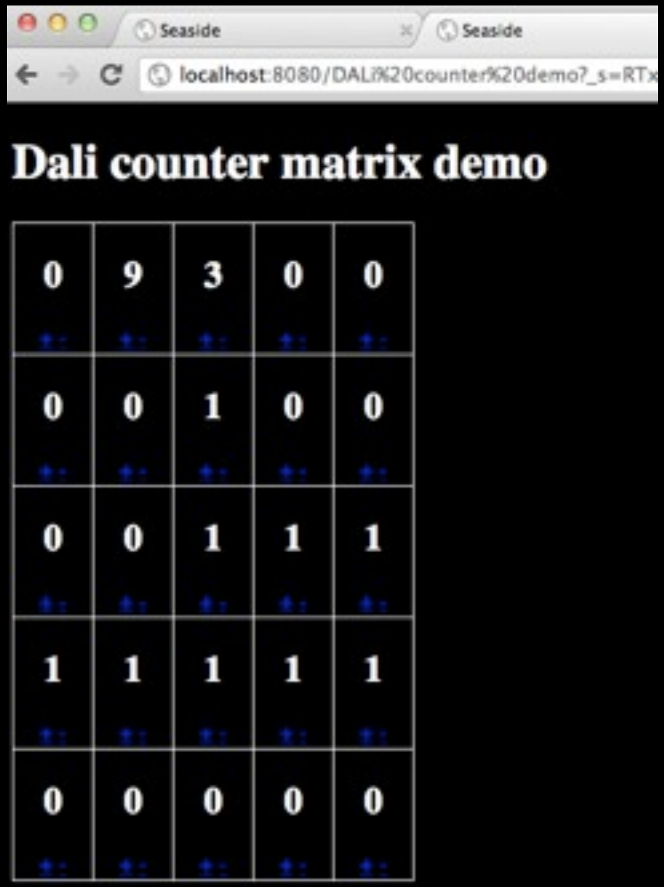
Goals of DALi

- **Independence of specific database platform**
  - Magma, GOODS
  - Gemstone (GLASS)
  - Image persistency
- **Focus on “Develop in Pharo, deploy in Gemstone”**
  - ... but don’t limit your options
- **Tx API for handling concurrency conflicts**
  - “Application-level” semantic concurrency conflicts
  - Parallel web request execution conflicts

