ORM (OBJECT RELATIONAL MAPPING)

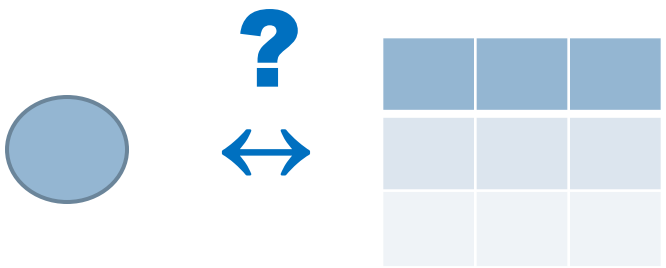
Veerle Ongenae

Inhoud

- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

ORM (Object Relational Mapping)

- Applicatie
  - Software
    - Object technologie (bv. Java, C#, …)
  - Data
    - Relationele gegevensbanken
- Afbeelding (mapping) tussen
  - Data in objecten
  - Data in tabellen
- Hoe en hoe realiseren?



Waarom ORM?

- JDBC – ADO.NET
  - tijdsintensief
- Gebruik DataSets
  - Beperkte controle bij compileren
- Data uit gegevensbank rechtstreeks binden aan componenten (.NET)
  - GridView …
  - Dichte koppeling
    - Front end
    - Gegevensbank
  - Groeiende applicatie
    - Onderhoud = nachtmerrie
- Scheiding datalaag en eigenlijke applicatie
  - Losse koppeling
  - Beter onderhoudbaar
  - Opsplitsbaar → werken in verschillende teams

Mogelijke implementaties

- Java ↔ DB
  - Hibernate (www.hibernate.org)
  - Java Persistence API (http://java.sun.com/javaee/5/docs/tutorial/doc/bnbpy.html en http://java.sun.com/javaee/technologies/persistence.jsp)
  - …
- .NET ↔ DB
  - NHibernate (C#) (http://www.nhibernate.org)
  - LINQ
    - Deel van het .NET-platform
  - …

Gebruik NHiberate in .NET

- DLL's downloaden en toevoegen aan project (in bin-map)
- web.config aanpassen
  - Informatie gegevensbank
    - Provider
    - Connectiestring
    - …
  - NHibernate parameters
  - (Eventueel) loging: log4net

Inhoud

- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

Object ↔ Tabel ??

- Eigenschap v.e. object → nul of meerdere kolommen
  - Nul?
    - Sommige informatie moet niet bewaard worden
    - Vb. Gemiddelde
      - Berekend op basis van informatie uit DB
  - Meer dan één?
    - Eigenschap = object
      - Heeft op zijn beurt eigenschappen
    - Vb. Persoon-object heeft als eigenschap een Adres-object

Schaduwinformatie

- Shadow Information
- Data
  - Nodig om informatie te bewaren (in DB)
    - Bv. Primaire sleutel
  - Geen deel van business model

Afbeelding bepalen

- NHibernate
  - XML
    - Één bestand per object (of per verzameling van objecten)
    - .hbm.xml-bestand
  - Attributen

Eigenschap object ↔ één kolom


```
public partial class ProductOrder {
    private int? _productOrderID;
    private int? _customerID;
    private string _productName;
    private int _quantity;
    private decimal _totalCost;
    // Property Definitions...
}
```

ProductOrder
ProductOrderID
CustomerID
ProductName
Quantity
TotalCost

DDL

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <class name="ProductOrder, __Code" table="ProductOrder">
    <id name="ProductOrderID" column="ProductOrderID"
      type="Int32" unsaved-value="null">
      <generator class="native" />
    </id>
    <property name="CustomerID" column="CustomerID"
    type="Int32" />
    <property name="ProductName" column="ProductName"
        type="String" />
    <property name="Quantity" column="Quantity" type="Int32" />
    <property name="TotalCost" column="TotalCost" type="Decimal"
    />
  </class>
</hibernate-mapping>
```

ID


