The JORM Meta information

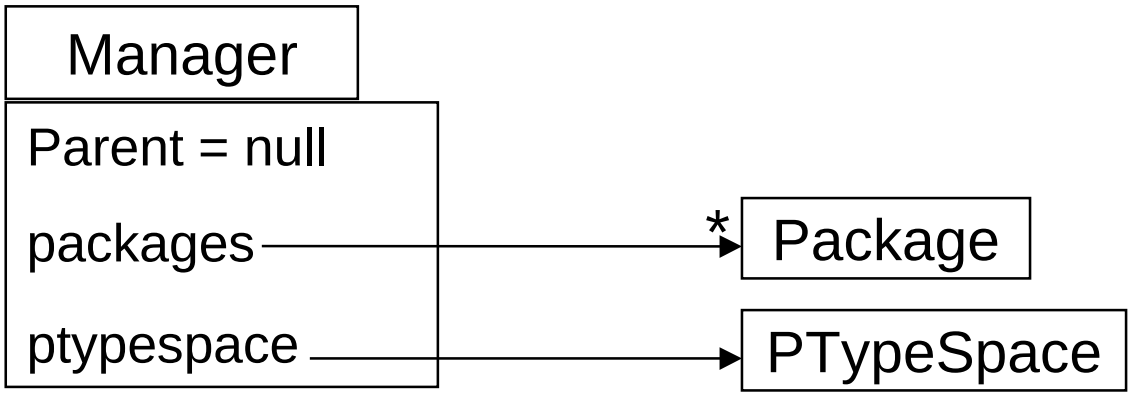
S. Chassande-Barrioz

Plan

- Generic part
  - Manager
  - CompositeName
  - Class
  - Fields
  - NameDef
- Mapping
  - Overview
  - Class mapping
  - Generic class mapping
  - Fields mapping
- Example
  - One-one relation

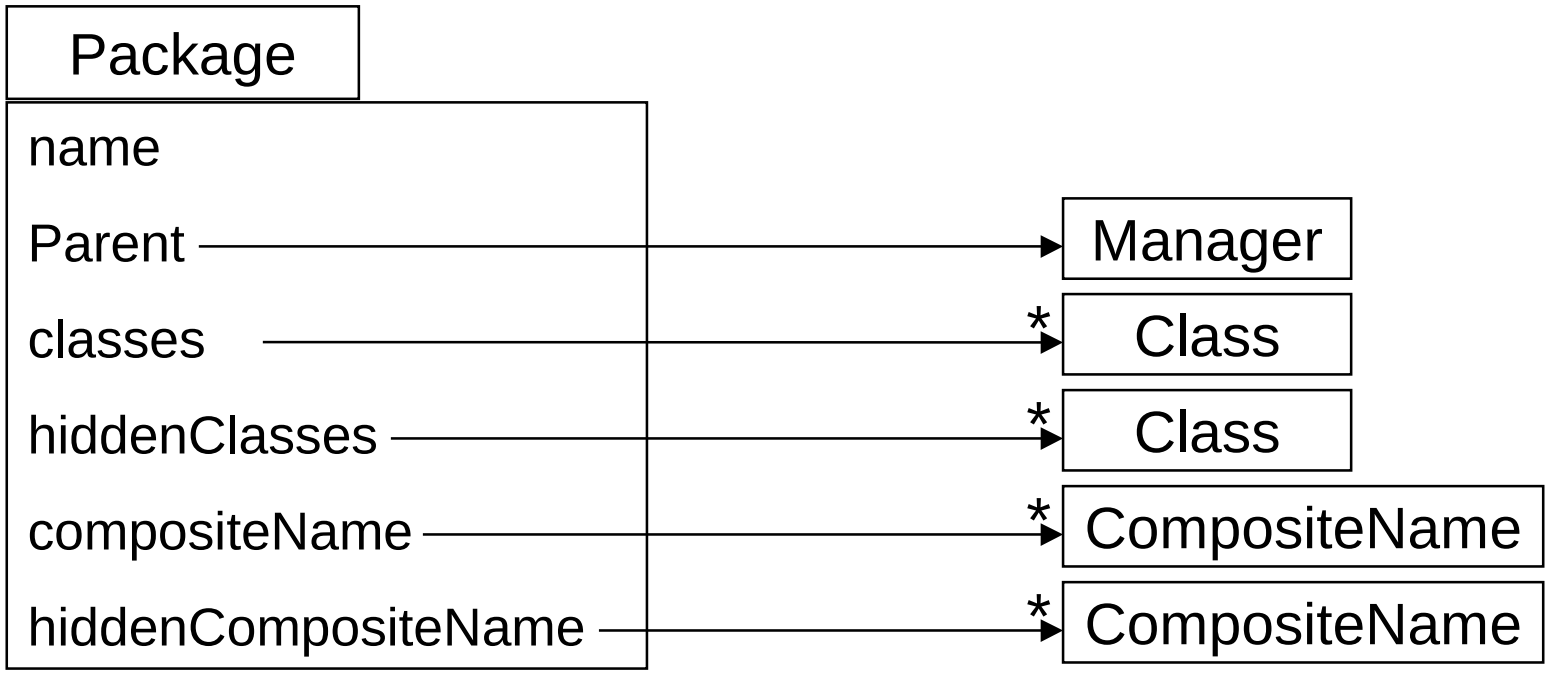
The Generic part

Manager



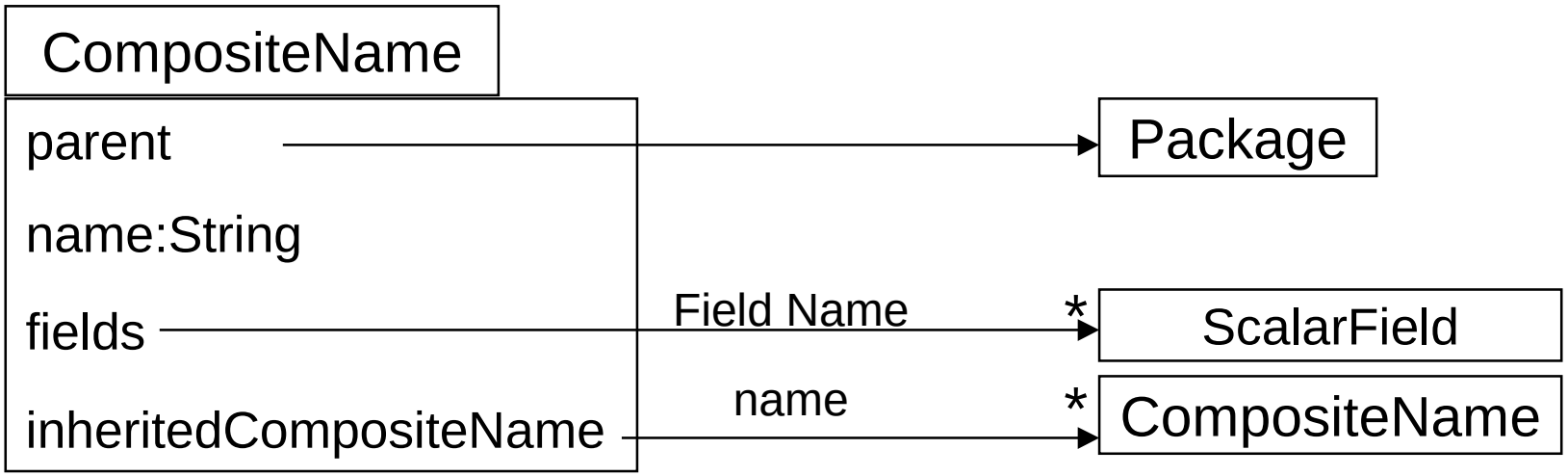
- A Manager is the hosting structure of the JORM meta information.
- It references the manager of the JORM type (PType).

