HighQSoft Query Language

Technical Overview

Andreas Hofmann
Manager Application Development

andreas.hofmann@highqsoft.de

Content

- Motivation to start HQL development
- Requirements for HQL
- HQL Architecture
- HQL Implementation
- Visions & Ideas

Motivation for HQL Development

Usage of Business Layer Entities

Business Layer
used entities: Test, Vehicle, User, …

ASAM ODS API
generic and free of business logic entities

Datamodel
defines the entities for the business logic: Test, Vehicle, User, …

Motivation for HQL Development

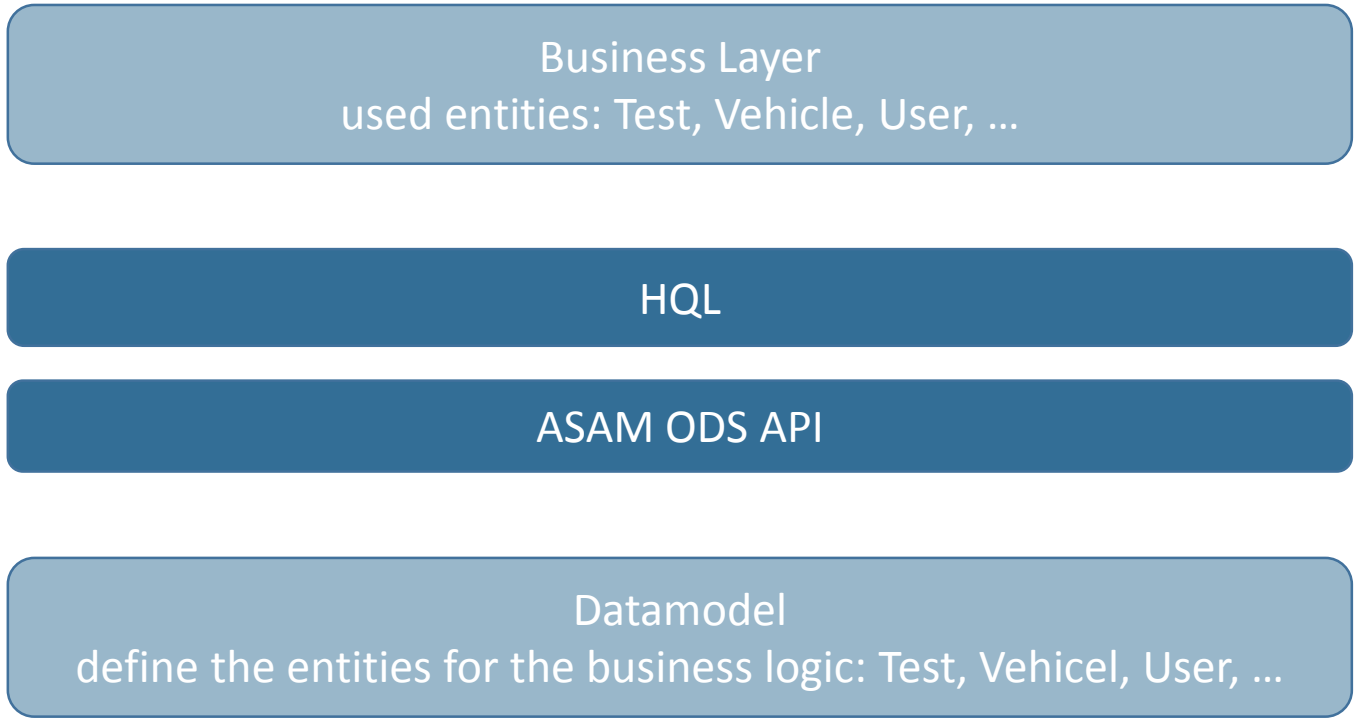
ASAM ODS Data Access

- Defining the ASAM ODS Query Structure is very complex
  - complex structure hierarchy
  - ODS know-how is required
  - error prone
- Possible wrong usage of the ASAM ODS API
  - performance issues by to many network calls
  - problems using multiple transactions
- Encapsulated queries are not allowed
- Difficult to use ASAM ODS by configuration