```
<id name="ProductOrderID" column="ProductOrderID"
        type="Int32" unsaved-value="null">
       <generator class="native" />
</id>
```

- <id> Vereist in NHibernate
  - NHibernate ondersteunt samengestelde sleutels
    - Gebruik enkel voor legacy databases
- <generator>
  - Hoe id aangemaakt bij nieuwe objecten?
  - native
    - Standaard generator huidige database

Inhoud

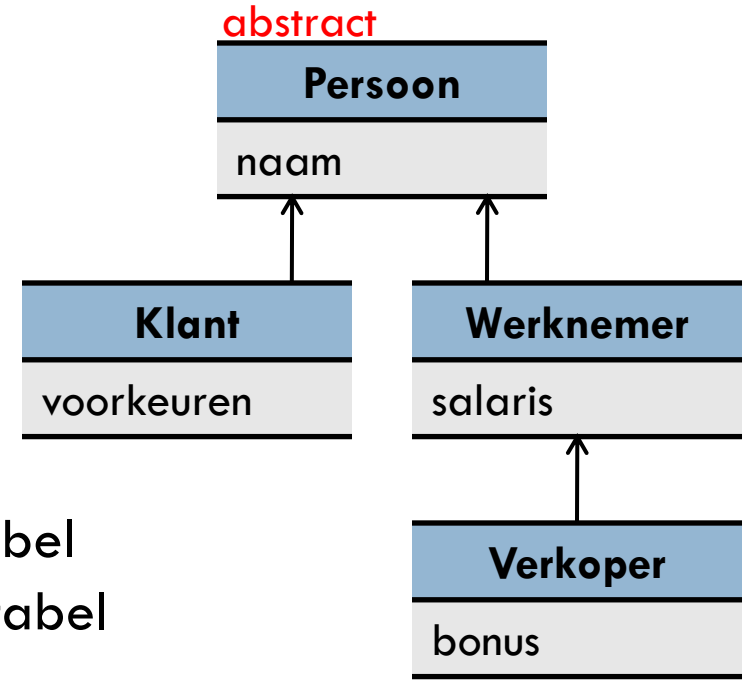
- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

Object ↔ Tabel ??

- Klasse → tabel
  - Vaak, maar niet altijd!
    - Soms anders o.w.v. performantie