Package



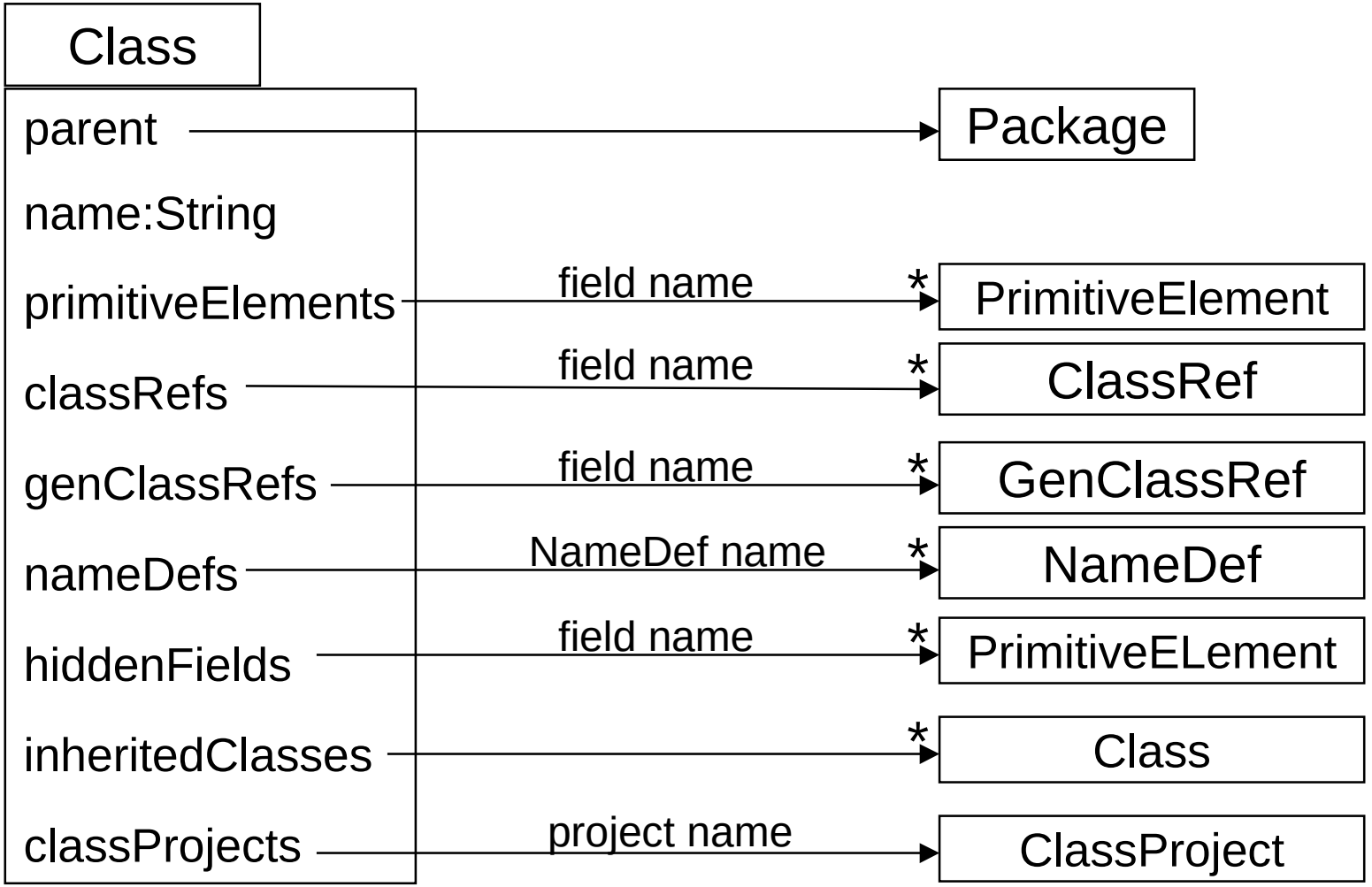
- A package represents a Java package.
- It contains Class definitions or CompositeName definitions.
- Each inner definition may be partially defined (only its name) depending on the scheduling of the meta information building.

CompositeName

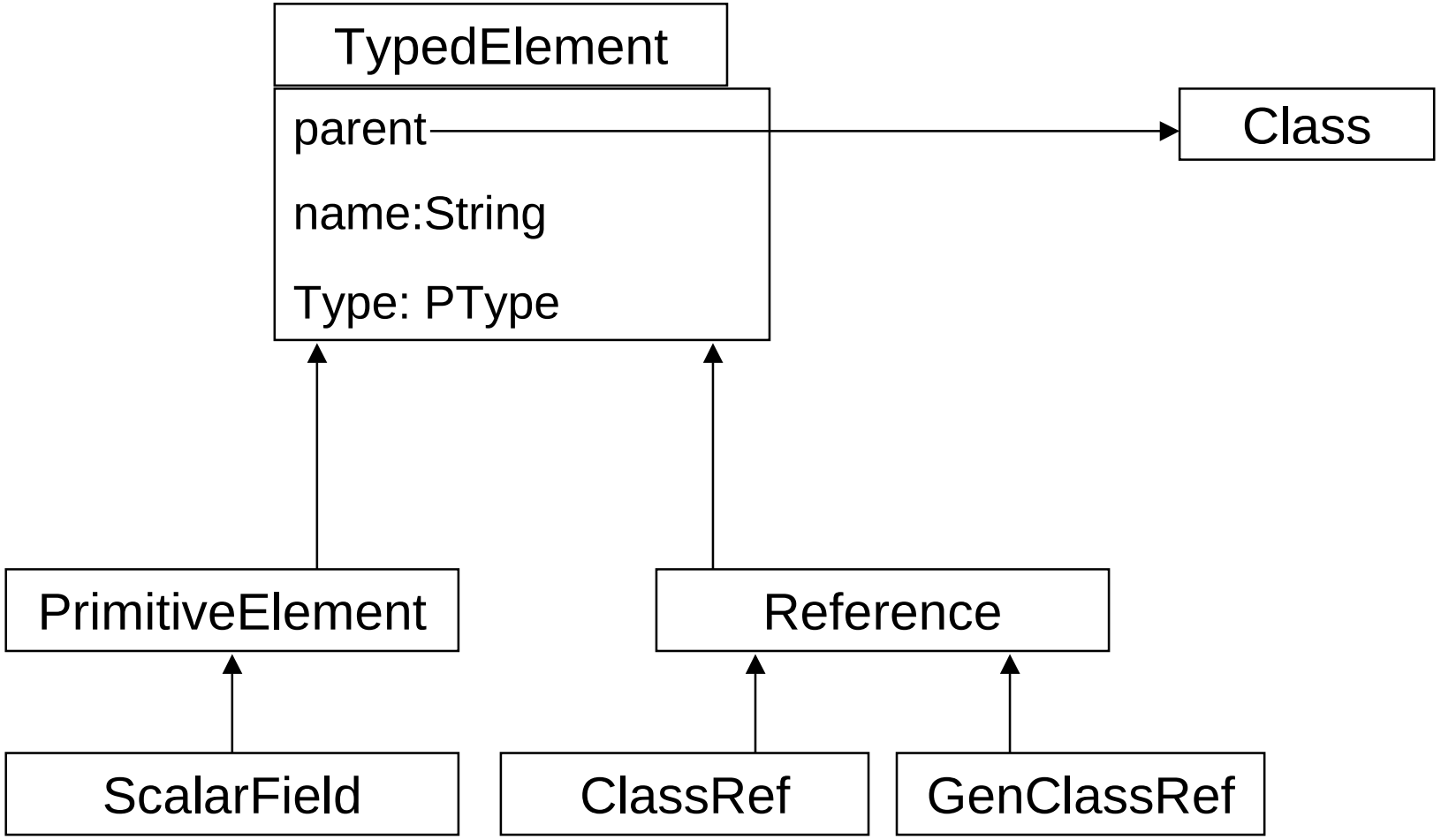


- A CompositeName defines an identifier of a persistent object.
- This identifier contains one or several fields.
- It is usually used for each primary key defined by a user.

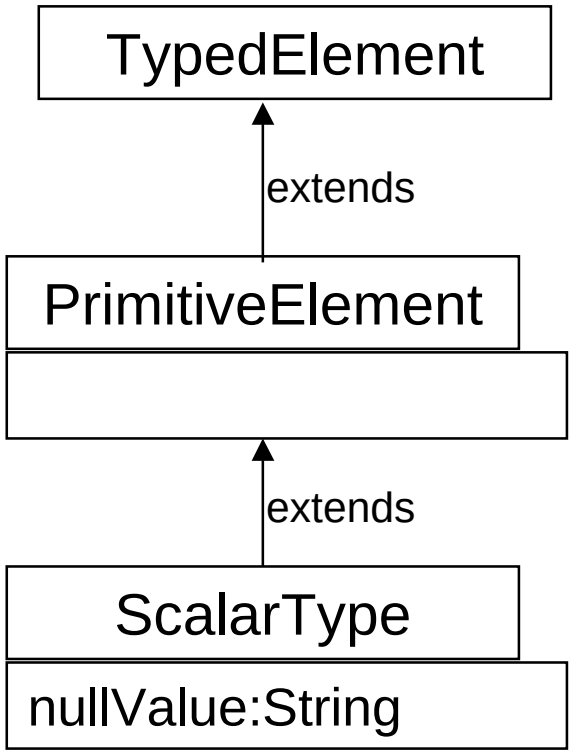
Class


Class
parent
Package
name:String
primitiveElements
field name
*
PrimitiveElement
classRefs
field name
*
ClassRef
genClassRefs
field name
*
GenClassRef
nameDefs
NameDef name
*
NameDef
hiddenFields
field name
*
PrimitiveELement
inheritedClasses
*
Class
classProjects
project name
ClassProject