→ One heavy weight component that hides the ASAM ODS complexity for the business logic
  - difficult to extend
  - difficult to maintain

Motivation for HQL Development

HQL as additional abstract layer to decouple business from ODS layer



Content

- Motivation to start HQL development
- Requirements for HQL
- HQL Architecture
- HQL Implementation
- Visions & Ideas

HQL Requirements

How to hide disadvantages

- Easy to use
- Usable for different ODS models
- Understands ODS base model
- Understands entities from business layer
- Object oriented API without ASAM ODS dependencies
  - but allows to be bypassed (e.g. get T_LONGLONG for other component call)
- Covers ODS API as complete as possible
- Enable encapsulated queries
- Enhanced transaction handling
- Usable by configuration
- Must be fast

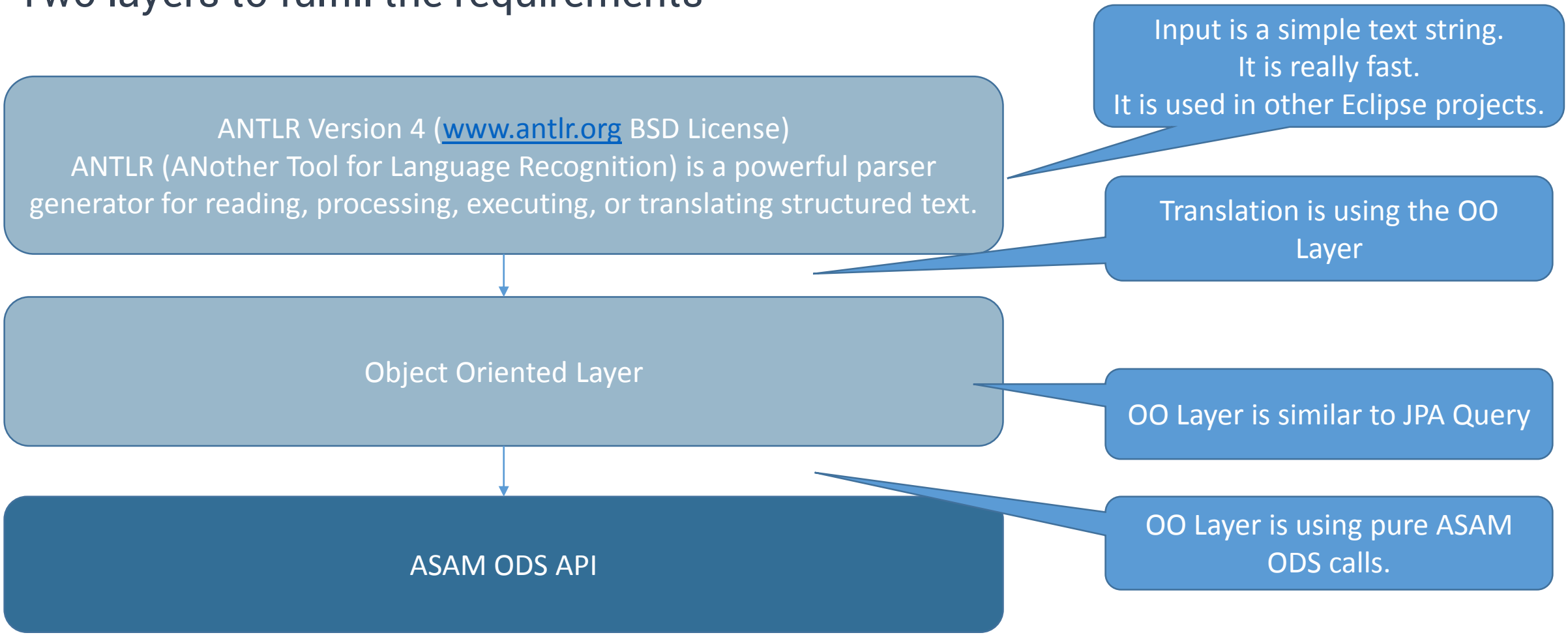
→ HQL as domain specific language to bridge the ASAM ODS layer