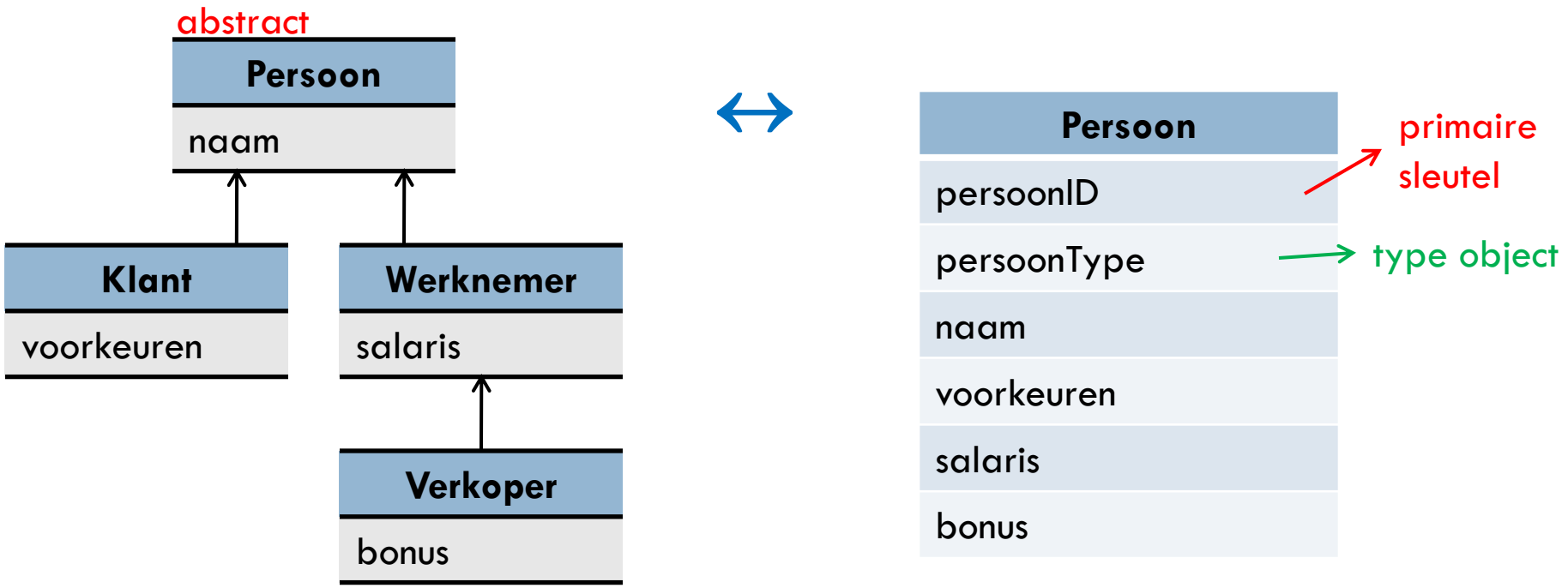
Overerving

- Hoe vertaal je overerving naar een gegevensbank?



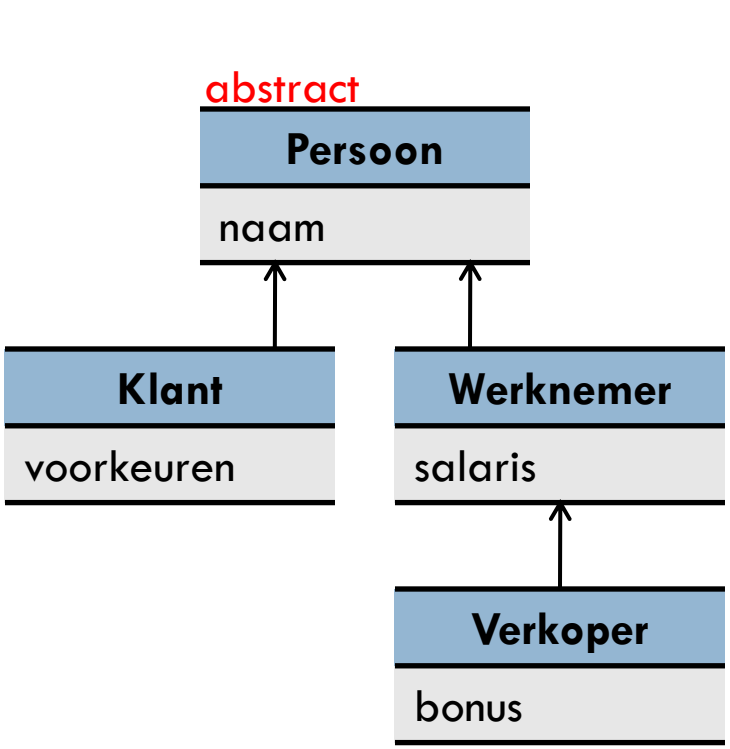
- Mogelijkheden
  - Volledige hiërarchie naar één tabel
  - Elke concrete klasse naar eigen tabel
  - Elke klasse naar een tabel
  - Alle klassen naar een generieke tabel

Volledige hiërarchie naar één tabel

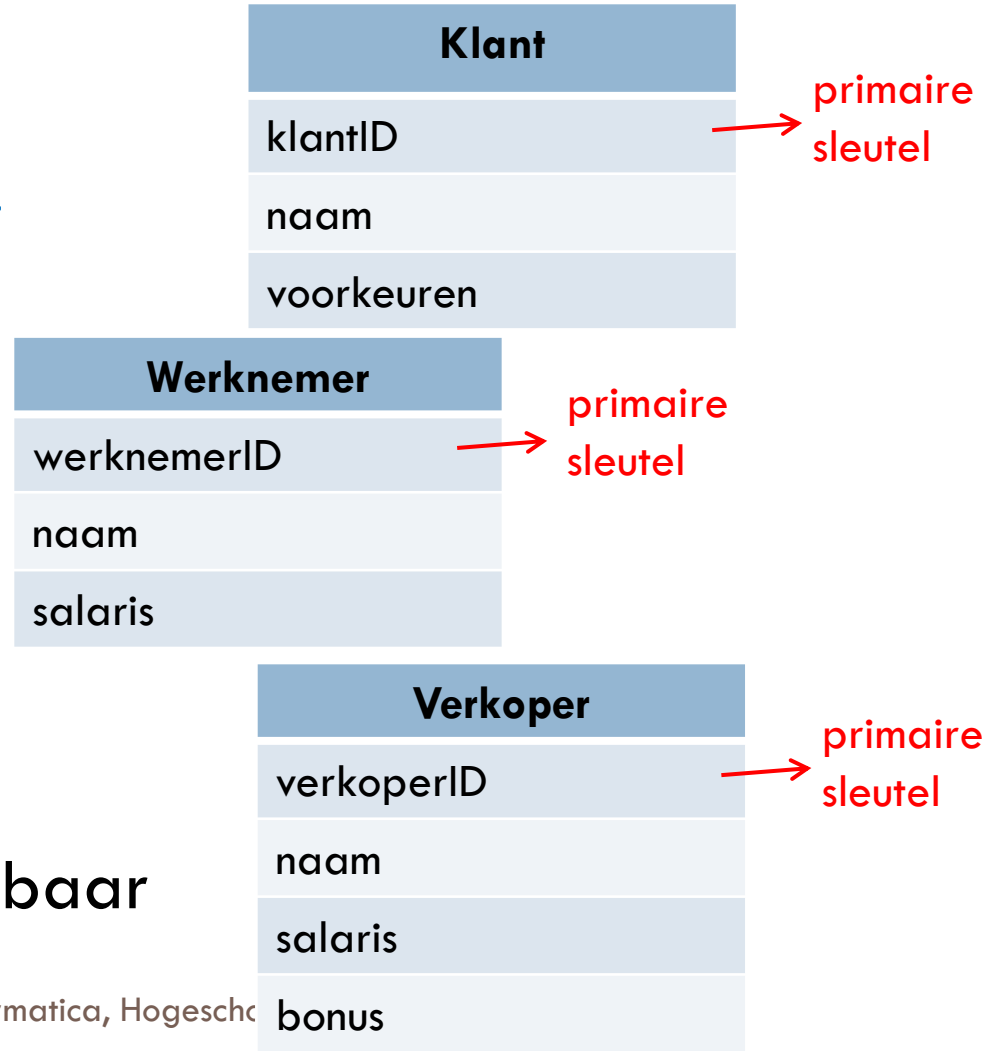


Nadeel: Moeilijk uitbreidbaar

Elke concrete klasse naar eigen tabel

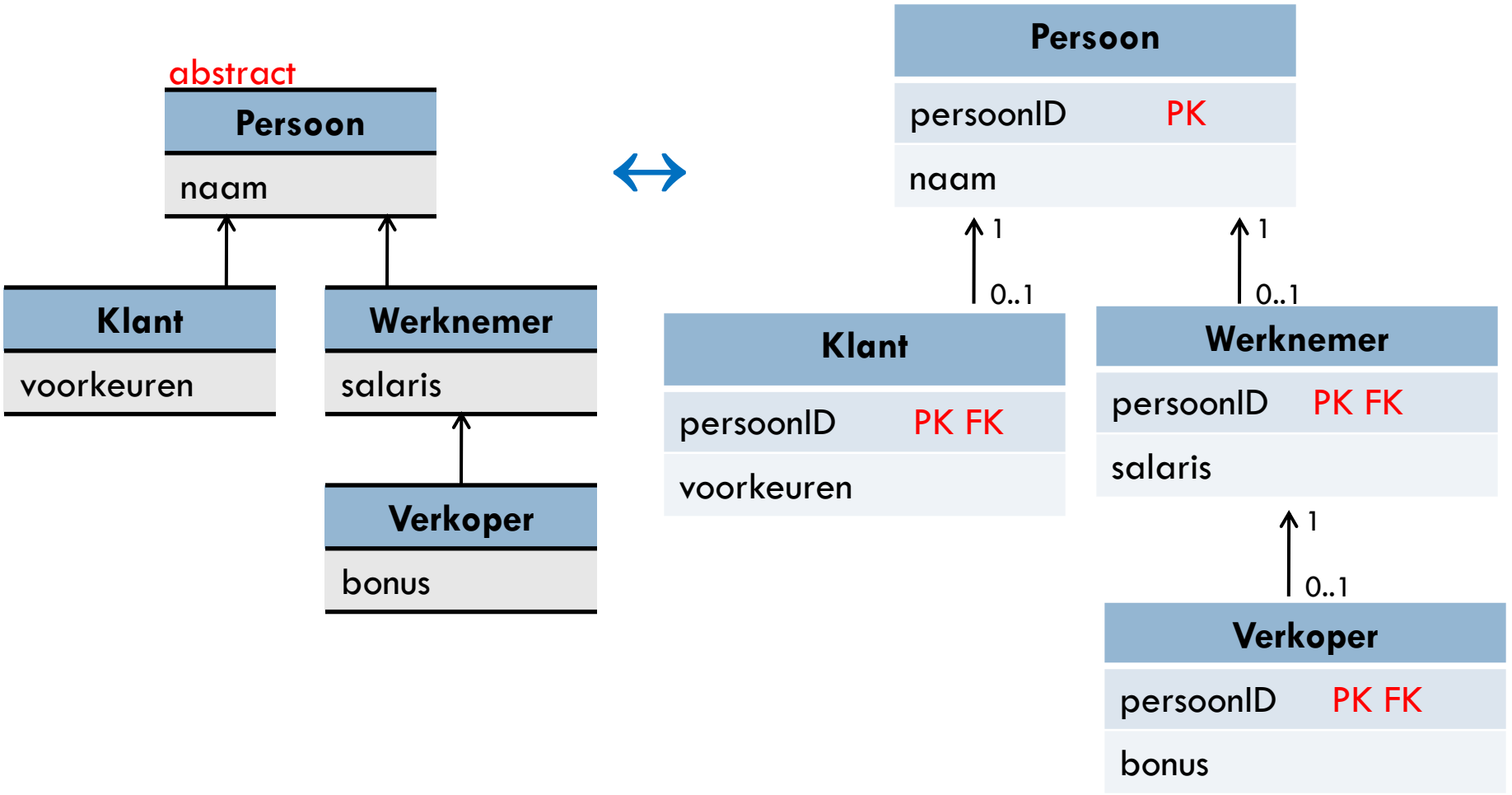


↔

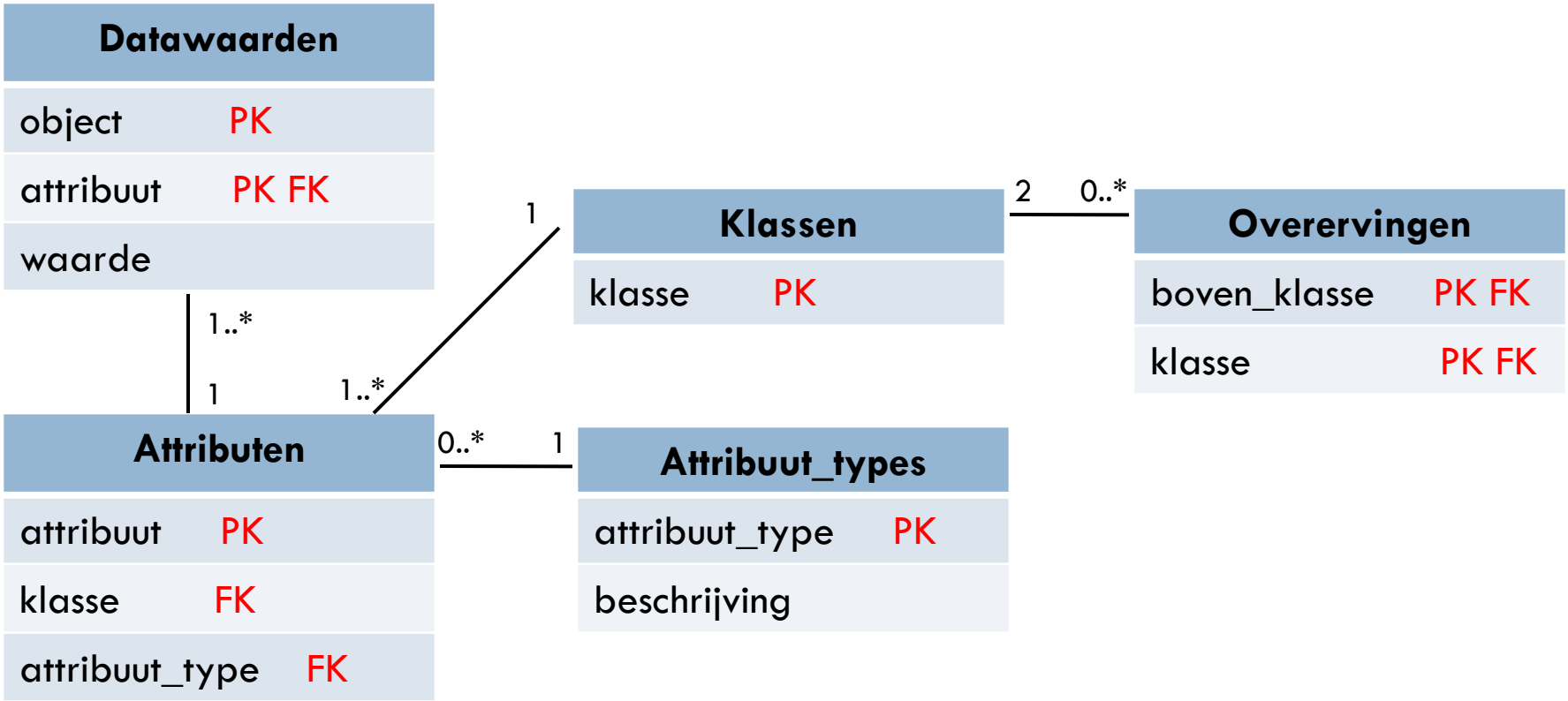


Nadeel: Moeilijk aanpasbaar