Fields: Overview

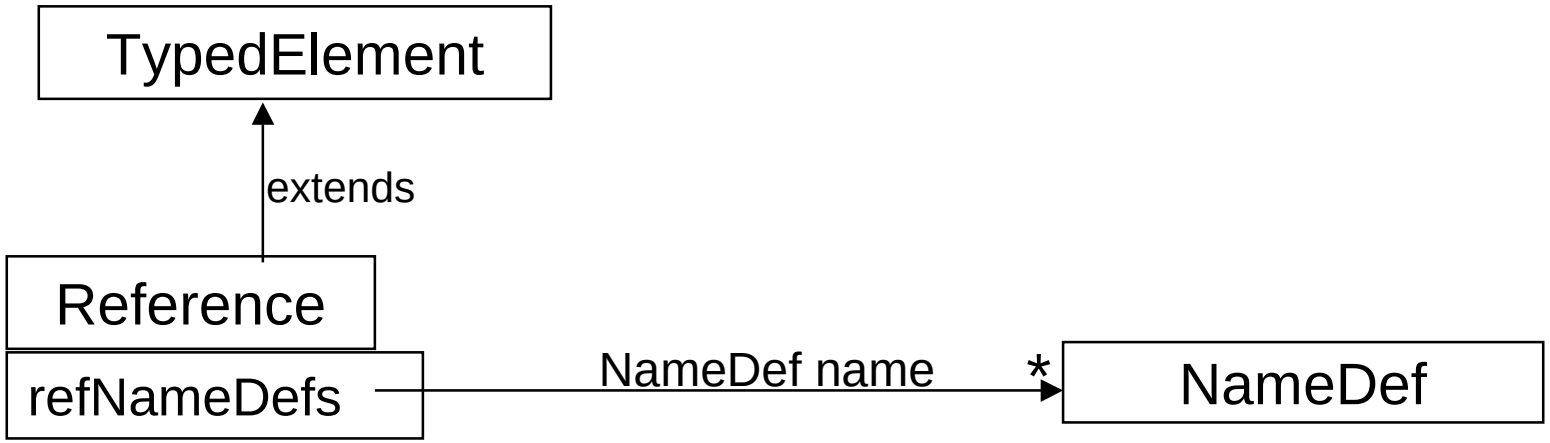


Fields: Primitive fields



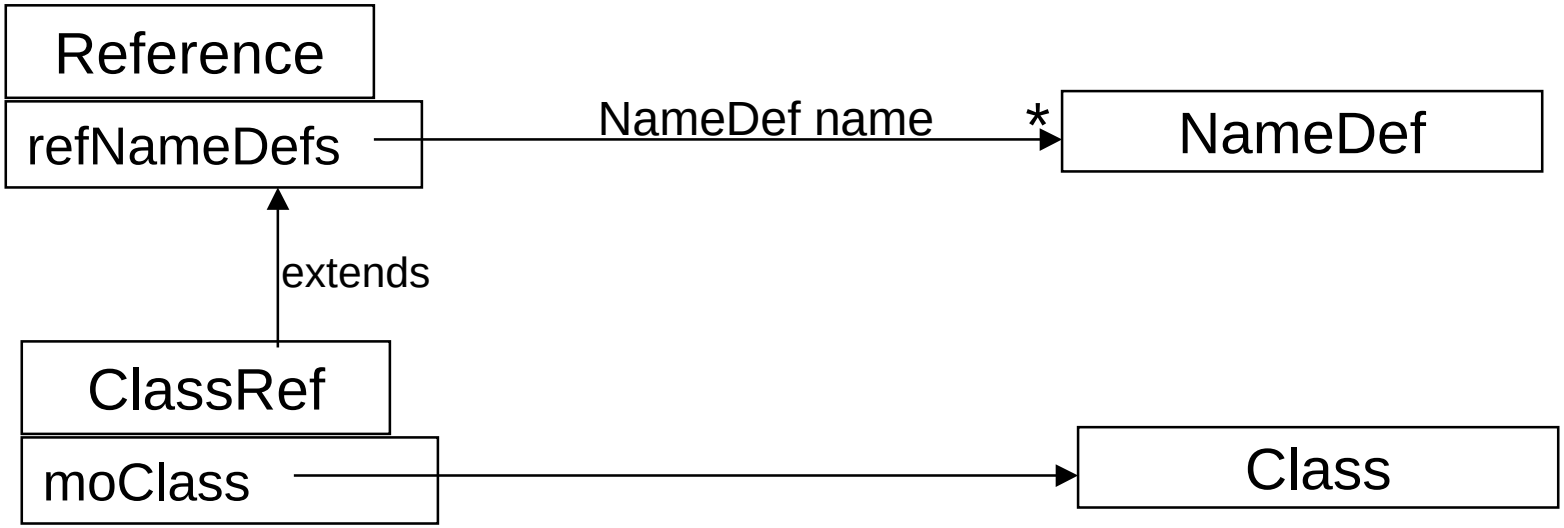
- A PrimitiveElement represents a field of a primitive type.
- A ScalarType represents a field which composes a NameDef.
  - The type of a ScalarField must be one of:
    - java.lang.Byte, byte
    - java.lang.Character, char
    - java.lang.Short, short
    - java.lang.Integer, int
    - java.lang.Long, long
    - java.lang.Short, short
    - java.lang.String
    - java.util.Date
    - byte[], char[]

Fields: Reference

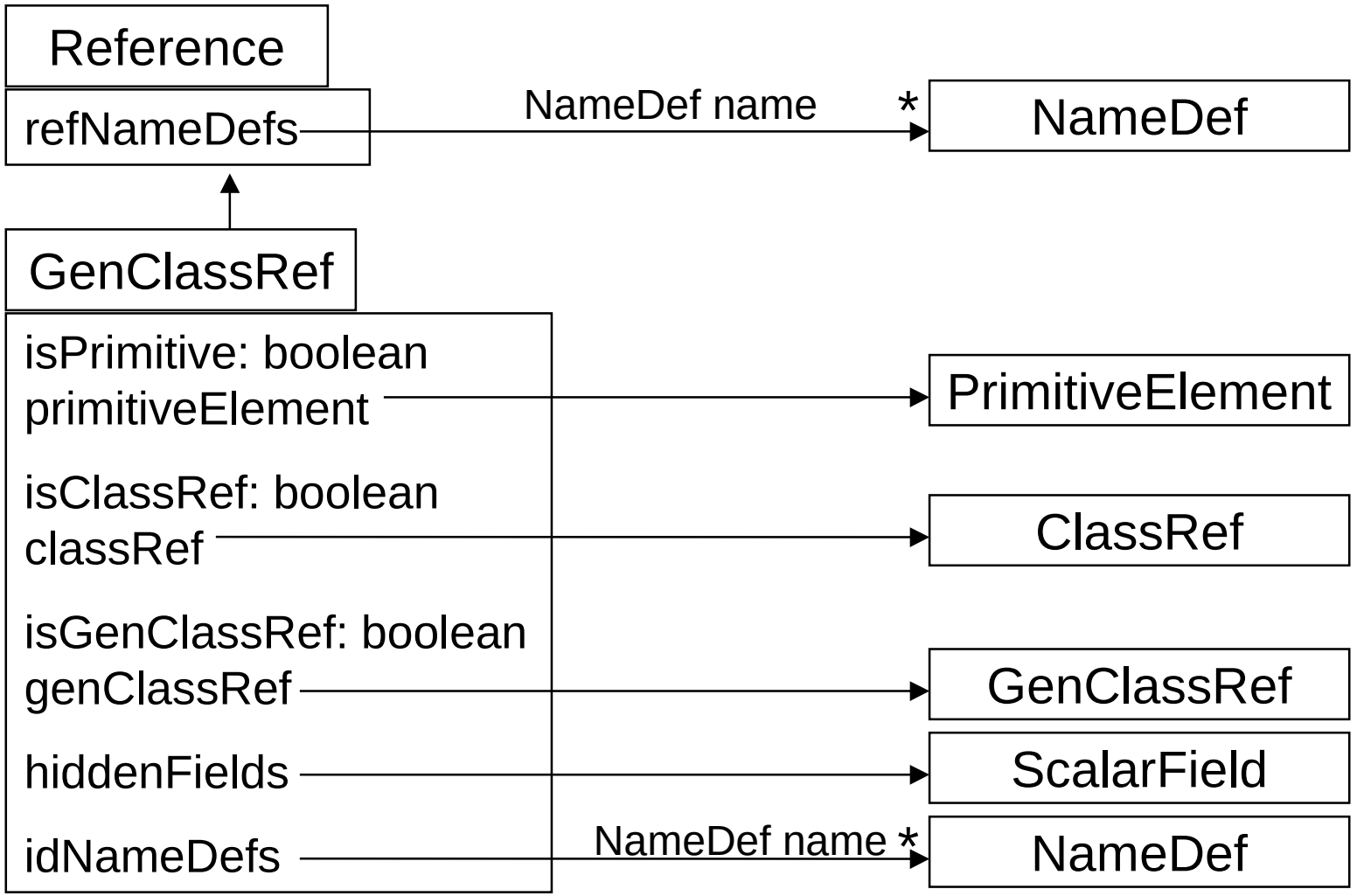


- The reference to a persistent name is described by a NamedDef.

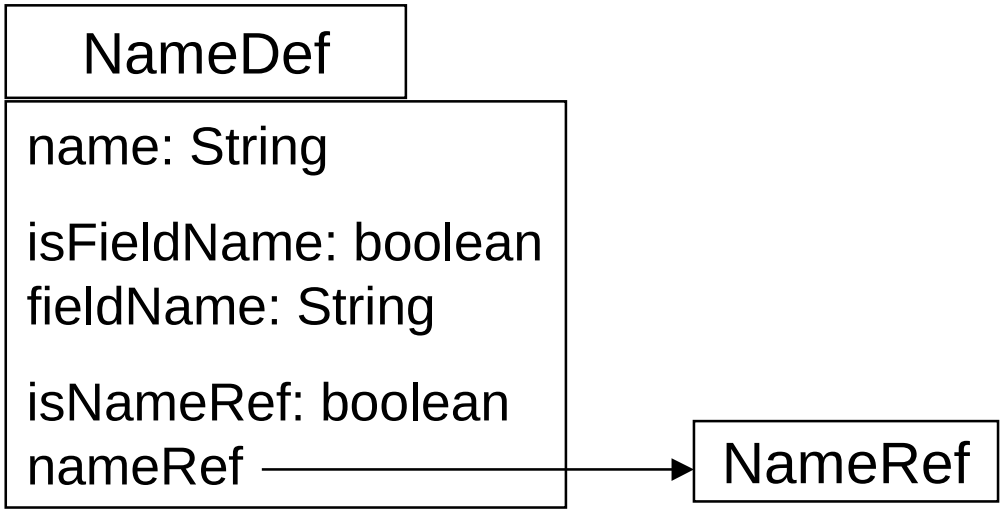
Fields: ClassRef



Fields: GenClassRef

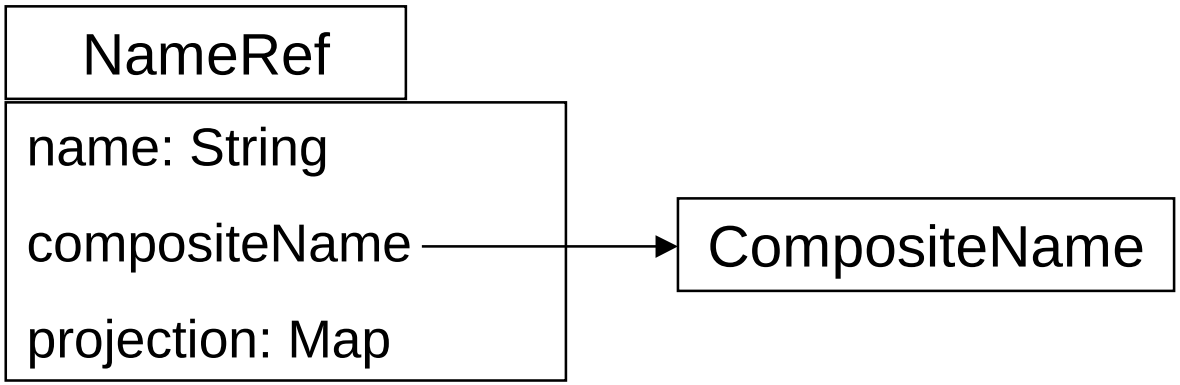


NameDef



- A NameDef is a name definition.
- It is used to *reference* or to *identify* a persistent object.
- There are two cases for the name composition
  - Single field: the NameDef indicates the Field name
  - Composite name:
    - It is defined externally in a CompositeName meta object.
    - A NameRef object describes the mapping of the composite name for the class (see next slide).