Use cases: shared data editing





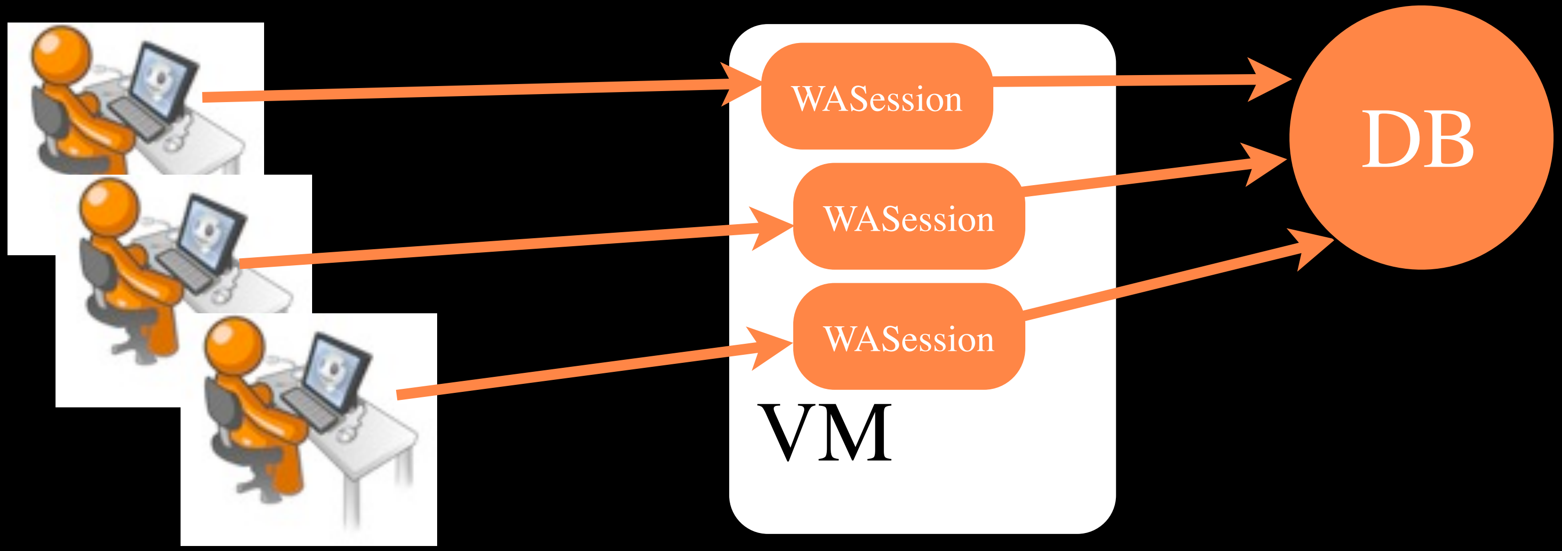


Dali counter matrix demo

0	9	3	0	0
0	0	1	0	0
0	0	1	1	1
1	1	1	1	1
0	0	0	0	0

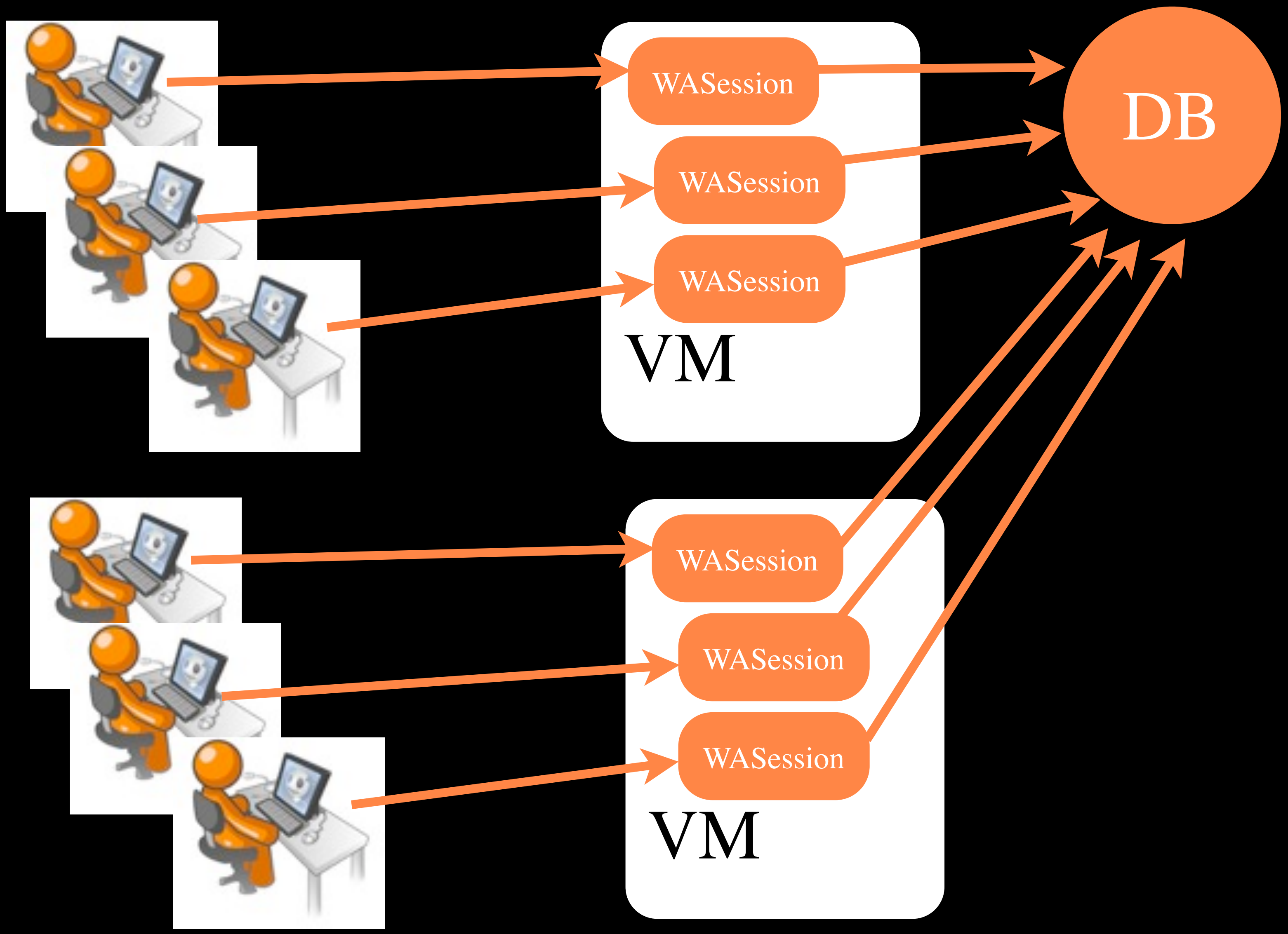
Transaction conflicts

- **GOODS / Magma / ...**
  - Keep a separate database view per web session
  - Concurrent web editing conflicts are database Tx conflicts



