

The image features the Hybris Software logo, which consists of a stylized 'h' inside a circle, followed by the words 'hybris software' in a bold, lowercase sans-serif font. Below this, the text 'An SAP Company' is written in a smaller, uppercase sans-serif font. The background is a dark blue gradient with a large, abstract, low-poly geometric shape in a lighter blue shade on the right side.

(h) hybris software
An SAP Company

hybris Developer Training Part I - Core Platform

Data Modeling



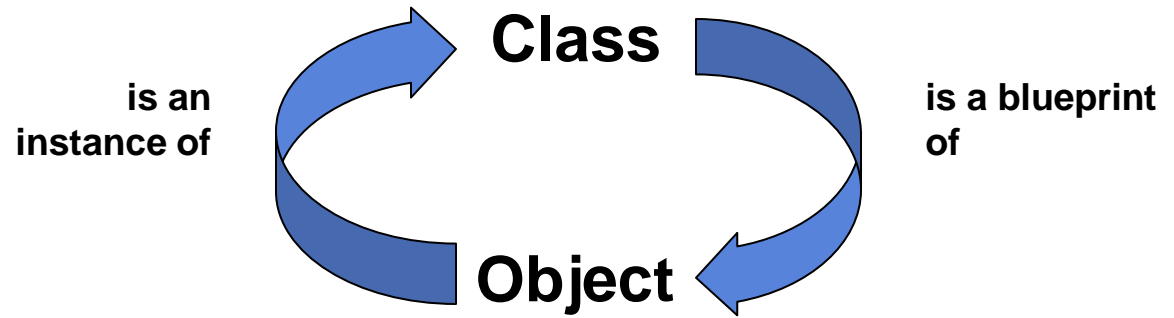
Introduction to the Type System

Collections & Relations

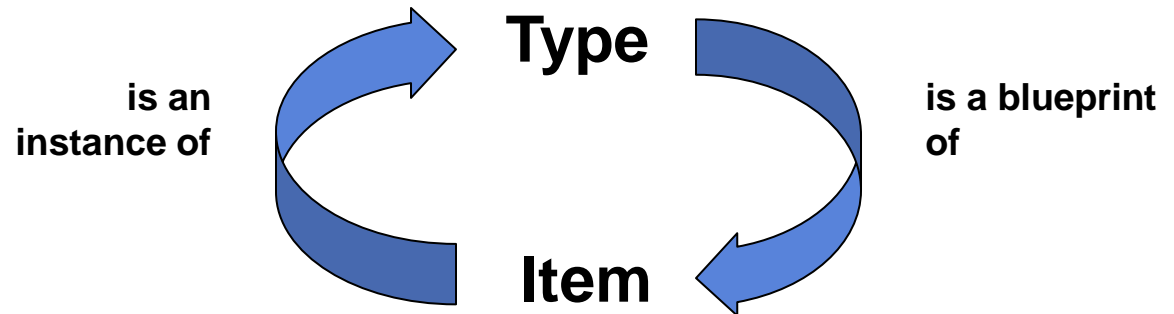
Deployment

Type System Localization

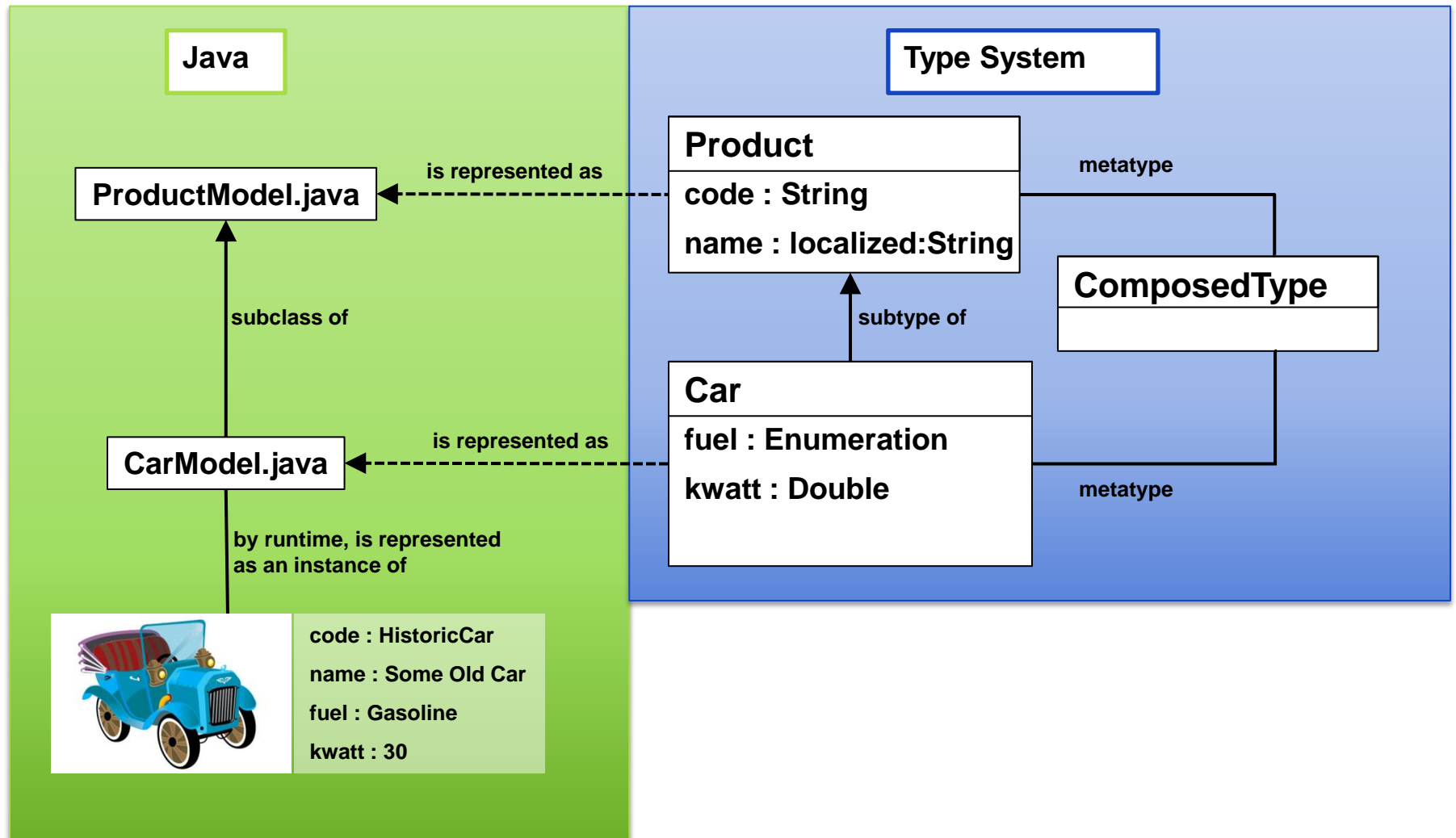
→ **Java:**



→ **hybris:**



Java Classes vs hybris Types



AtomicType

- Represents Java value objects which are mapped to database types
 - Java Primitives: `int`
 - Wrapper: `Integer`
 - Some Reference types: `java.util.Date`

CollectionType

- Represents a typed collection

MapType

- Represents a typed Map

ComposedType

→ Composed object of other hybris types

EnumerationMetaType

→ ComposedType which describes enumerations

RelationType

→ ComposedType which describes binary relations between items

→ Create new types:

- Define a type by extending already existing types, such as:

```
<itemtype code="Car" extends="Product">
```

- Define “completely new types”, such as:

```
<itemtype code="Car">
```