Content

- Motivation to start HQL development
- Requirement for HQL
- HQL Architecture
- HQL Implementation
- Visions & Ideas

HQL Architecture

Two layers to fulfill the requirements

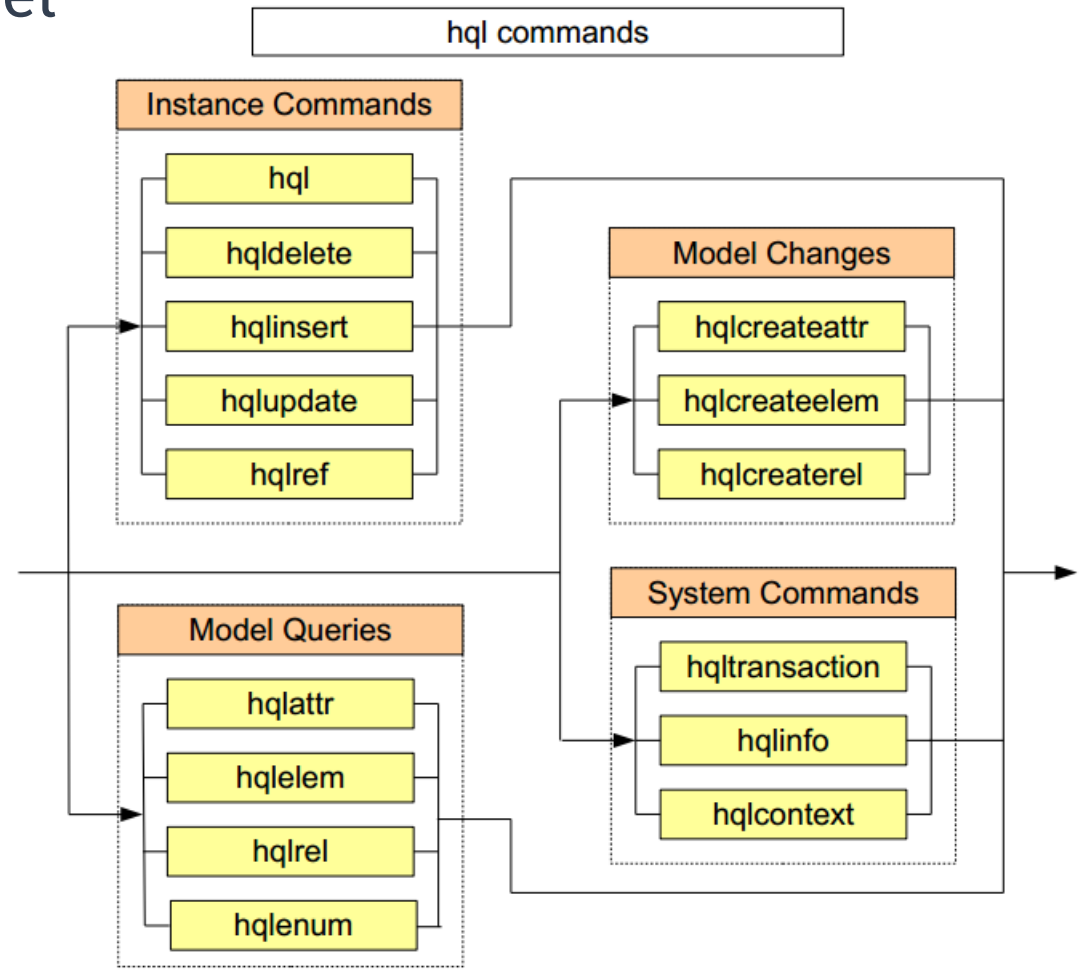


Content

- Motivation to start HQL development
- Requirement for HQL
- HQL Architecture
- HQL Implementation
- Visions & Ideas