NameRef



- A NameRef defines the mapping between the fields of a CompositeName and the fields of the Class.
  - A key in the projection Map is the composite field name
  - The value is the class field name onto which the composite field is mapped.

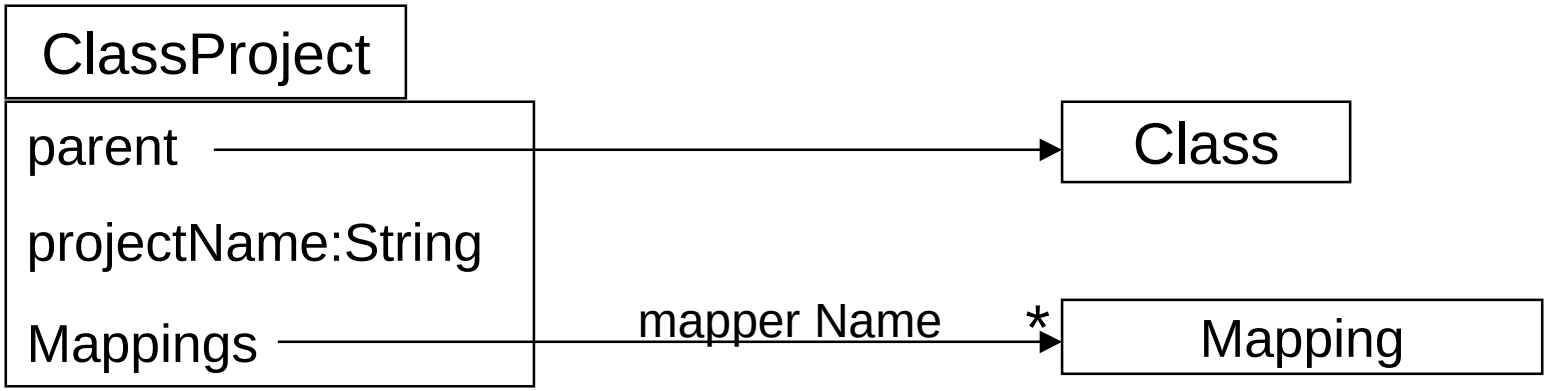
Mapping

Mapping: principles

- In addition to the definition of the generic structure, it is necessary to define the mapping of some generic elements
  - Class
  - PrimitiveElement
  - GenClassRef
  - …
- The mapping definitions are grouped by Class

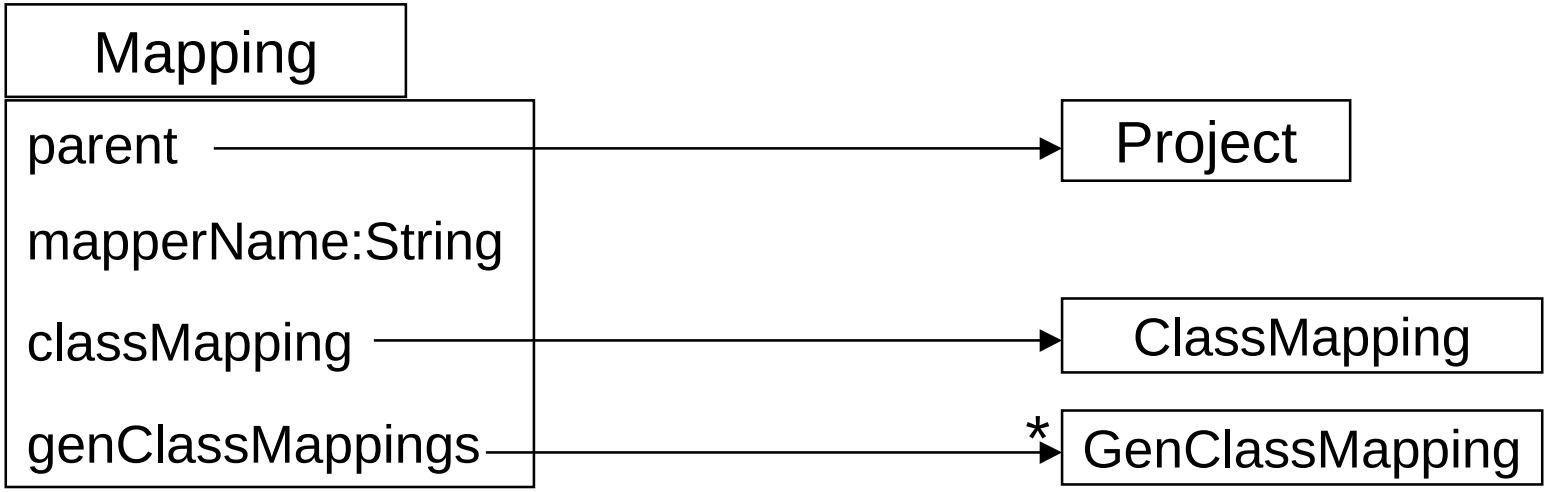
ClassProject

- Reminder: A Class references several ClassProject by their name (project name).



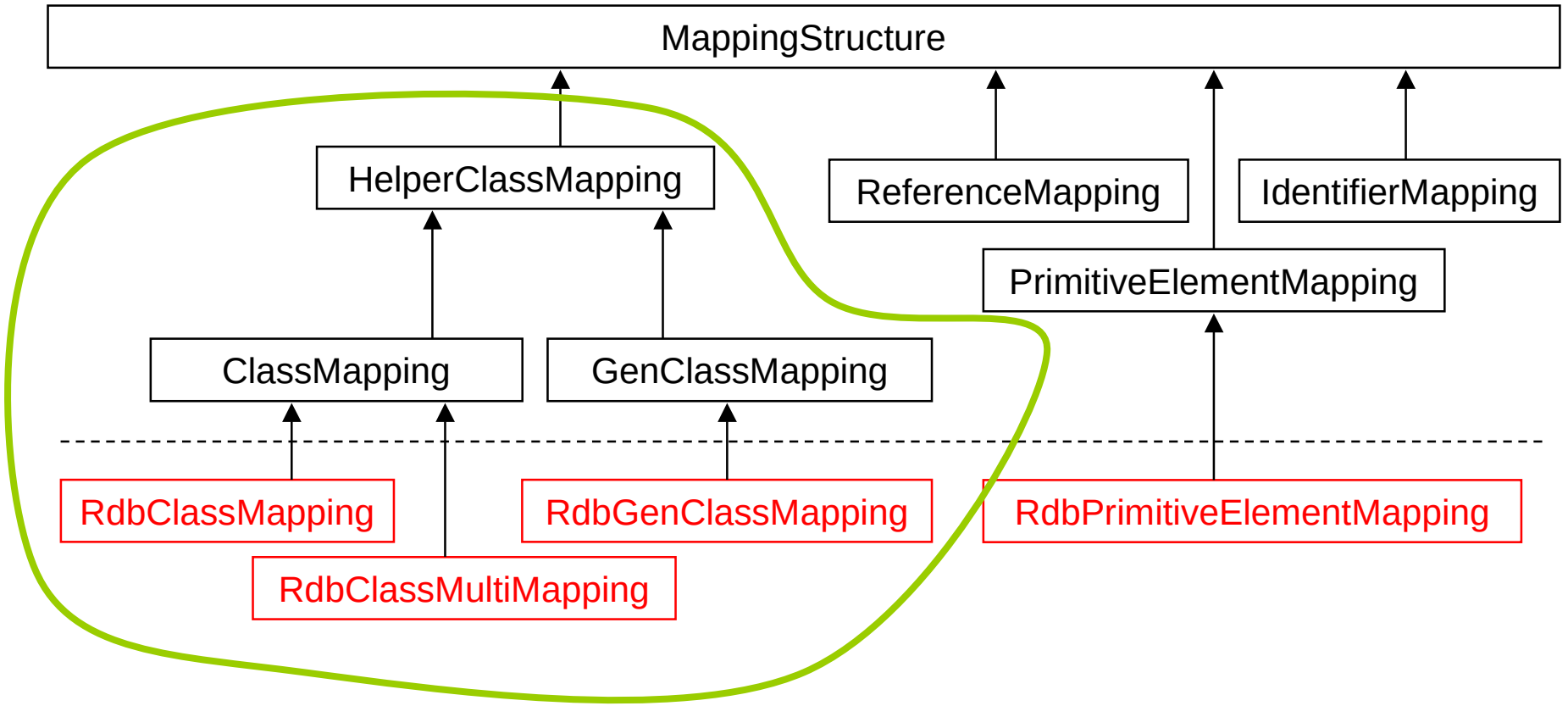
- A ClassProject represents a project for a given class.
- A ClassProject may contain mappings to several mappers (rdb.postgres, rdb.mysql, fos, …).