(implicitly extends from **GenericItem**)

→ Extend existing types:

- Add attribute definitions to existing types (attribute injection), such as:

```
<itemtype code="Product" ...>
```

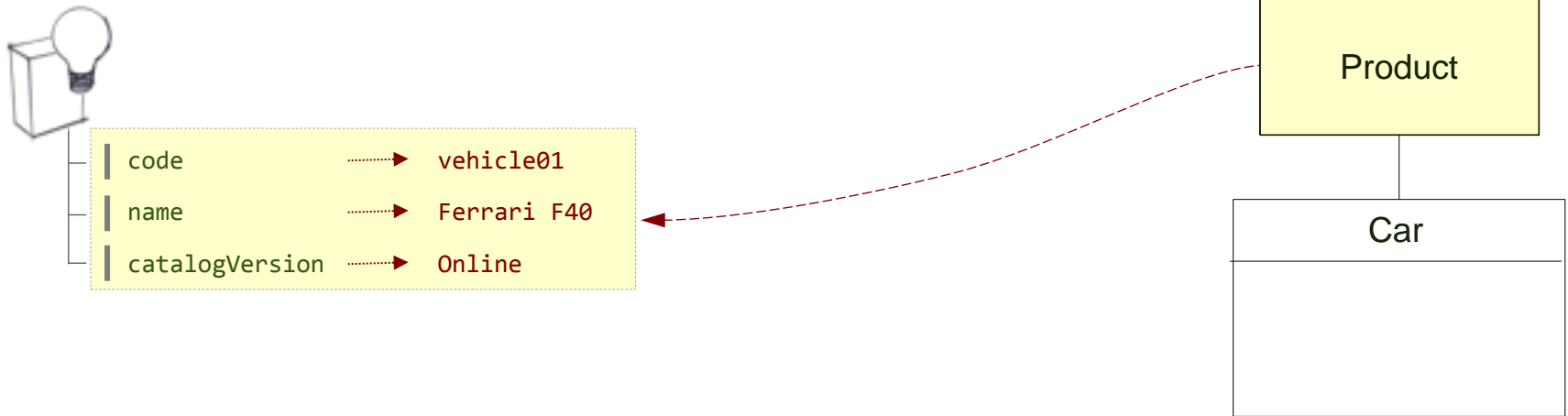
```
...
```

```
    <attribute qualifier="MyAttribute">  
</itemtype>
```

- Redefine inherited attribute definitions from super type

```
<attribute qualifier="code" redeclare="true">
```

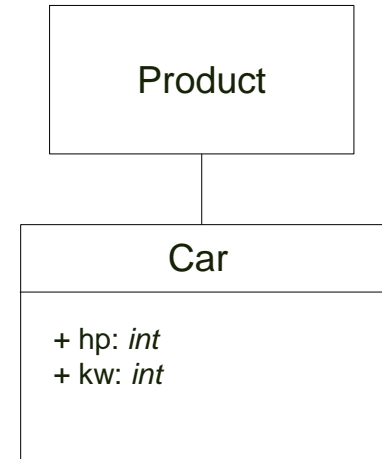
If you change the attribute's java type, the new type must extend the original type



```
<items>
  <itemtypes>
    <itemtype code="Car" extends="Product" autocreate="true" generate="true"
      jaloclass="de.hybris.platform.leture.jalo.car">
      ...
```



code	vehicle01
name	Ferrari F40
catalogVersion	Online
hp	300
kw	212

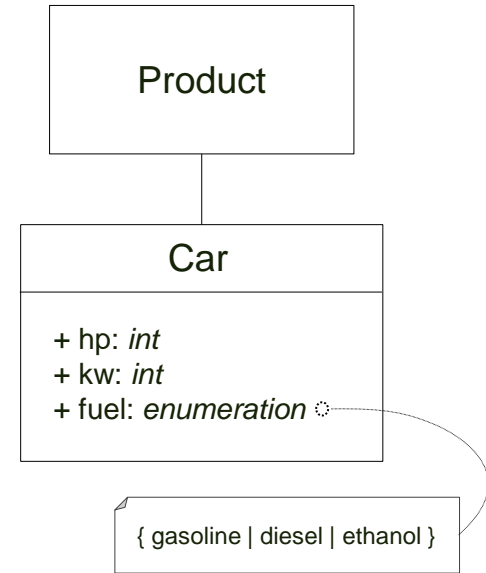


```
<items>
  <itemtypes>
    <itemtype code="Car" extends="Product" autocreate="true" generate="true">
      <attributes>
        <attribute qualifier="hp" type="java.lang.Integer">
          <description>Horse Power</description>
          <persistence type="property"/>
        </attribute>
        <attribute qualifier="kw" type="java.lang.Integer">
          <description>Kilowatt</description>
          <persistence type="dynamic"
            attributeHandler="kwPowerAttributeHandler"/>
          <modifier write="false" />
        </attribute>
      </attributes>
    </itemtype>
  </itemtypes>
</items>
```

...