Elke klasse naar een tabel

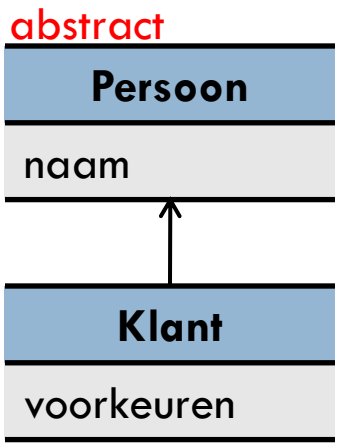


Alle klassen naar een generieke tabel



Gebruik: complexe applicaties met weinig data

Alle klassen naar een generieke tabel



Objecten

object	attribuut	waarde
OBJ_1	naam	Peeters
OBJ_1	voorkeuren	…

Klassen

klasse
Persoon
Klant

Attributen

attribuut	klasse	attribuut_type
naam	Persoon	String
voorkeuren	Klant	VoorkeurType

Overervingen

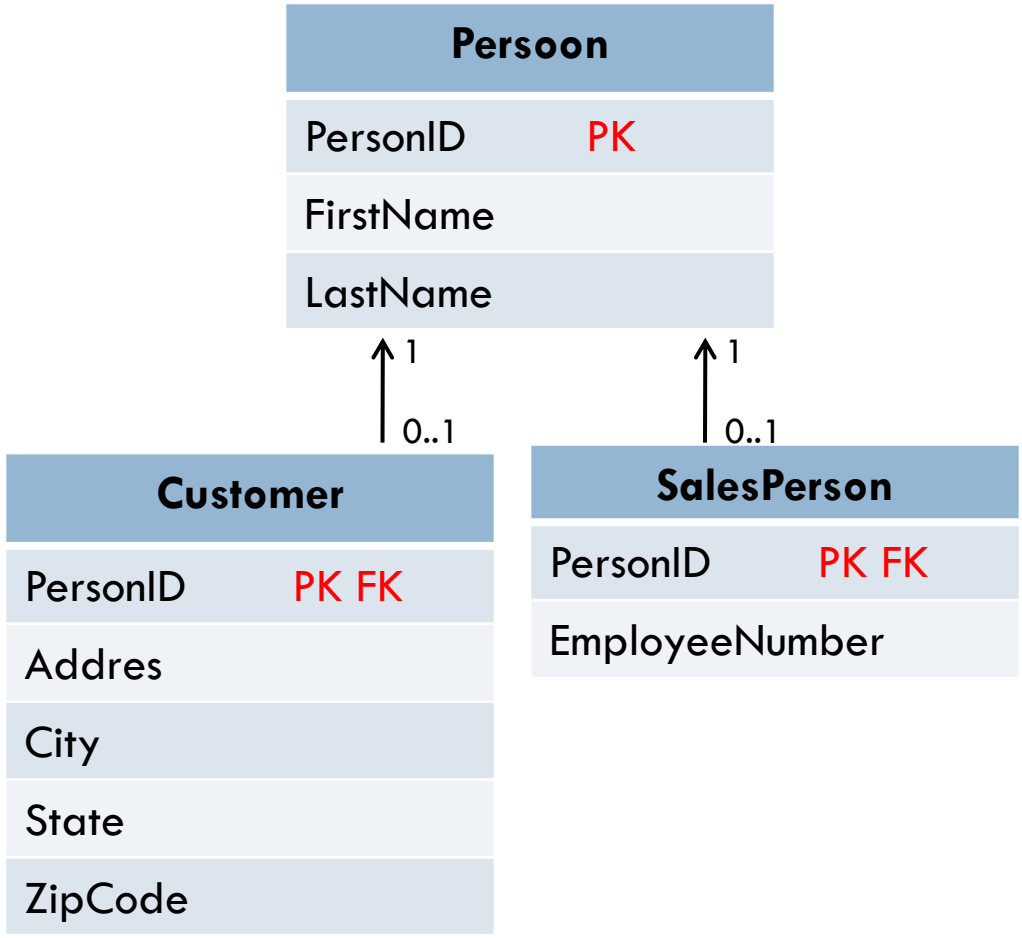
boven_klasse	klasse
Persoon	Klant

Attribuut_types

attribuut_type	beschrijving
String	…
VoorkeurType	…

Voorbeeld overerving