Mapping

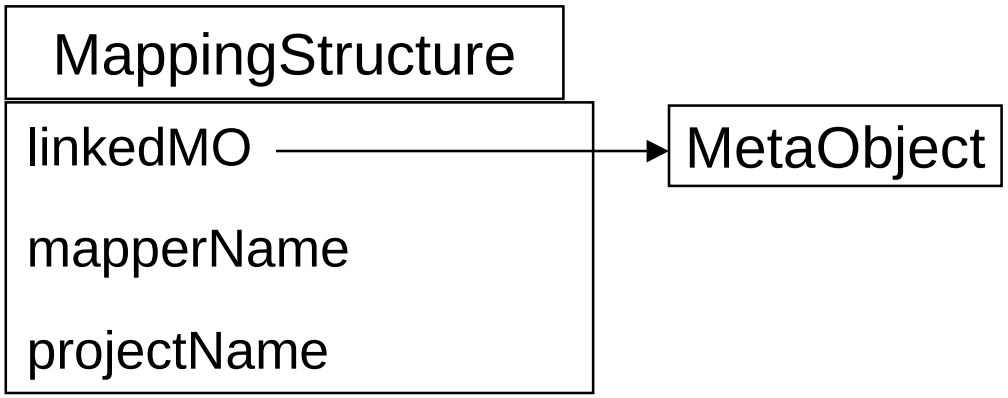


- A Mapping meta object represents the mapping rules (on a mapper) for a persistent class.
- It defines
  - the ClassMapping
  - the mapping of all generic classes referenced (directly or not) by the class.
- The mapper name is an attribute of a Mapping MetaObject.

Mapping: overview

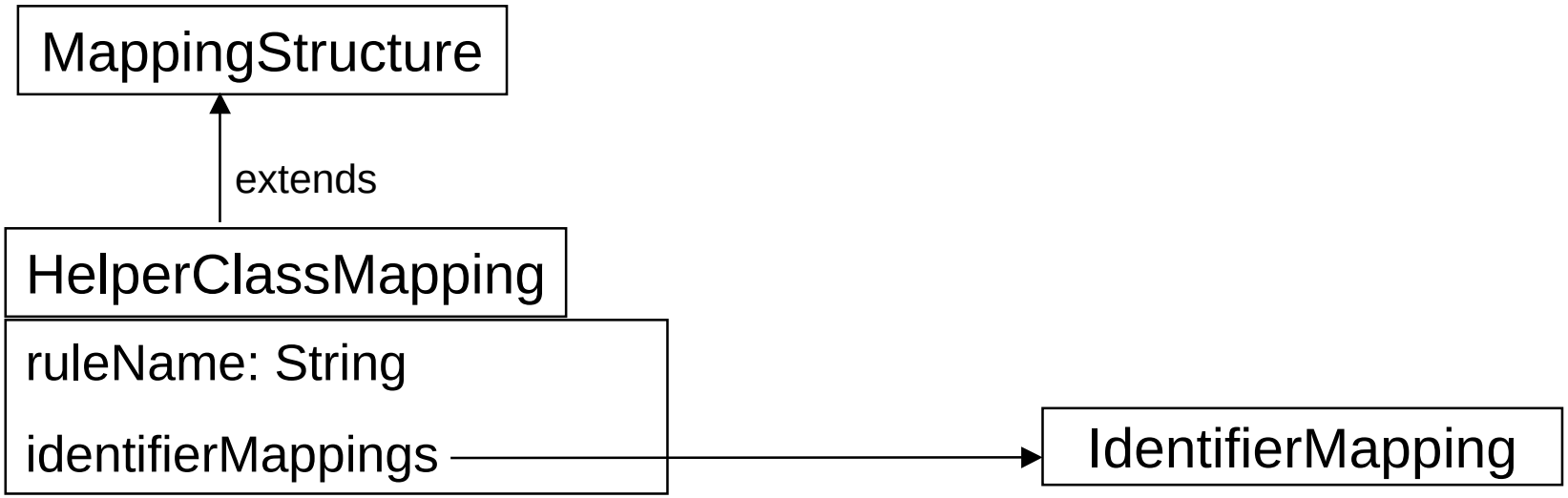


MappingStructure

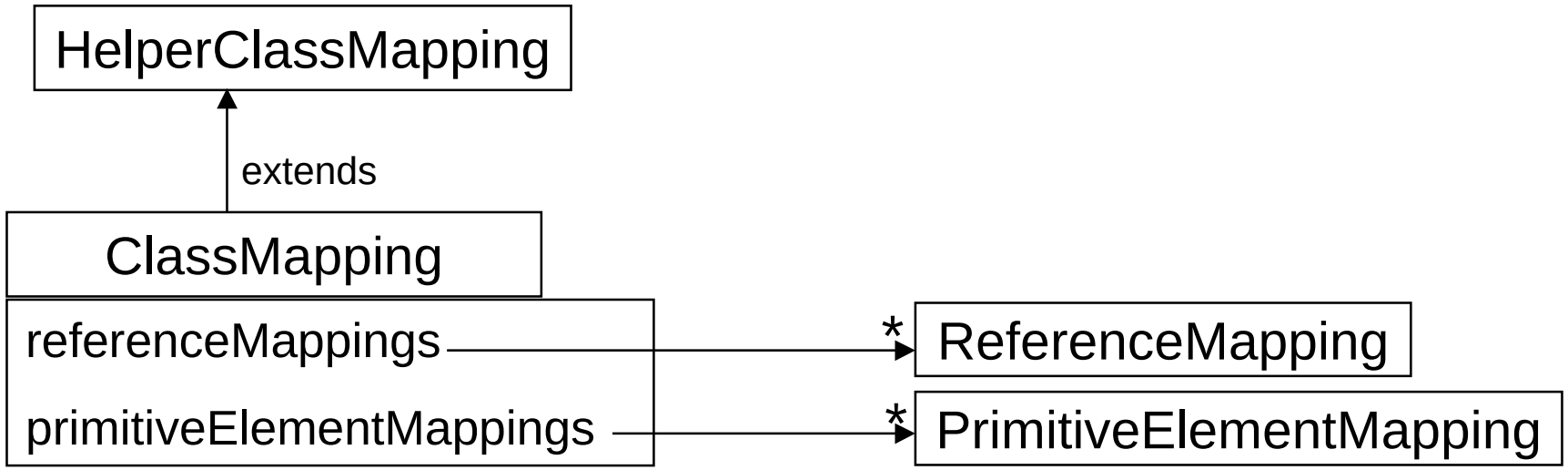


- A MappingStructure defines the mapping of a generic meta object (linkedMO) according to
  - a mapper name and
  - a project name.

HelperClassMapping

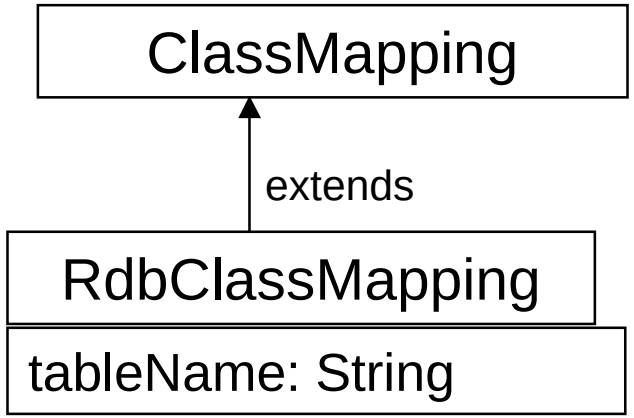


ClassMapping



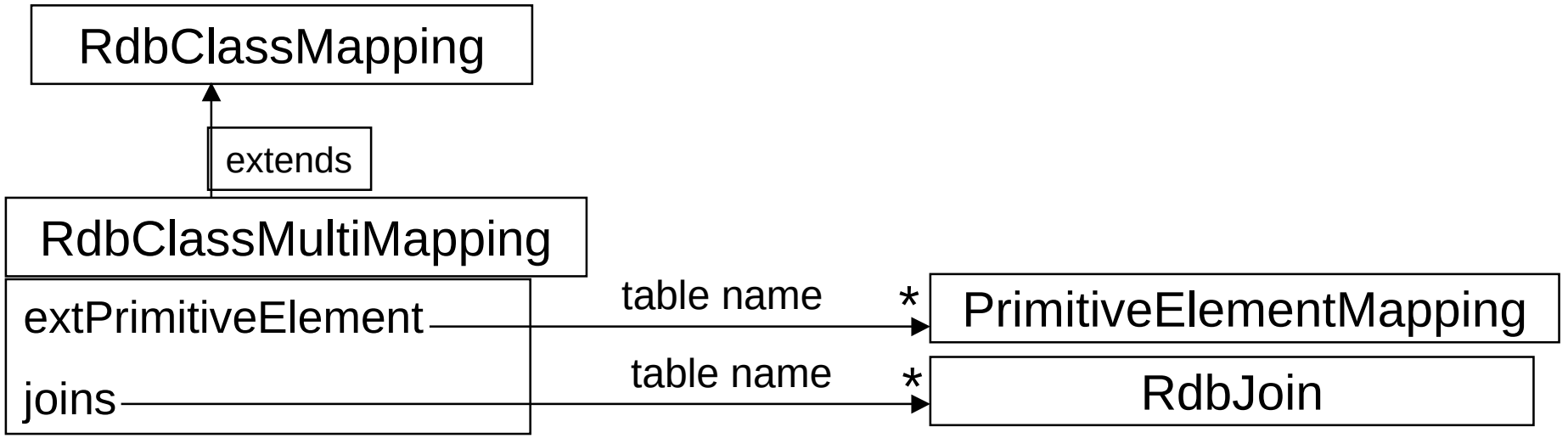
- A ClassMapping defines the mapping of class.
- The inherited ‘linkedMO’ attribute references the Class MetaObject.
- It contains one or several ReferenceMapping and one or several PrimitiveElementMapping.