Transaction conflicts

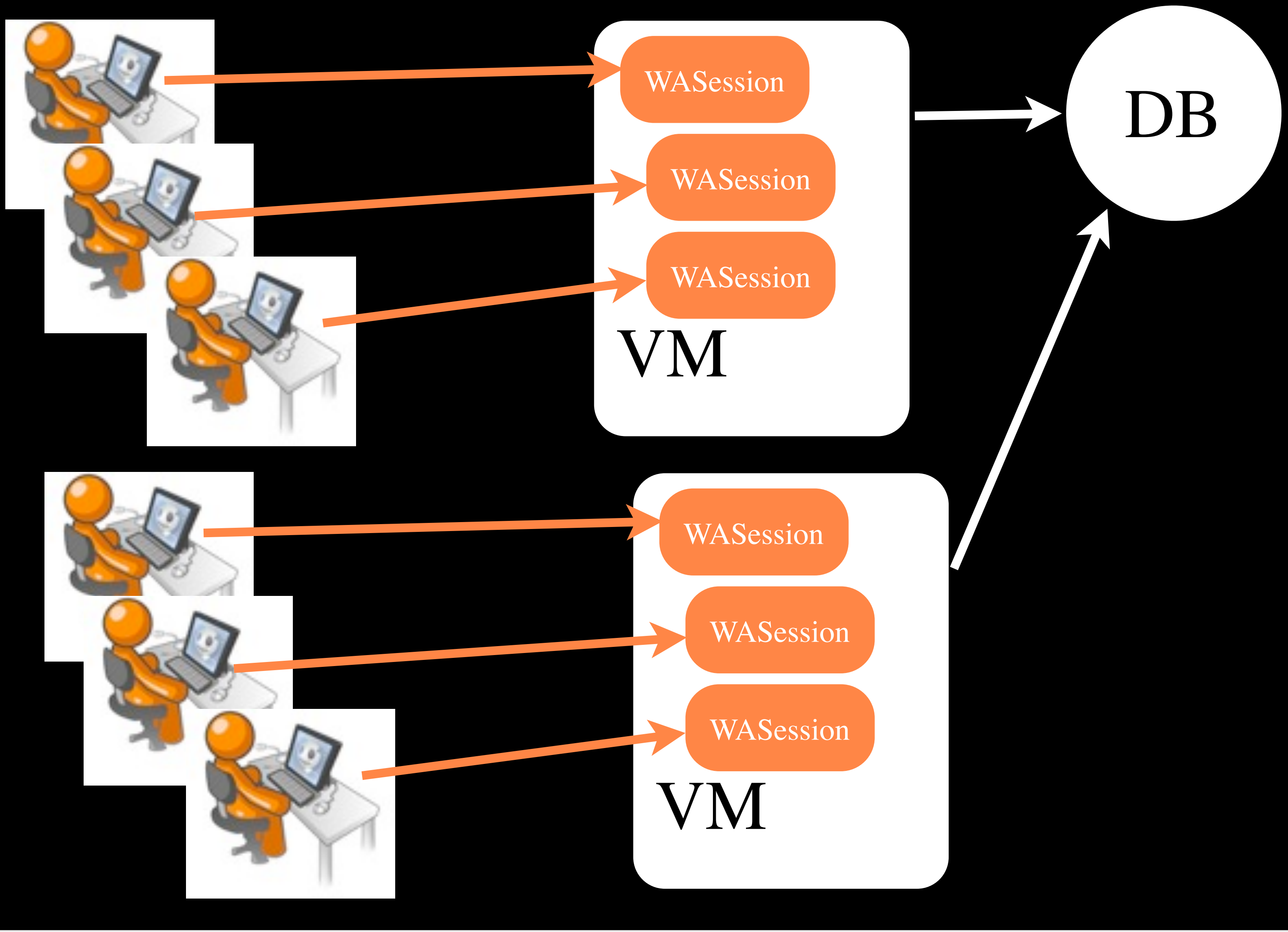
- **GOODS / Magma / ...**
  - Keep a separate database view per web session
  - Concurrent web editing conflicts are database Tx conflicts