```
public partial class Person {
   private int? _personID;
   private string _firstName;
   private string _lastName;
   // Property Definitions...
}
public partial class Customer : Person {
   private string _address;
   private string _city;
   private string _state;
   private string _zipCode;
 // Property Definitions...
}
public partial class SalesPerson : Person {
   private string _employeeNumber;
   // Property Definitions...
}
```



Voorbeeld overerving


```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <class name="Person, __Code" table="Person">
    <id name="PersonID" column="PersonID" type="Int32“    unsaved-
    value="null">
      <generator class="native" />
    </id>
    <property name="FirstName" column="FirstName" type="String"/>
    <property name="LastName"  column="LastName"
    type="String"/>

    <joined-subclass name="Customer, __Code" table="Customer">
      <key column="PersonID"/>
      <property name="Address" column="Address" type="String"/>
      <property name="City"    column="City"    type="String"/>
      <property name="State"   column="State"   type="String"/>
      <property name="ZipCode" column="ZipCode" type="String"/>
    </joined-subclass>

    <joined-subclass name="SalesPerson, __Code“
      table="SalesPerson">
        <key column="PersonID"/>
        <property name="EmployeeNumber"
      column="EmployeeNumber“  type="String"/>
      </joined-subclass>
  </class>
</hibernate-mapping>
```

Inhoud

- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

Relaties

- Object relaties
  - Associatie
  - Aggregatie
  - Compositie
- Indeling volgens multipliciteit
  - 1-op-1
  - 1-op-veel
  - Veel-op-veel
- Indeling volgens richting
  - Eénrichtingsrelatie
  - Tweerichtingsrelatie

Onbestaand in DB ←

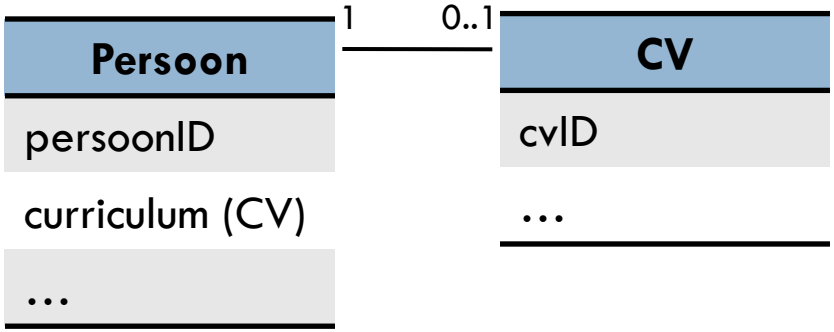