RdbClassMapping



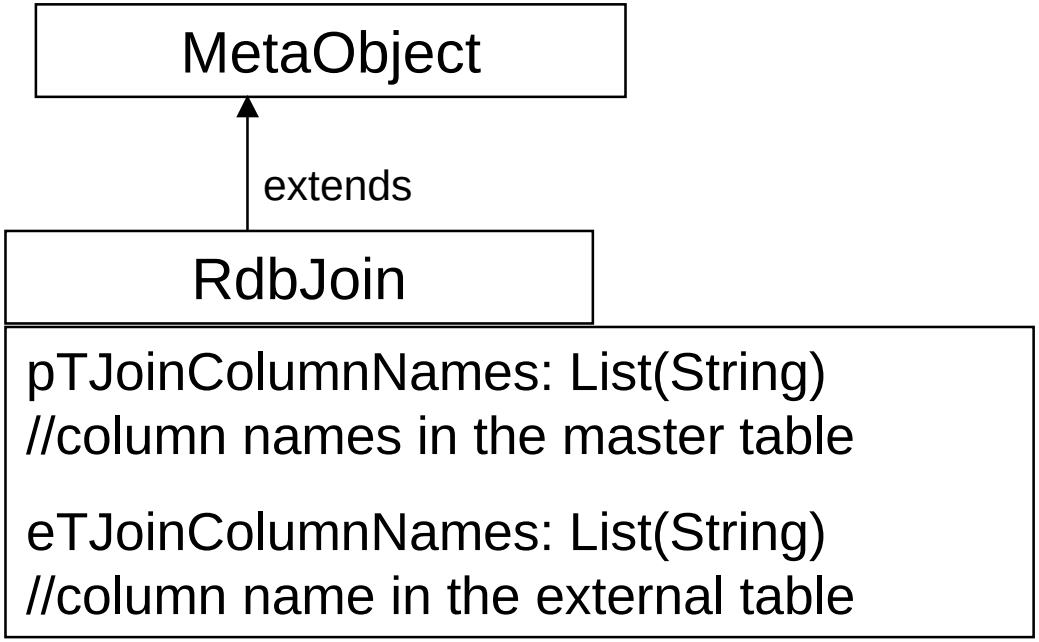
- A RdbClassMapping defines the mapping of a class onto a single relational table.

RdbClassMultiMapping



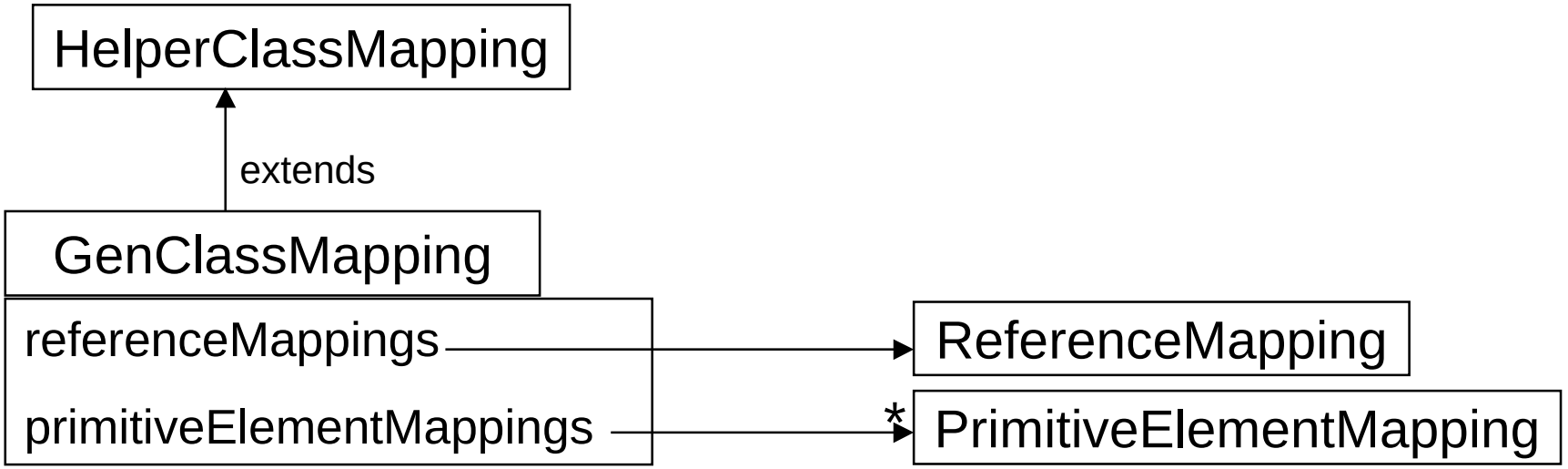
- A RdbClassMultiMapping defines the mapping of class onto several relational tables.
- It is considered that there is one “master” table
  - This is the table named in RdbClassMapping
- In addition there are other external tables
  - containing other external PrimitiveElement
  - accessible through joins

RdbJoin



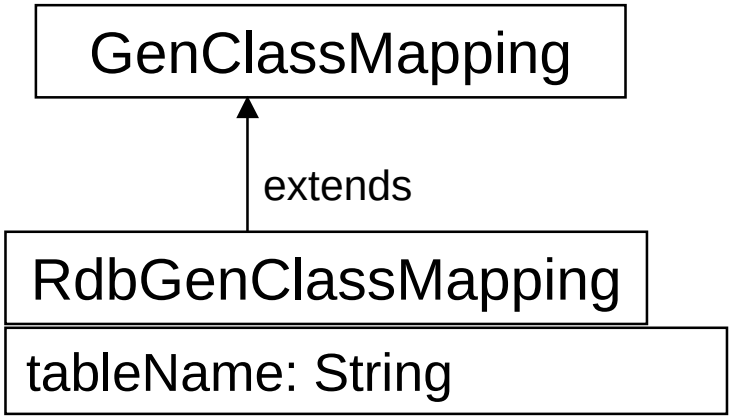
- A RdbJoin defines a join between the master table and an external table.

GenClassMapping



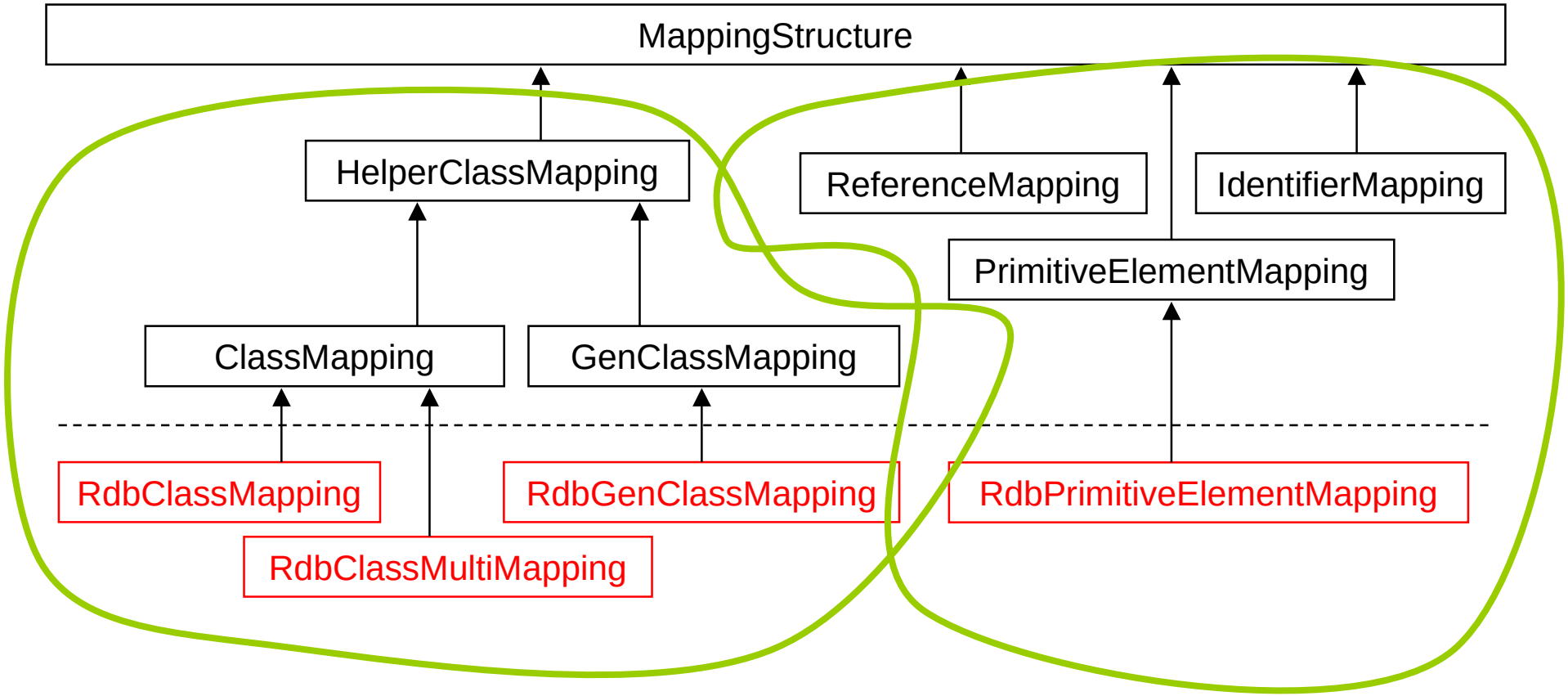
- A GenClassMapping defines a mapping of a generic class (Collection, Set, …).
- The inherited ‘linkedMO’ references the GenClassRef MetaObject.