HQL Implementation

HQL Command Set



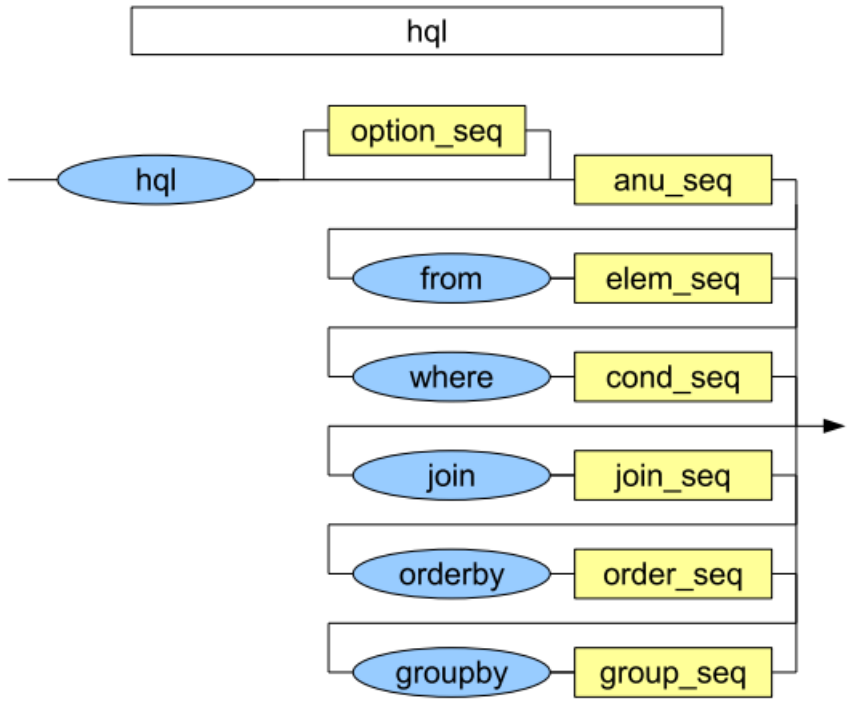
HQL Implementation

Example: Query ASAM ODS data

```
hql Id,
    User.Name,
    UserGroup.Name
  where Name like ('*')
  join User to UserGroup
  orderby -Name
```

or using the base model

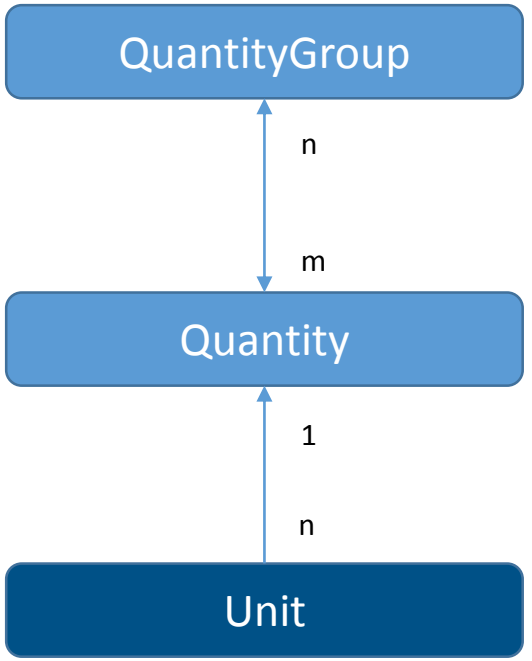
```
hql aouser.id,
    aouser.name,
    aousergroup.name
  where name like ('*')
  join aouser to aousergroup
  orderby -name
```