Relaties gerealiseerd

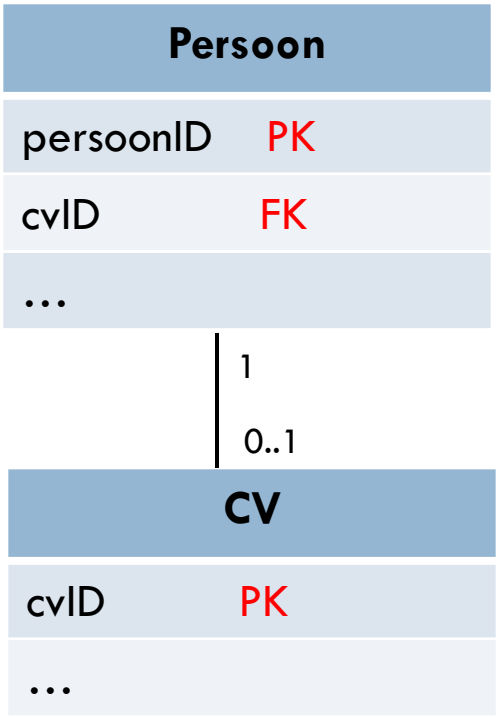
- In Objecten
  - Referentie naar object (enkelvoudige relatie)
    - getter en setter
  - Referentie naar een verzameling van objecten (meervoudige relatie)
    - getter en setter
    - add- en remove-methodes
  - In één object (éénrichtingsrelatie)
  - In beide objecten (tweerichtingsrelatie)
- In DB
  - Verwijssleutels (foreign keys)
  - 1-1 of 1-veel
    - Tabel met PK
    - Tabel met FK
  - Veel-veel
    - Associatietabel (FK, FK)
    - Twee tabellen met PK
  - Altijd tweerichtingsrelatie

Vertalen relaties

- Behoud de multipliciteit
  - 1-1
  - 1-veel
  - veel-veel

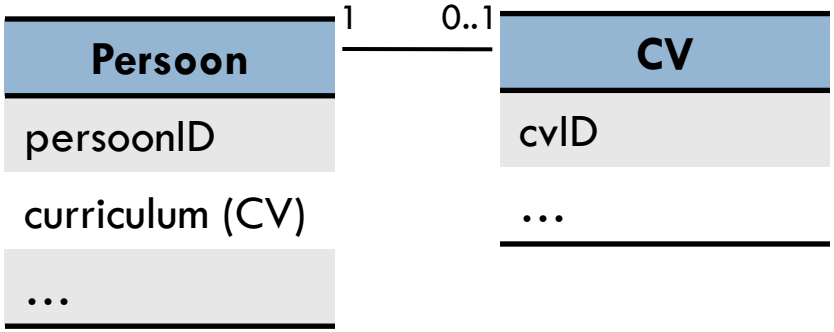
1-1 relatie ophalen

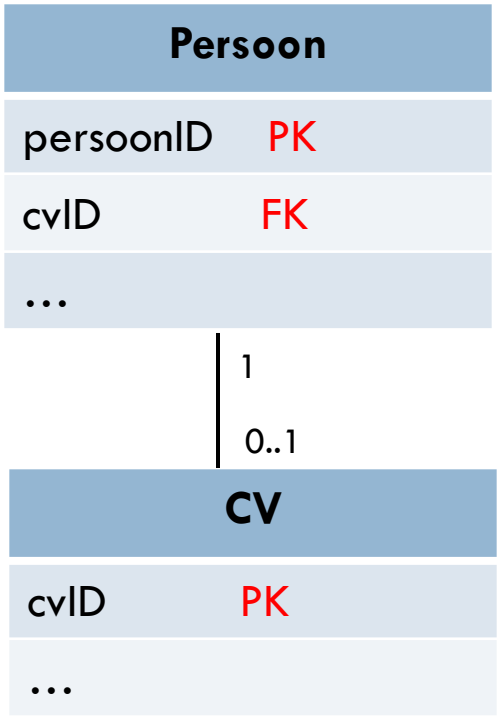




- Ophalen Persoon- en CV-object
  - Persoon-object inlezen
  - Kolom cvID gebruikt om CV te identificeren
  - Zoeken in tabel naar CV
  - Eventueel CV-object inlezen
  - (Indien tweerichtingsrelatie: omgekeerde referentie instellen)

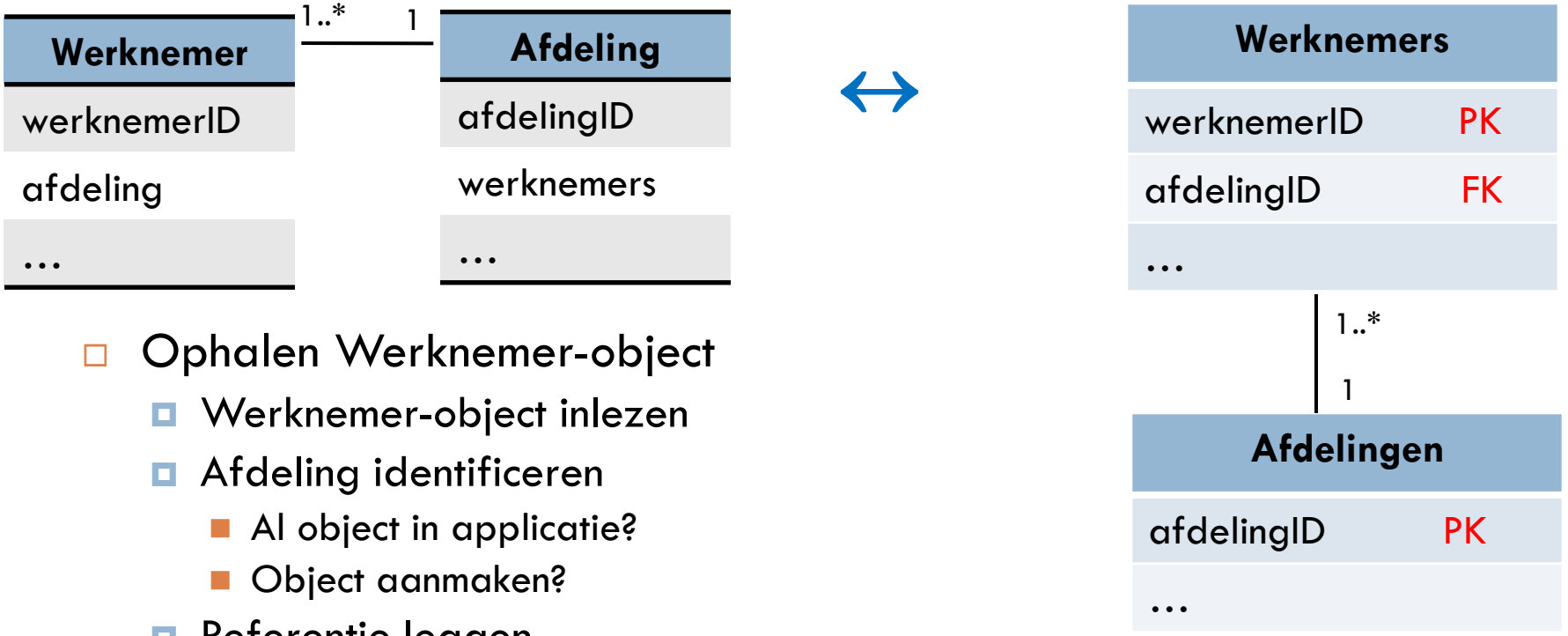
1-1 relatie bewaren





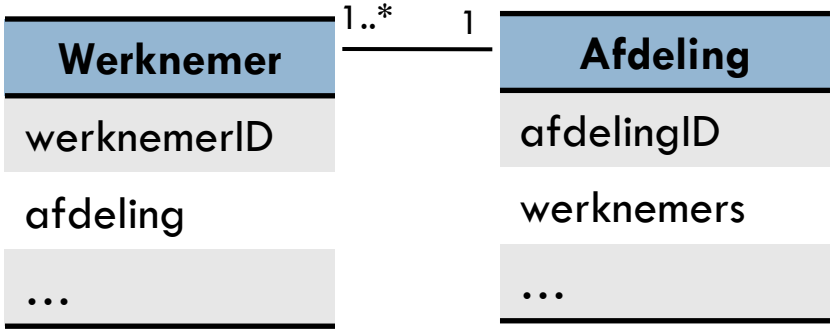
- Bewaren Persoon- en CV-object in DB
  - Start transactie
  - Bewaar beide objecten
    - Attributen
    - ID's
  - Commit
  - Beëindig transactie

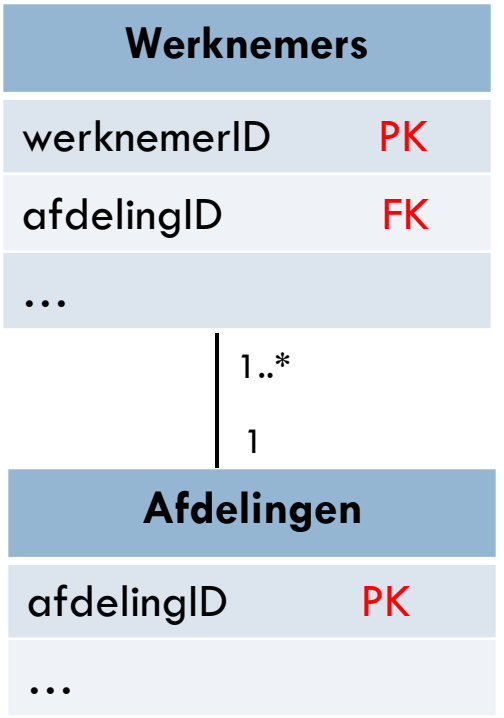
1-veel relatie ophalen



- Ophalen Werknemer-object
  - Werknemer-object inlezen
  - Afdeling identificeren
    - Al object in applicatie?
    - Object aanmaken?
  - Referentie leggen
  - Werknemer-object toevoegen aan Afdeling-object

1-veel relatie bewaren

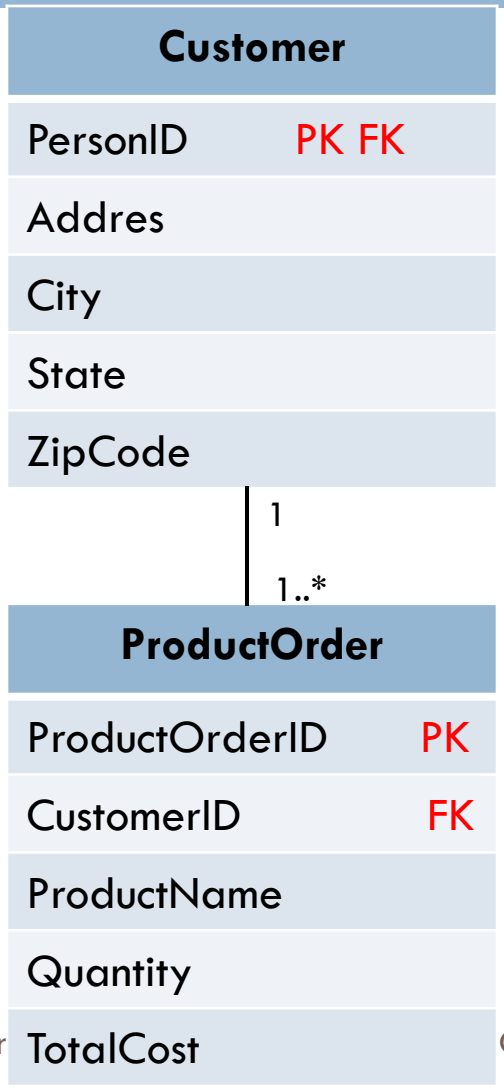




- Bewaren Werknemer en Afdeling-objecten in DB
  - Start transactie
  - Bewaar objecten
    - Attributen
    - ID's
  - Commit
  - Beëindig transactie

Voorbeeld 1-veel relatie

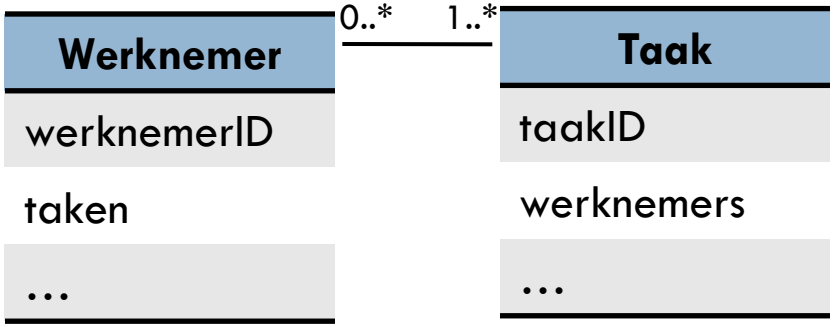