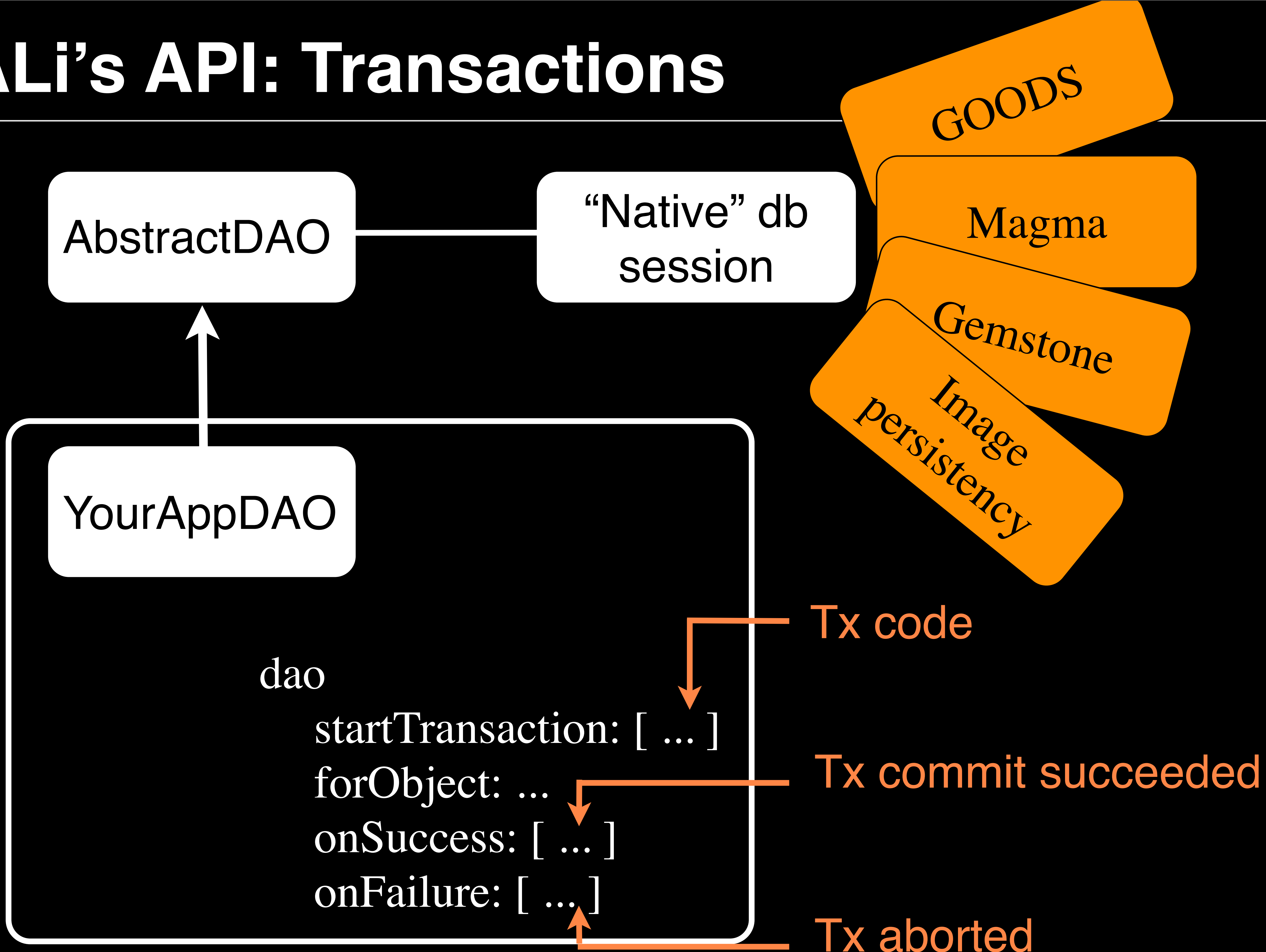
Transaction conflicts

GEMSTONE (GLASS)

- Keep a database view per VM (aka gem)
- Concurrent web editing conflicts are **not** database Tx conflicts