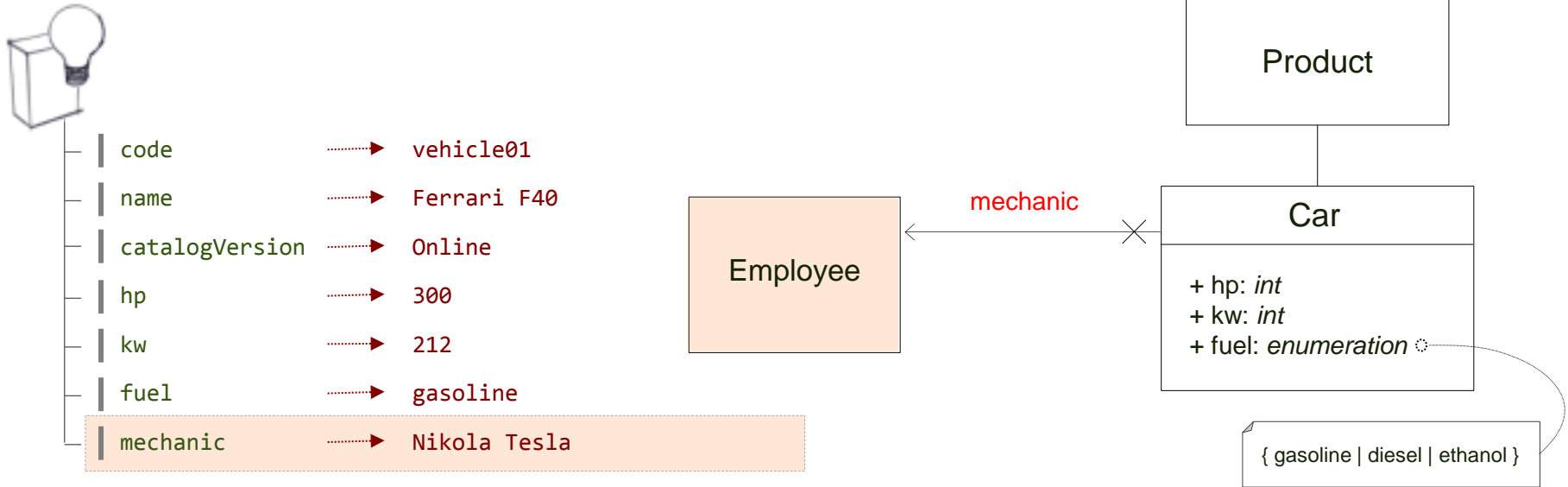


code	vehicle01
name	Ferrari F40
catalogVersion	Online
hp	300
kw	212
fuel	gasoline

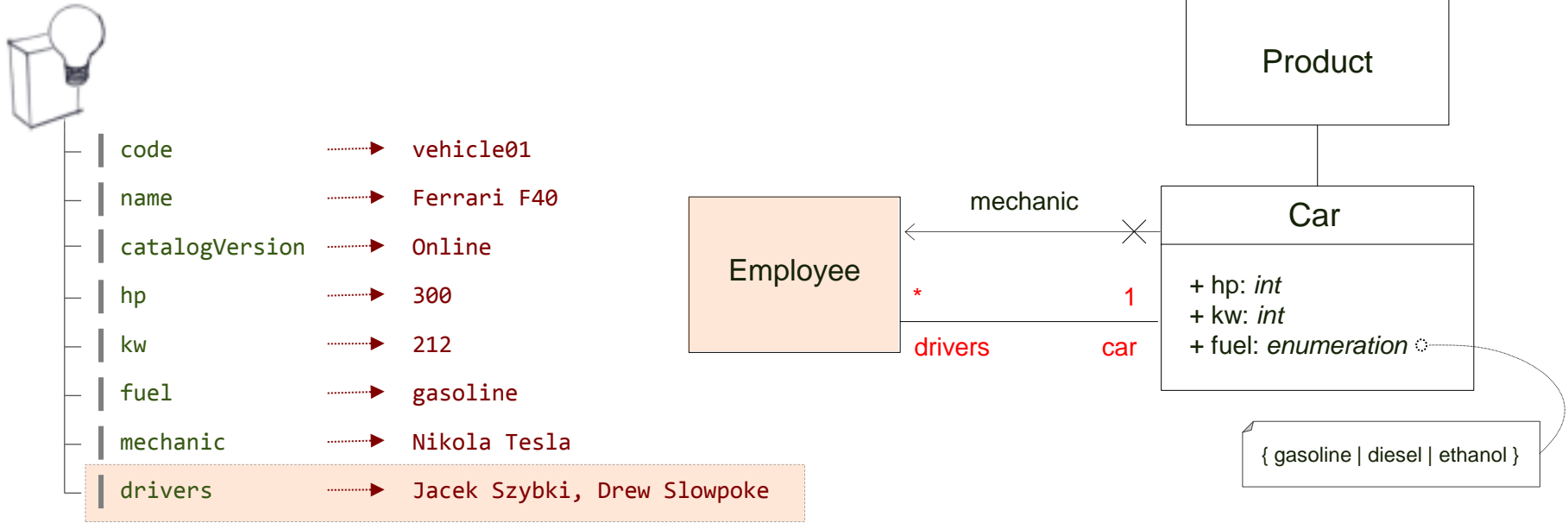


```
<items>
  <enumtypes>
    <enumtype code="FuelEnumeration" generate="true" autocreate="true"
      dynamic="true">
      <value code="diesel"></value>
      <value code="gasoline"></value>
      <value code="ethanol"></value>
    </enumtype>
  </enumtypes>

  <itemtypes>
    <itemtype code="Car" extends="Product" autocreate="true" generate="true">
      <attributes>
        ...
        <attribute qualifier="fuel" type="FuelEnumeration">
          <description>Fuel for this car</description>
          <persistence type="property"></persistence>
        </attribute>
      </attributes>
    </itemtype>
  </itemtypes>
```