RdbGenClassMapping

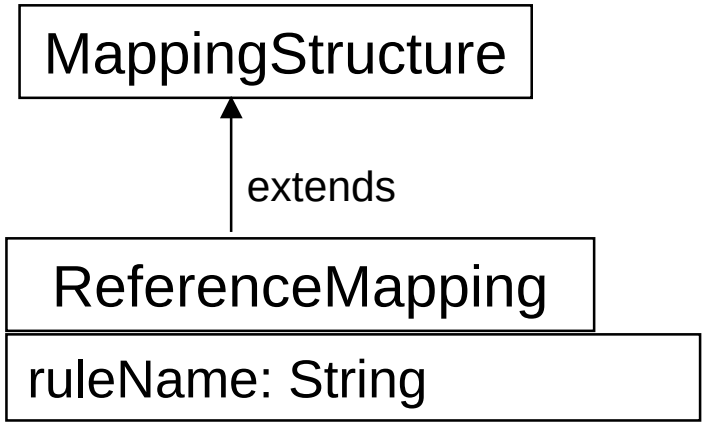


- A RdbGenClassMapping defines a mapping of a generic class on a single relational table.

Mappings overview

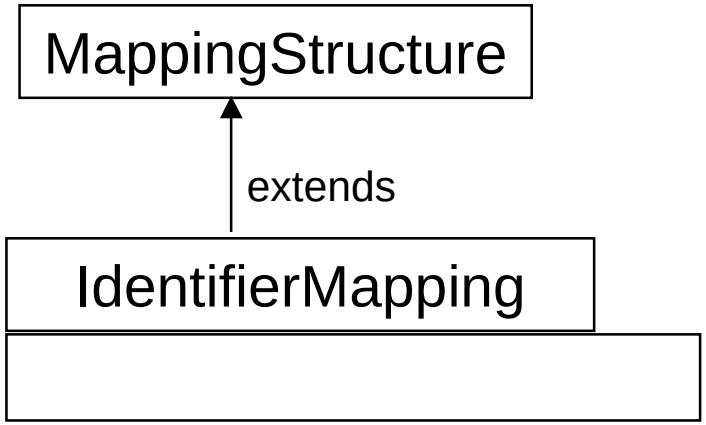


ReferenceMapping



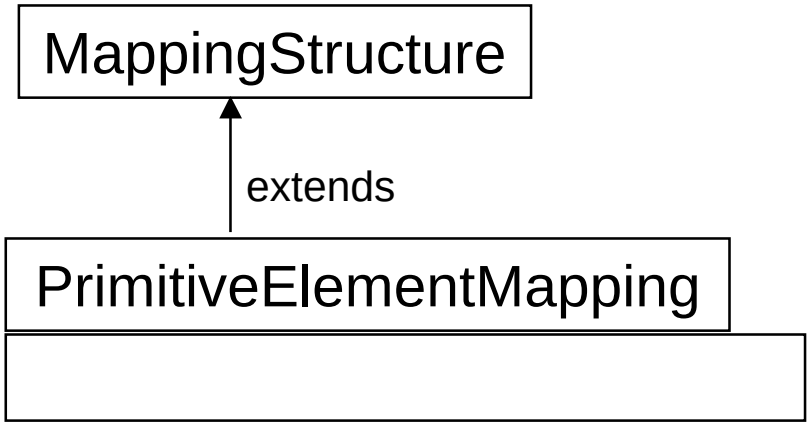
- A ReferenceMapping is used to link the Mapping to the particular NameDef to map.
- The mapping of the NameDef fields is described in the corresponding ClassMapping or GenClassMapping.
- Several rules are possible for mapping the reference
  - The rule name indicates how the reference is mapped.
  - For more information see the rule definitions.
- The inherited ‘linkedMO’ attribute references the NameDef MetaObject.

IdentifierMapping



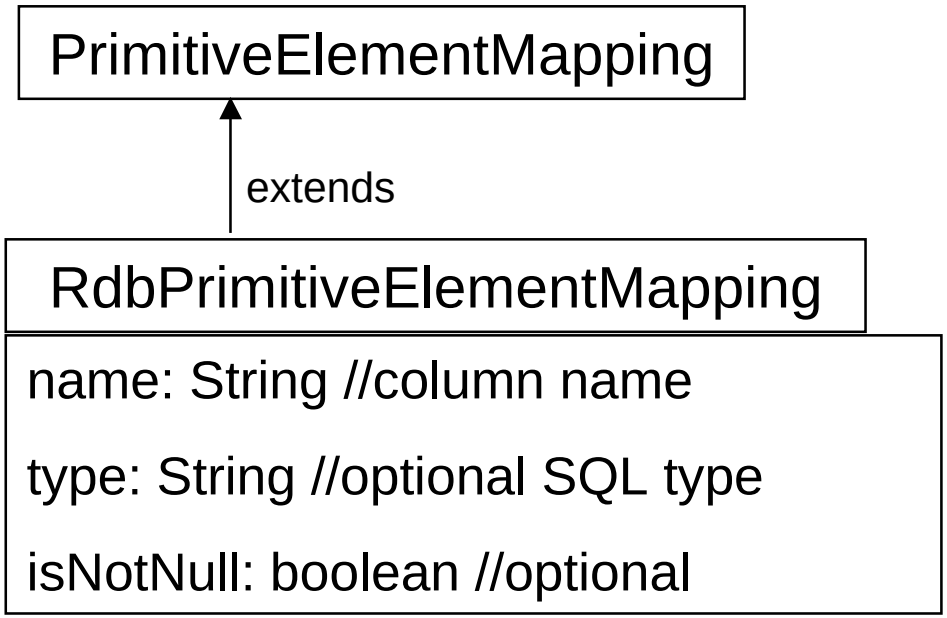
- An IdenfitierMapping is used to link the Mapping to the particular NameDef to map.
- The mapping of the NameDef fields is described in the corresponding ClassMapping or GenClassMapping.
- Unlike references, there is only one implicit way to map identifiers (no choice of rule).
- The inherited ‘linkedMO’ attribute references the NameDef MetaObject.

PrimitiveElementMapping



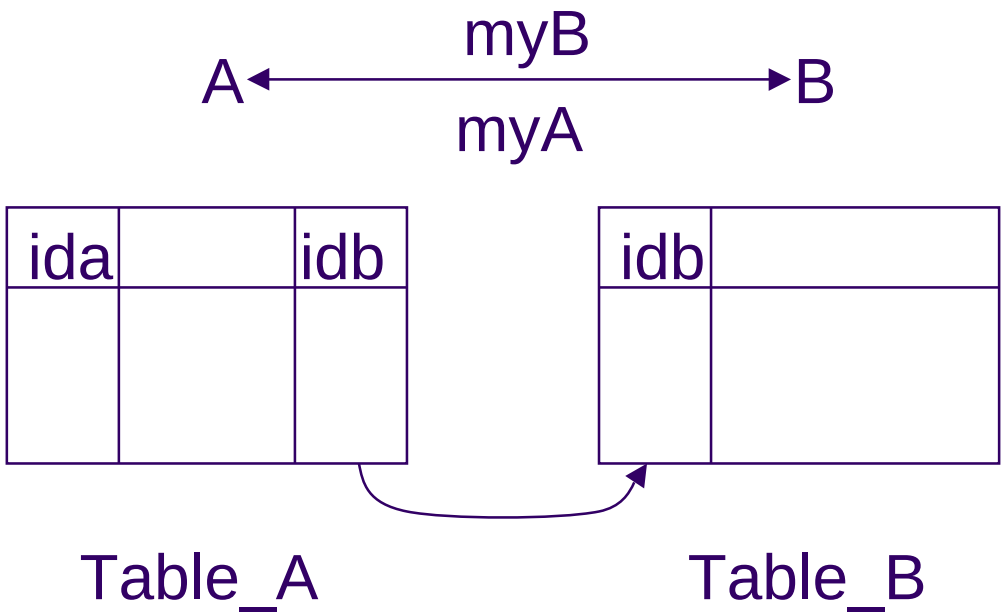
- A PrimitiveElementMapping defines the mapping of a primitive element (or scalar field).
- The inherited ‘linkedMO’ attribute references the PrimitiveElement MetaObject.