HQL Implementation

Example: Encapsulated Query

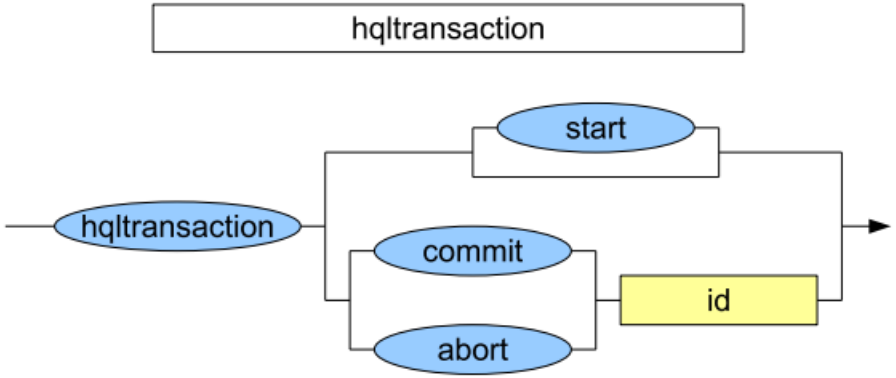
```
hql Id
   from QuantityGroup
   where Quantity.Id inset(
      {hql Id
         from Quantity
         where Unit.name = "K"})
```



HQL Implementation

Example: Transaction Handling

```
hqltransaction start
...
hqltransaction [transaction=4711] commit
  or
hqltransaction [transaction=4711] abort
```

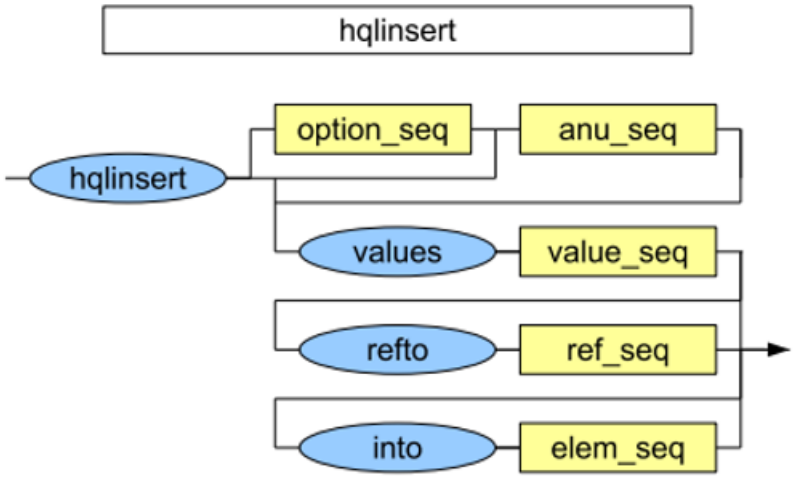


HQL Implementation

Example: Creating Instances

```
hqlinsert values (name=kg, factor=1, offset=0)
          into aounit;

hqlinsert [transaction=3]
          values (name=kg, factor=1, offset=0)
          into aounit;
```



HQL Implementation

Example: Using the OO API

```
// first we need a builder.
HQLStatementBuilder builder = hql.prepareStatement(HQLStatmentBuilder.class);

// define the command and the columns to be returned.
HQLStatement statement = builder.hqlInsert(build.column("id"));

// add the values.
HQLAttribute attr = builder.attribute(“name”, true)
statement = statement.values(builder.value(attr, “The new Testname”));

// define the destination element.
statement = statement.into(builder.element("Test", false));

// Getting the result.
Query query = hql.createQuery(statement);
QueryResult result = query.execute();
```

HQL Implementation
Example: Table structure as result

- The result of a call is always a table-like structure.
  - table has rows and columns
  - columns and rows have cells
- The result is free of ASAM ODS definitions.
- ASAM ODS definitions can be retrieved by using generic types.
  - Example: `T_LONGLONG id = cell.get(T_LONGLONG.class);`

Content

- Motivation to start HQL development
- Requirement for HQL
- HQL Architecture
- HQL Implementation
- Visions & Ideas

HQL Visions and Ideas

Enhancement is possible and wanted

- Plugins for specific aggregate functions
  - `Hql asampath(id) from Measurement where …`
- HQL access to ValueMatrix is not implemented yet.
- JPA Query on top of HQL

Conclusion
HQL to relieve developers

- HQL is easy to use, either in full-text mode as well as in OO mode.
- HQL hides complex ASAM ODS API
- HQL enhances ASAM ODS API
  - transaction handling
  - encapsulated queries
- HQL uses ANTLR that is accepted by the Eclipse rules.
- HQL is used in various projects
  - one reference partner is Canoo AG

Thank you very much !