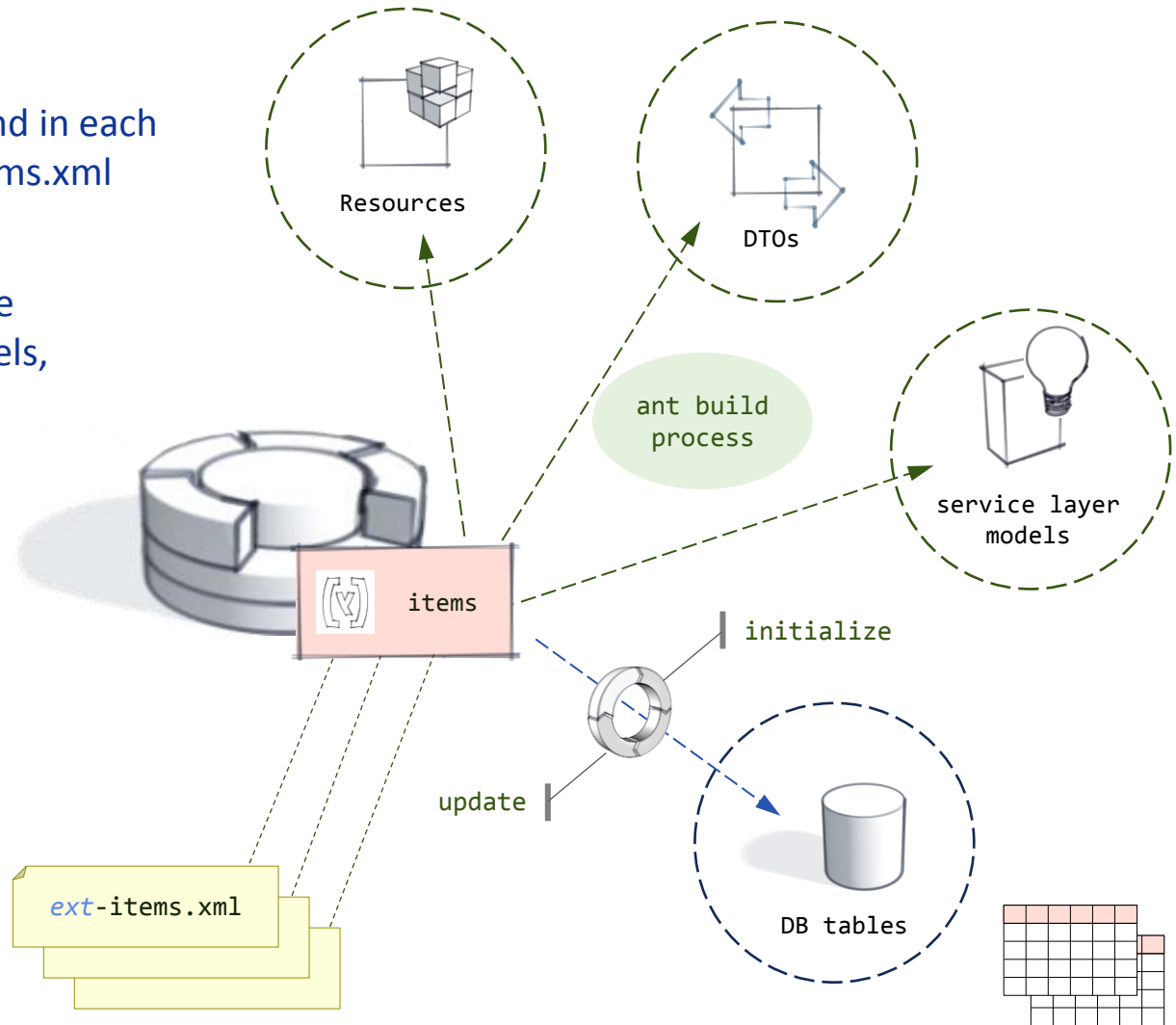
```
<items>
  <itemtypes>
    <itemtype code="Car" extends="Product" autocreate="true" generate="true">
      <attributes>
        ...
        <attribute qualifier="mechanic" type="Employee">
          <modifier read="true" write="true" search="true" />
          <persistence type="property" />
        </attribute>
        ...
      </attributes>
    </itemtype>
  </itemtypes>
</items>
```



```
<items>
...
<relations>
  <relation code="Car2EmployeeRelation"
    localized="false" autocreate="true" generate="true">
    <sourceElement qualifier="car" type="Car" cardinality="one" />
    <targetElement qualifier="drivers" type="Employee" cardinality="many" />
  </relation>
</relations>

<itemtypes>
...
```

1. hybris item definitions are found in each extension's *extensionName-items.xml*
2. The ant process assembles type definitions and generates Models, DTOs, and Resources.
3. hybris also creates the required tables.
4. Invoking `initialize` or `update` creates the required table.



1. What is the equivalent of a Java object in hybris?
2. What is the difference between a composed type and an atomic type in hybris?
3. What file is used to configure types in hybris?
4. In hybris, can you create new types, or only extend existing types?
5. Do you have to define a model-class in Java after defining it in xml?