```
public partial class Customer : Person {
   private string _address;
   private string _city;
   private string _state;
   private string _zipCode;
   private ISet _orders;
 // Property Definitions...
}

public partial class ProductOrder {
   private int? _productOrderID;
   private int? _customerID;
   private string _productName;
   private int _quantity;
   private decimal _totalCost;
   // Property Definitions...
}
```

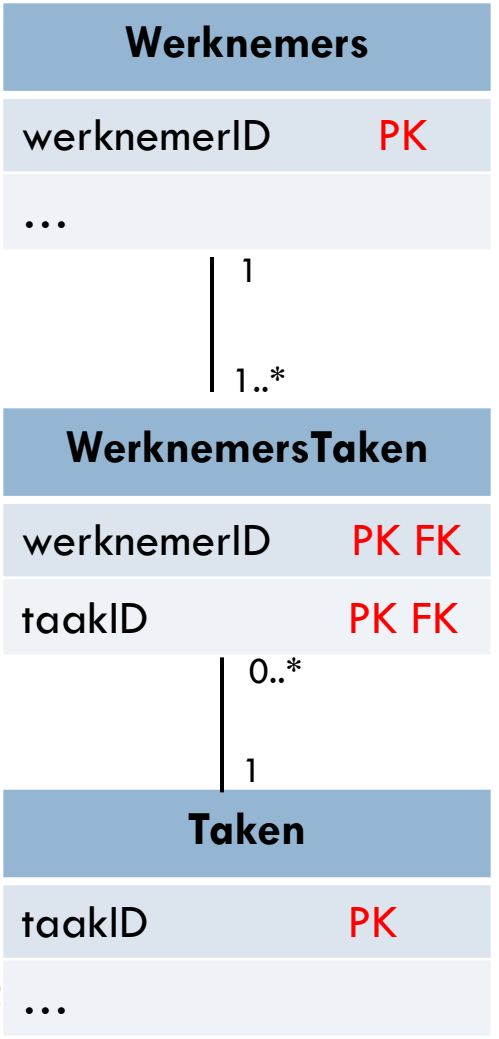



```
<joined-subclass name="Customer,
    __Code" table="Customer">
<set name="Orders">
    <key column="CustomerID"/>
    <one-to-many class="ProductOrder,
    __Code"/>
</set>
</joined-subclass>
```

- Beschrijving vertaling ProductOrder
  - Eigen bestand ProductOrder.hbm.xml

Veel-veel relatie ophalen





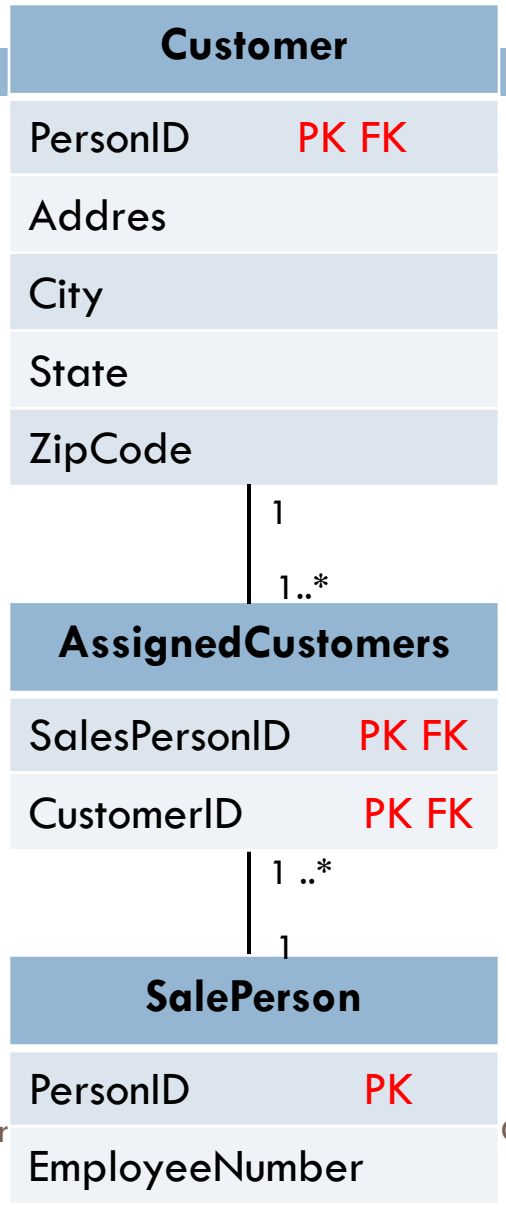
- Lijst taken ophalen voor Werknemer-object
  - SELECT-opdracht uitvoeren om taken te bepalen
  - Taak-object aanmaken of bepalen
    - Nieuw object?
    - Bestaand object? Vernieuwen?
  - Taak-object toevoegen aan verzameling
- Lijst werknemers ophalen voor Taak-object
  - analoog

Veel-veel relatie bewaren

- Taken van Werknemer-object bewaren
  - Start transactie
  - Update voor alle veranderde Taak-objecten
  - Insert voor alle nieuwe Taak-objecten
  - Delete voor alle verwijderde Taak-objecten (indien nodig)
  - Delete voor alle “WerknemersTaken” (indien nodig)
  - Delete voor alle taken die niet meer toegekend zijn aan de werknemer
  - Commit
  - Einde transactie

Voorbeeld veel-veel relatie