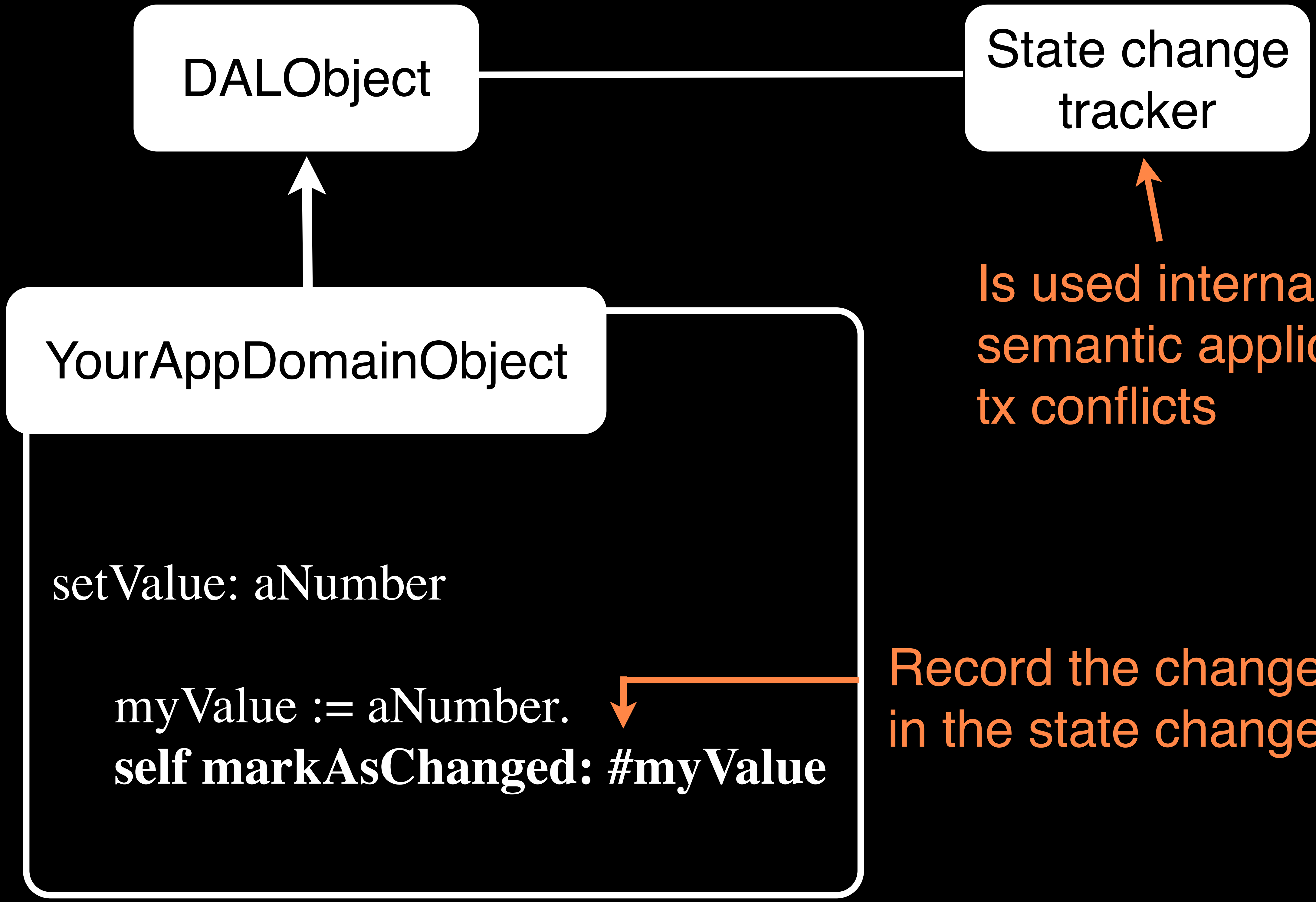


DALi's API: Transactions



```
dao
    startTransaction: [ ... ]
    forObject: ...
    onSuccess: [ ... ]
    onFailure: [ ... ]
```

DALi’s API: change tracking



```
setValue: aNumber

    myValue := aNumber.
    self markAsChanged: #myValue
```

Is used internally to detect semantic application-level tx conflicts

Record the changed attribute in the state change tracker

Why change tracking?

Essential in GLASS

- Only means to detect conflicts between different Seaside sessions
- Integrates with GLASS' 1 commit per web request
  - DALi's Tx blocks prevent conflicts driven by the state tracker
  - Gemstone Tx block prevent actual db conflicts
  - Both do an abort and retry the request
  - DALi's retried request will execute the onFailure blocks

Finer-grained tx conflicts in GOODS

- Attribute-level instead of object-level conflicts

Some more details

Identification & change tracking

- Unique ids for each DALObject
- Versionstamps for each DALObject

DALi-GOODS

- Nesting of transactions
- Manual write barrier
- Tracked collections
- Proper unique ids

DALi-GEMSTONE

- Nesting of transactions
- Integrated with Seaside GLASS
- Use of oops for unique ids

Future

- **Support for communicating state changes between sessions**
  - We have change tracking already for conflict detection
  - We could send out notifications on db updates on commit/abort
- **Update backends**
  - Magma backend has not been maintained
  - Need to try an RDB backend

Get it...

Want to give it a try?

http://www.squeaksource.com/DALi
http://ss3.gemstone.com/DALi

License: MIT