Introduction to the Type System

Collections & Relations

Deployment

Type System Localization

- Collection of target type
- Can be used as attribute type of a ComposedType
- Allows you also to define **AtomicTypes** collections
- Performance considerations: Accessing, Searching
- Database integrity considerations

```
<collectiontype code="StringCollection"
               elementtype="java.lang.String" autocreate="true" />
<collectiontype code="LanguageCollection"
               elementtype="Language" autocreate="true"/>

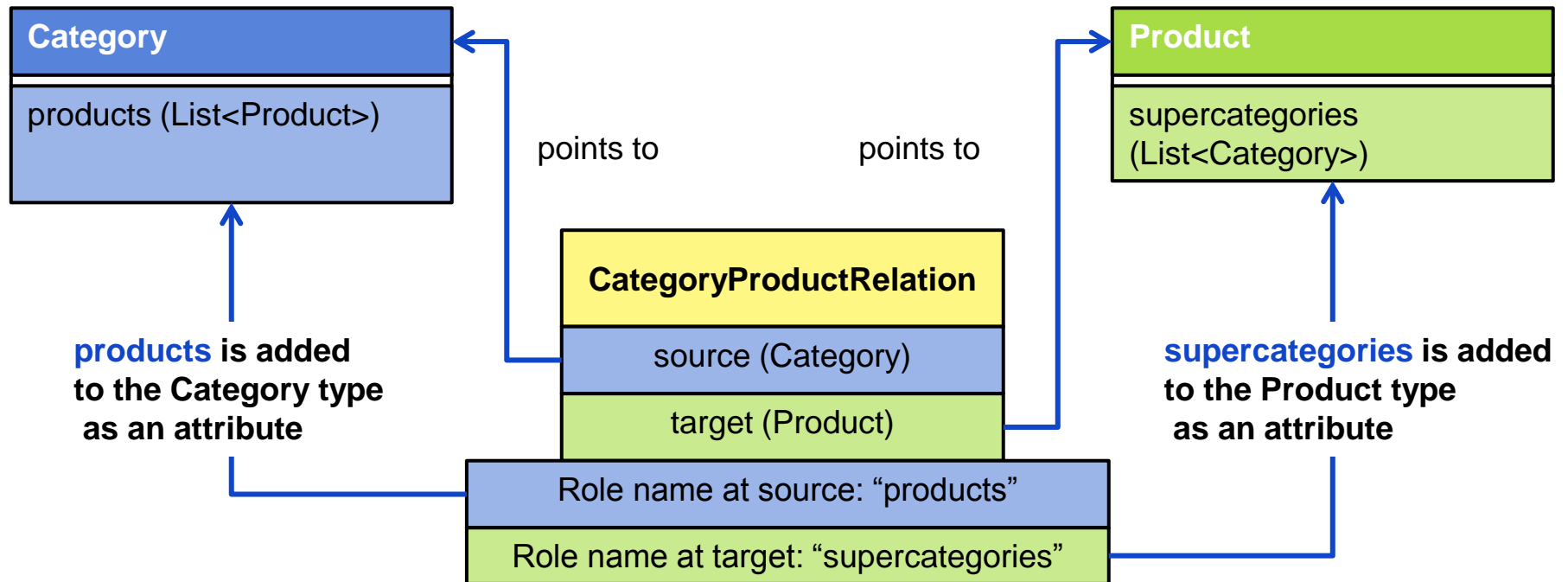
...

<itemtype code="..." ...
  <attribute qualifier="urlPatterns" type="StringCollection">
  <attribute qualifier="writeableLanguages" type="LanguageCollection">
  ...
```

Relations



- ➔ One2Many and Many2Many
- ➔ Both sides are (can be) aware of the other



What's so Important About Relations?



If in doubt: Use **relations**, not **CollectionTypes**, because:

- Opposite side is not “aware” of the **CollectionType**
- **CollectionTypes** are stored in a database field as a comma-separated list of references (PKs) or atomic values
- Can cause overflow
- More difficult to search and generally lower performance