```
public partial class Customer : Person {
   private string _address;
   private string _city;
   private string _state;
   private string _zipCode;
   private ISet _orders;
 // Property Definitions...
}

public partial class SalesPerson : Person
{
   private string _employeeNumber;
   private ISet _assignedCustomers;

   // Property Definitions...
}
```




```
<joined-subclass name="SalesPerson,
     __Code" table="SalesPerson">
...
<set name="AssignedCustomers"
     table="AssignedCustomers"
     cascade="save-update">
   <key column="SalesPersonID"/>
   <many-to-many class="Customer,
     __Code" column="CustomerID"/>
</set>
</joined-subclass>
```

Bewaart ook Customer-objecten die toegevoegd zijn aan de Iset.

Standaard: enkel properties bewaard

Inhoud

- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

Objecten uitwisselen

- NHibernate
  - Vertaling vastleggen: xml-bestanden
  - Objecten ophalen en bewaren
    - Session Factory
    - Session

Session

- NHibernate.ISession
- Voert opdrachten uit
  - Ophalen, bewaren, aanpassen, verwijderen objecten in gegevensbank
- Bevat functionaliteit gegevensbank
- Bevat connectie met gegevensbank
- Light weight
  - Telkens aanmaken en vernietigen
  - Niet thread safe

Session Factory

- NHibernate.ISessionFactory
- Beheren Session-objecten
  - Aanmaken
  - …
- Één per applicatie
- Inlezen
  - Configuratie
  - Vertalingen (.hbm.xml-bestanden)
  - …
- Heavy weight
- Thread Safe

Voorbeeld aanmaak sessies

- Webapplicatie
  - Session-object
    - Aangemaakt bij start verwerken HTTP-aanvraag
    - Verwijderd bij doorsturen HTTP-antwoord
- NHibernateHelper.cs
  - Statische constructor
    - Aanmaken Session Factory
      - Configuratie inlezen
      - Object aanmaken
  - Metodes
    - Aanmaken Session-object
    - Sluiten Session-object
    - Afsluiten Session Factory

Inhoud

- Wat is ORM?
- Objecten afbeelden
  - Eigenschappen
  - Overerving
  - Relaties
- Nhibernate
  - Session en SessionFactory
  - Objecten ophalen en bewaren

Object ophalen (m.b.v. ID)


```
public static Customer getCustomer(int? customerID)  {
   // Open the session
   ISession session = NHibernateHelper.GetCurrentSession();

   // Load the order from the database
   Customer customer = (Customer)session.Load(typeof(Customer), customerID);

   return customer;
}
```

- Afsluiten sessie
  - Application_EndRequest (Global.asax)

```
void Application_EndRequest(object sender, EventArgs e)    {
       NHibernateHelper.CloseSession();
}
```

Object ophalen (a.d.v. criteria)


```
public static List<ProductOrder> getProductOrdersForCustomer(int? customerID)  {
   List<ProductOrder> orders = new List<ProductOrder>();

   // Get the session
   ISession session = NHibernateHelper.GetCurrentSession();

   // Load the order from the database
   ICriteria criteria = session.CreateCriteria(typeof(ProductOrder));
   criteria.Add(Expression.Eq("CustomerID", customerID));
   criteria.List(orders);

   return orders;
}
```

- Opbouwen WHERE-clausule
- Andere opdrachten: <, >, tussen, …

Bewaren of aanpassen object


```
public static void saveOrUpdateCustomer(Customer customer) {
  ISession session = NHibernateHelper.GetCurrentSession();
  ITransaction transaction = null;

  try   {
    // Start transaction
    transaction = session.BeginTransaction();

    // Save or update (saves if id is null; otherwise, updates)
    session.SaveOrUpdate(customer);

    // Commit transaction
    transaction.Commit();
  }   catch (Exception ex)   {
    transaction.Rollback();
    throw ex;
  }
}
```

- SaveOrUpdate
  - Programmeur hoeft operatie niet te kiezen
  - Save, Update kan ook
- Delete
- Gebruik steeds transacties
  - Achter schermen vaak meerdere opdrachten

Lazy Loading

- Attribuut: verzameling
  - Wanneer geladen?
- Aanmaken Customer-object
  - Enkel informatie uit Customer-tabel
    - address, city, state, zipCode
  - Proxy-object voor orders
    - Geladen als het opgevraagd wordt
- Kan enkel als de sessie nog actief is
- Kan uitgeschakeld worden in de .hbm.xml-bestanden

```
public partial class Customer : Person {
   private string _address;
   private string _city;
   private string _state;
   private string _zipCode;
   private ISet _orders;
 // Property Definitions...
}

public partial class SalesPerson : Person
{
   private string _employeeNumber;
   private ISet _assignedCustomers;
   // Property Definitions...
}
```

Meer informatie

- Mapping Objects to Relational Databases: O/R Mapping In Detail (http://www.agiledata.org/essays/mappingObjects.html)
- Using NHibernate as an ORM Solution for .NET (http://www.developer.com/net/asp/article.php/10917_3709346_6)
- Nhibernate (http://www.theserverside.net/tt/articles/showarticle.tss?id=NHibernate)
- Nhibernate – Part 2 (http://www.theserverside.net/tt/articles/showarticle.tss?id=NHibernateP2)

Meer informatie

- Your first NHibernate based application (http://blogs.hibernatingrhinos.com/nhibernate/archive/2008/04/01/your-first-nhibernate-based-application.aspx)
- Nhiberate als ORM oplossing (http://mark.mymonster.nl/wp-content/uploads/2007/07/nhibernate.pdf)
- NHibernate Quick Start Guide (http://www.hibernate.org/362.html)