hybris software

An SAP Company

hybris software
An SAP Company

hybris Developer Training Part I - Core Platform

Flexible Search

Overview

Syntax

API examples

Overview

- SQL-like syntax
- Abstracts a database query into a hybris Item query
- Returns a list of objects (hybris Items)
- Makes properties easily queryable
- Is translated into native SQL statements on execution
- Allows nearly every feature of SQL SELECT statements
- Queries go through cache

Overview

Syntax

API examples

Syntax

- Basic Syntax:

```
SELECT <selects> FROM {types} (where <conditions>)
   ?(ORDER BY <order>)?
```

- Mandatory:
  - `SELECT  <selects>`
  - `FROM {types}`
- Optional:
  - `where <conditions>`
  - `ORDER BY <order>`
- SQL Command / Keywords:
  - ASC, DESC, DISTINCT, AND, OR, LIKE, LEFT JOIN, CONCAT, ..

Query examples

- Simple queries:

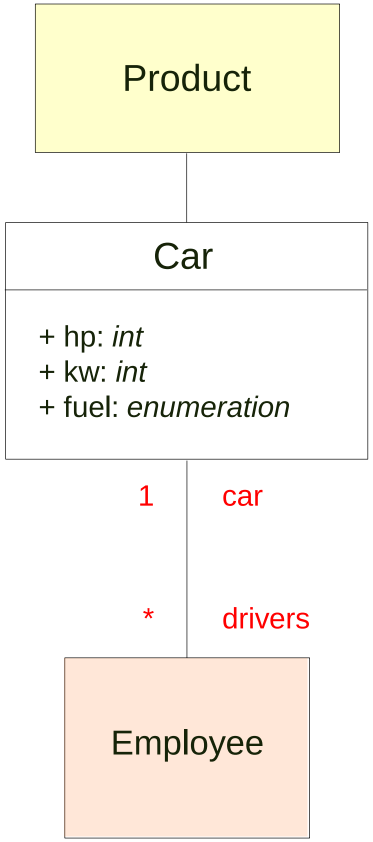
```
SELECT {code},{hp} FROM {Car}
```

- Single type queries:

```
SELECT {code} FROM {Product!}
```

- Joins:

```
 SELECT {c.code},{e.uid} FROM {
    Car as c JOIN Employee as e
               ON {c.pk} = {e.car}
          } WHERE {e.uid} LIKE ‘%Szybki’
```



More query examples

- Inner queries:

```
SELECT {c.code} FROM {Car as c}
  WHERE {c.mechanic} IN
    ({{
                 SELECT {pk} FROM {Employee}
        WHERE {uid} LIKE ‘%Tesla’
            }})
```

- Parametrized queries:

```
SELECT count(*) FROM {Car}
  WHERE {hp} > ?hpMin
  AND   {hp} < ?hpMax
```

hybris software
An SAP Company

Overview

Syntax

API examples

Querying with parameters

```
String fsq = "SELECT {PK} FROM {Car} WHERE {mechanic} =?mechanic”;
FlexibleSearchQuery query =
   new FlexibleSearchQuery(fsq,
      Collections.singletonMap( "mechanic", mechanic ) );
SearchResult<CarModel> result =
   flexibleSearchService.search( query );
List<CarModel> cars = result.getResult();
```

Querying for other types than Models

```
String fsq = "SELECT COUNT( {PK} ) FROM {Car}”;
FlexibleSearchQuery query = new FlexibleSearchQuery( fsq );
query.setResultClassList( Arrays.asList( Integer.class ) );
SearchResult<Integer> result =
      flexibleSearchService.search( query );
List<Integer> carsCount = result.getResult();
```

Pagination

```
public List<CarModel> getCars(int start, int range)
{
    String fsq= "SELECT {PK} FROM {Car}";

    FlexibleSearchQuery query = new FlexibleSearchQuery( fsq );

    query.setNeedTotal( true );

    query.setCount( range );

    query.setStart( start );

    return  flexibleSearchService.<CarModel>search( query
).getResult();
}
```

Performance gain only if underlying DB supports pagination

GenericSearch

- Similar to *HibernateCriteriaSearches*
- Search for items as well as raw data fields
- Unlimited number of conditions
- Inner joins and outer joins between item types possible
- Unlimited number of “order by” clauses
- Sub-selects

GenericSearch Example

```
GenericQuery query = new GenericQuery(CarModel._TYPECODE);
GenericSerchField carField = new
   GenericSearchField( CarModel.PK, CarModel.Name );
GenericCondition condition =
   GenericCondition.createConditionForValueComparison(carField,
                                                      Operator.LIKE,
                                                      “BMW”);
query.addCondition( condition );
query.addOrderBy(new GenericSearchOrderBy( carField, true ));

List<CarModel> cars = genericSearchService.search( query );
```