Introduction to the Type System

Collections & Relations

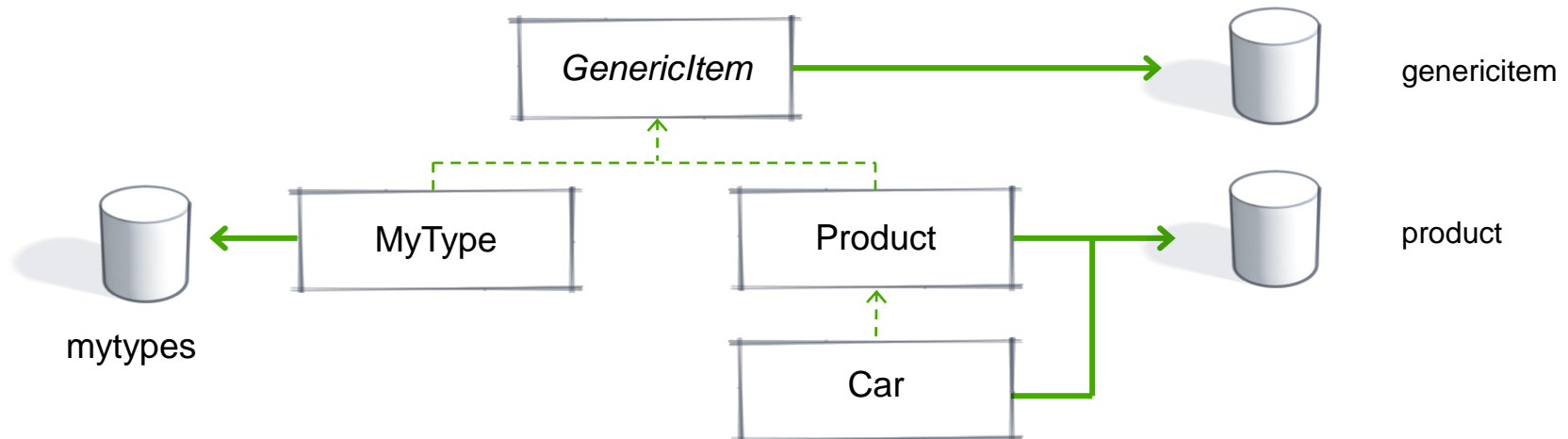
Deployment

Type System Localization

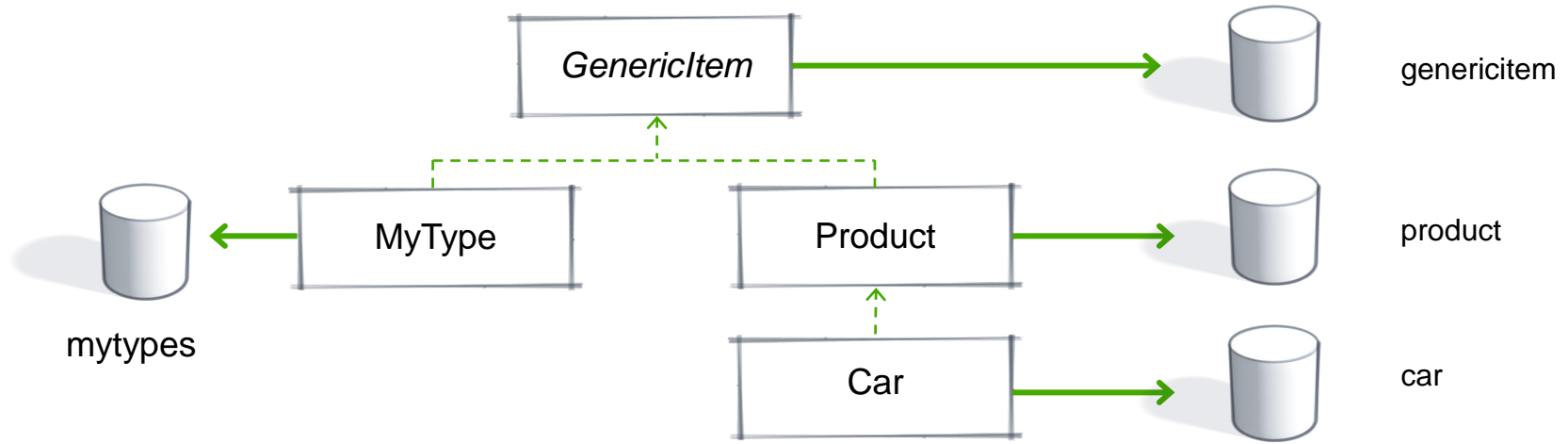
Object Relational Mapping — Storing objects in the DB



- ➔ By default, items for a given type are stored in the same database tables as its supertype
- ➔ Specify any item type's *deployment* to store its items in its own db tables.
- ➔ hybris recommends that deployment be specified for the first layer of *GenericItem* subtypes
 - ➔ Consider carefully the performance implications of specifying deployment for other item types
 - ➔ Set `build.development.mode = true` in `local.properties` to mandate that all direct children of *GenericItem* have deployment specified.