RdbPrimitiveElementMapping

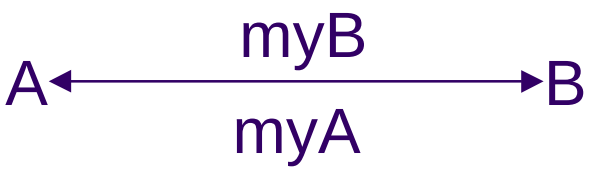


Example

One-One (1/5)



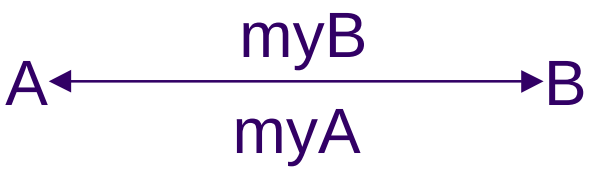
One-One (2/5)



Creation of the generic structure of A:

```
Package p = manager.createPackage(“”);
Class aclass = p.createClass(“A”, false);

// Primitive elements
aclass.createHiddenField(“ida”, PTypeSpace.LONG);
aclass.createHiddenField(“idb”, PTypeSpace.LONG);

// identifier of the A class
NameDef aNameDef = aclass.createNameDef();
aNameDef.setFieldName(“ida”);

// reference “myB”
Class bclass = p. createHiddenClass(“B”);
ClassRef bcr = aclass.createClassRef(“myB”, bclass);
NameDef myBNameDef = bcr.createNameDef();
 myBNameDef.setFieldName(“idb”);
```

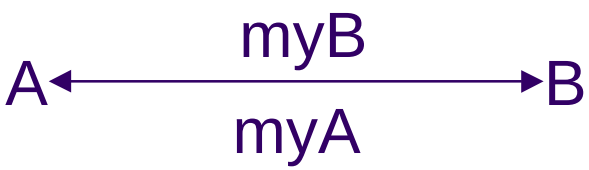
One-One (3/5)



Creation of the generic structure of B:

```
p.moveHiddenToClass("B");

// Primitive elements
bclass.createHiddenField("idb", PTypeSpace.LONG);
bclass.createHiddenField("ida", PTypeSpace.LONG);

// identifier of the B class
NameDef bNameDef = bclass.createNameDef();
bNameDef.setFieldName("idb");

// reference "myA"
ClassRef acr = bclass.createClassRef("myA", aclass);
NameDef myANameDef = acr.createNameDef();
myANameDef.setFieldName("ida");
```

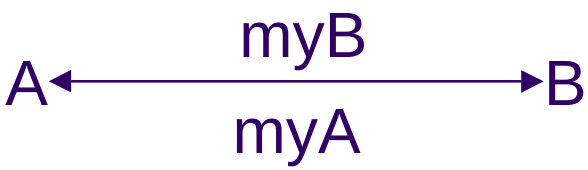
One-One (4/5)



Creation of the mapping part of A:

```
ClassProject acp = aClass.createClassProject(“”);
Mapping am = acp.createMapping(“rdb”);
acp.addMapping(am);
ClassMapping acm = ((RdbMapping) am).createClassMapping(“”, aClass, am);
((RdbClassMapping) acm).setTableName(“table_A”);

//Specify the class NameDef to use
acm.createIdentifierMapping(aNameDef);

//Specify the NameDef to use for the “myB” field
acm.createReferenceMapping(myBNameDef);

//Mapping the primitive elements
PrimitiveElementMapping ida_map = ((RdbClassMapping) acm)
    .createPrimitiveElementMapping( aClass.getTypedElement(“ida”), “ida_col”);
acm. addPrimitiveElementMapping(ida_map);

PrimitiveElementMapping idb_map = ((RdbClassMapping) acm)
    .createPrimitiveElementMapping( aClass.getTypedElement(“idb”), “idb_col”);
acm. addPrimitiveElementMapping(idb_map);
```

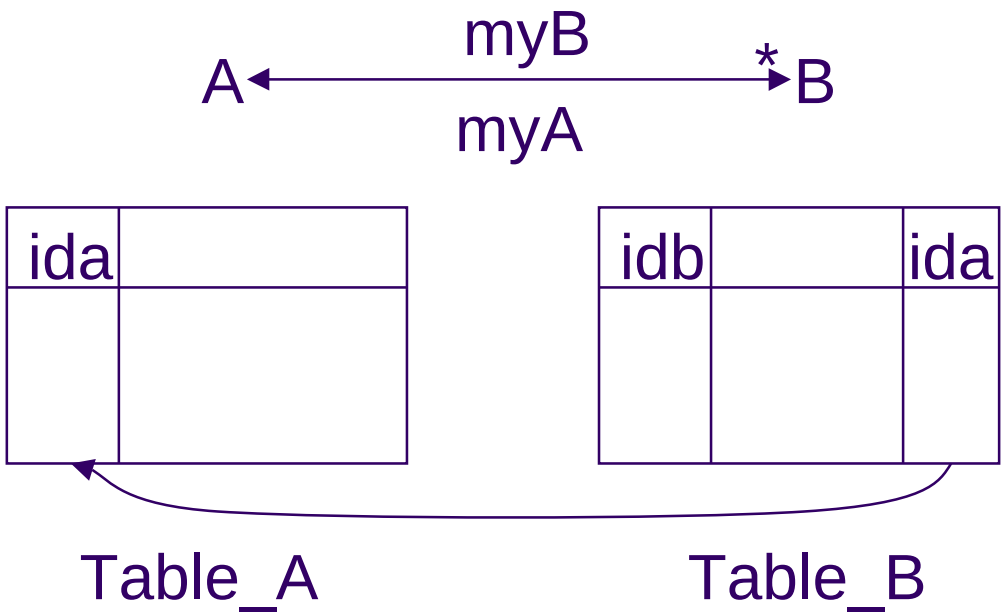
One-One (5/5)



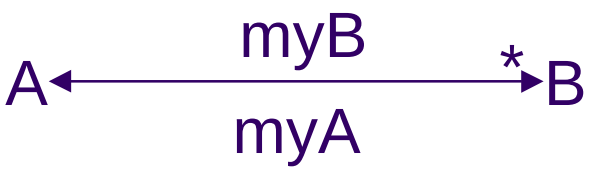
Creation of the mapping part of B:

```
ClassProject bcp = bClass.createClassProject("");
Mapping bm = bcp.createMapping("rdb");
bcp.addMapping(bm);
ClassMapping bcm = ((RdbMapping) bm).createClassMultiMapping("", bClass, bm);
((RdbClassMapping) bcm).setTableName("table_B");
//Specify the class NameDef to use
bcm.createIdentifierMapping(bNameDef);
//Specify the NameDef to use for the "myA" field
bcm.createReferenceMapping(myANameDef);

//Mapping the primitive elements
PrimitiveElementMapping idb_map = ((RdbClassMapping) bcm)
    .createPrimitiveElementMapping( bClass.getTypedElement("idb"), "idb_col");
bcm. addPrimitiveElementMapping(idb_map);
PrimitiveElementMapping ida_map2 = ((RdbClassMultiMapping) bcm)
    .createPrimitiveElementMapping( aClass.getTypedElement("ida"), "ida_col");
((RdbClassMultiMapping) bcm). addExtPrimitiveElementMapping("table_A", ida_map2);
RdbJoin joinToA = ((RdbClassMultiMapping) bcm).createJoin();
joinToA. addJoinColumnNames("idb_col", "idb_col");
((RdbClassMultiMapping) bcm). addJoin("table_A", joinToA );
```

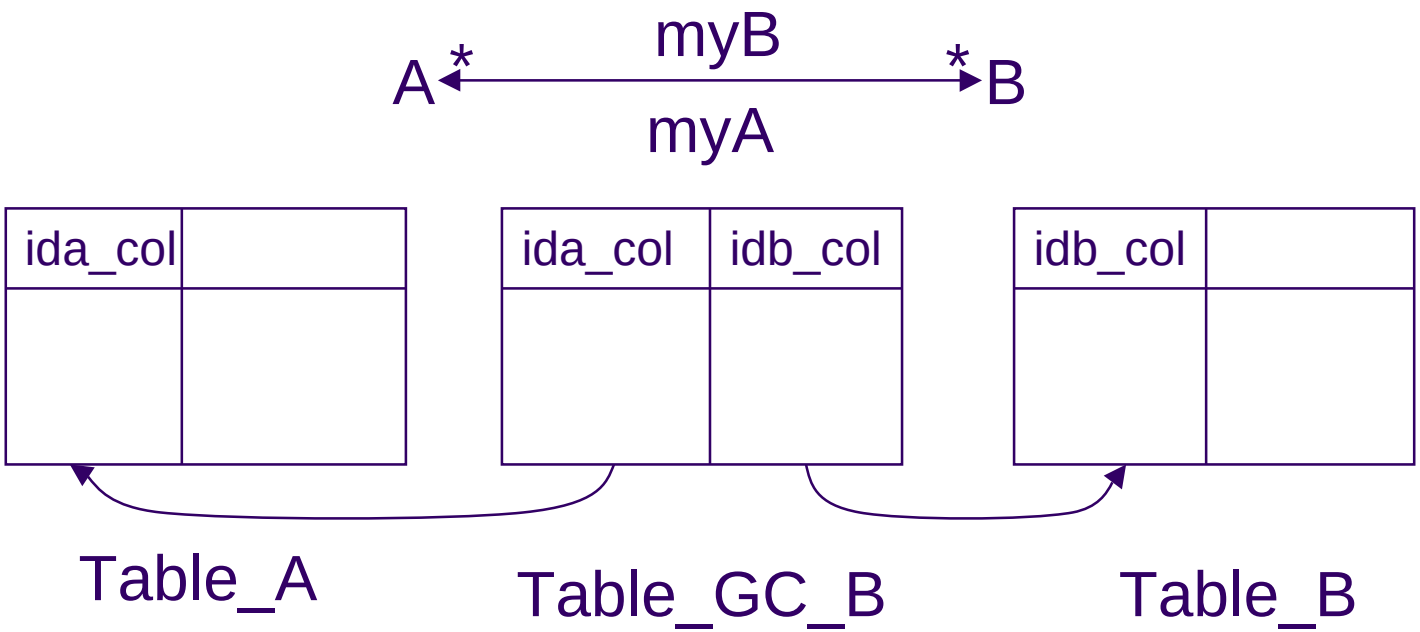
One-Many (1/X)



One-Many (2/X)



Many-Many (1/X)


A
*
myB
*
B
myA
ida_col
ida_col
idb_col
idb_col
Table_A
Table_GC_B
Table_B

Many-Many (2/X)