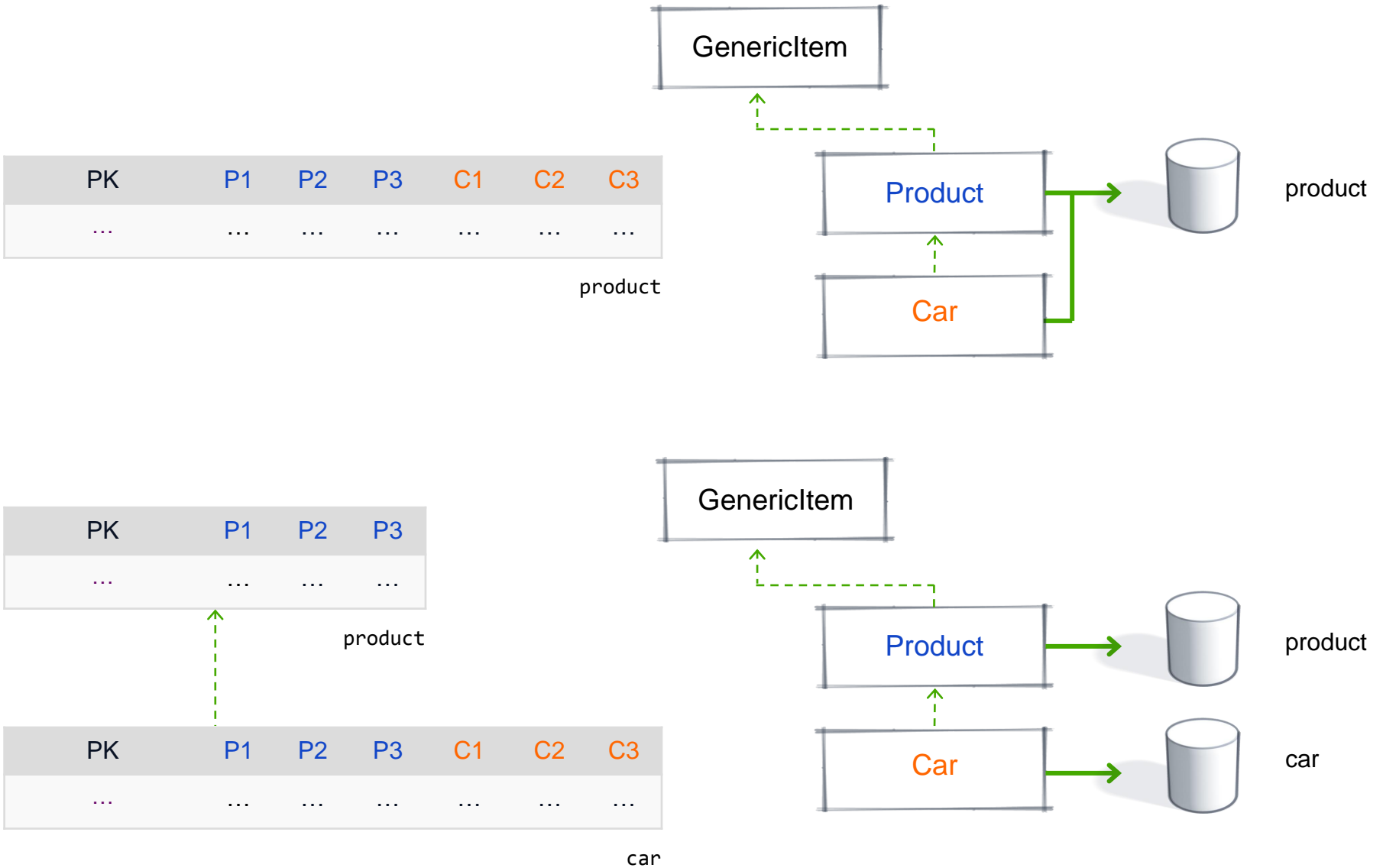
O-R Mapping – Deployment Example



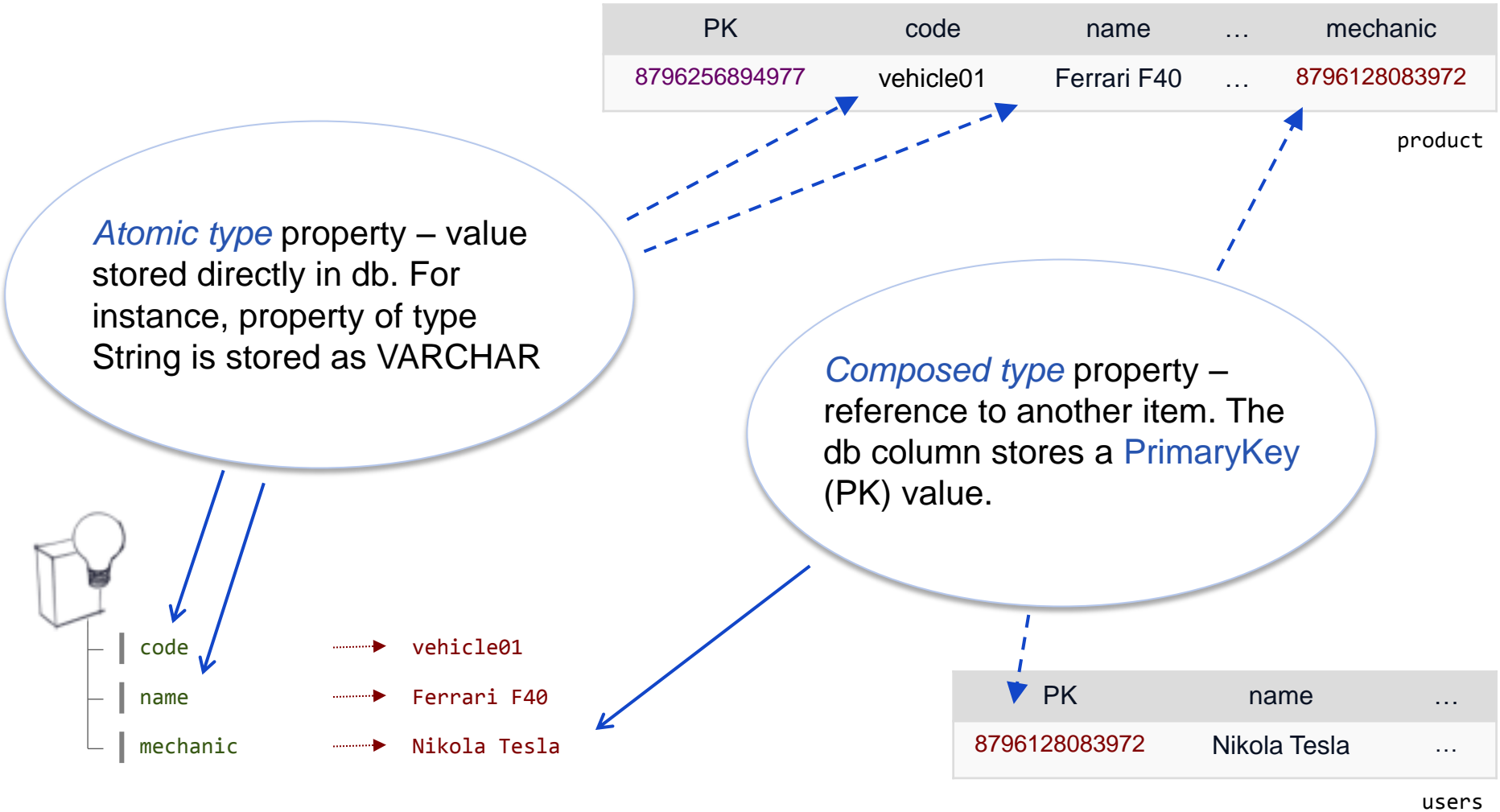
```
<itemtype code="Car" ...  
  <deployment table="cars"  
    typecode="20100" ...
```

0 .. 10000 are reserved by hybris
10000 .. 32767 are free for use,
with *some exceptions*

O-R Mapping – Table Structure



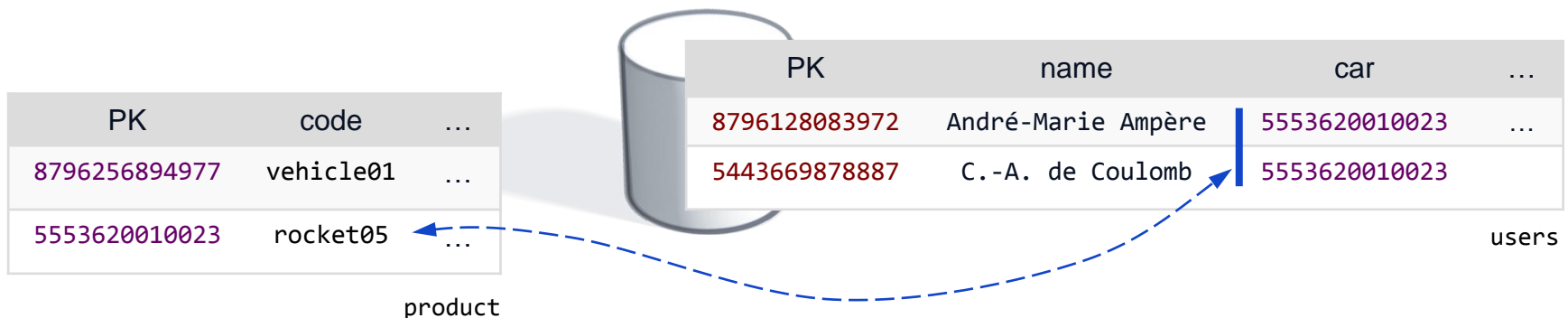
O-R Mapping – Attributes of a (Composed) Type



➔ One-to-Many

- ➔ Additional column at the many side which holds the PK of the One side
- ➔ Users table from the example below would have an additional column **car**

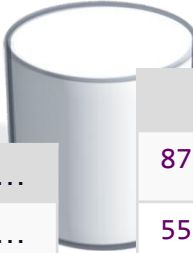
```
<relation generate="true" localized="false" code="Car2EmployeeRelation"
  autocreate="true">
  <sourceElement qualifier="car" type="Car" cardinality="one" />
  <targetElement qualifier="drivers" type="Employee"
cardinality="many"/>
</relation>
```



➔ Many-to-Many

➔ New database table which holds the **source** and **target** PKs

```
<relation code="CategoryProductRelation" autocreate="true" generate="true"
  localized="false">
  <deployment table="Cat2ProdRel" typecode="143"/>
  <sourceElement qualifier="supercategories" type="Category" cardinality="many"
    ordered="false">
    <modifiers read="true" write="true" search="true" optional="true"/>
  </sourceElement>
  <targetElement qualifier="products" type="Product" cardinality="many"
    collectiontype="list" ordered="true">
    <modifiers read="true" write="true" search="true" optional="true"/>
  </targetElement>
</relation>
```



PK	code	...
8796256894977	vehicle01	...
5553620010023	rocket05	...

product

Source	Target
8796256894977	3776876789221
5553620010023	3776876789221
5553620010023	6152677365115

Cat2ProdRel

PK	code	...
3776876789221	Commuting	...
6152677365115	Exploration	...

category

Collections

- ➔ Stored in one database column
- ➔ Comma separated list of **PKs** or **Atomic Values**

```
<collectiontype code="StringCollection"
                elementtype="java.lang.String" autocreate="true" />
<collectiontype code="LanguageCollection"
                elementtype="Language" autocreate="true" />
...
<itemtype code="..." ...
    <attribute qualifier="urlPatterns" type="StringCollection" />
    <attribute qualifier="writeableLanguages" type="LanguageCollection" />
    ...
```

PK	code	urlpatterns	writeableLanguages	...
8796256894977	Example1	http://ex1.com/a,http://ex2.com/b,http://ex3.com/c	93938293,93029304,01920394	...

product

Introduction to the Type System

Collections & Relations

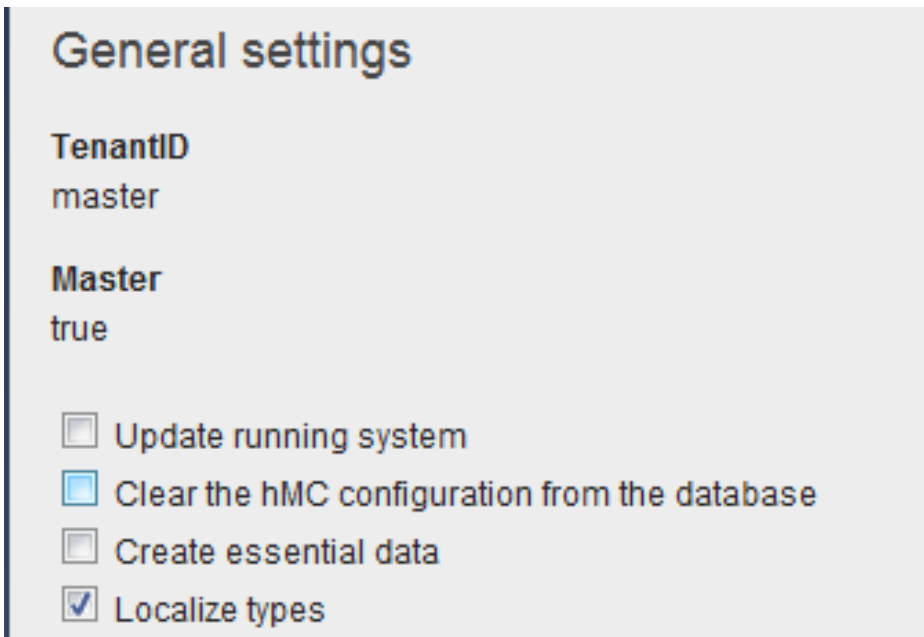
Deployment

Type System Localization

- ➔ The localization strings for types and attributes are stored in files named *extension-locales_XY.properties*
 - ➔ *extension*: the name of the extension
 - ➔ *XY*: the ISO code of the language
 - ➔ Properties convention:

```
type.{typename}.name=value
type.{typename}.description=value
type.{typename}.{attributename}.name=value
type.{typename}.{attributename}.description=value
type.{enumcode}.{valuecode}.name=value
```
- ➔ Use generator from hmc to get all properties keys generated for selected extensions

→ Type localizations are stored in the database



General settings

TenantID
master

Master
true

- ☐ Update running system
- ☒ Clear the hMC configuration from the database
- ☐ Create essential data
- ☒ Localize types

→ Overrides type localizations in the database with the ones from the `locales_XY.properties` files

1. Name the two ways to model a collection in hybris
2. How are collection types stored in the database?
3. Specify some advantages and disadvantages of collection types
4. Explain the notion of *deployment* in hybris. When should you use it?

- ➔ wiki.hybris.com/display/release5/Type+System+Documentation
- ➔ wiki.hybris.com/display/release5/items.xml
- ➔ [wiki.hybris.com/display/release5/
Specifying+a+Deployment+for+hybris+Platform+Types](https://wiki.hybris.com/display/release5/Specifying+a+Deployment+for+hybris+Platform+Types)

(x)