Konzepte der Datenbanktechnologie

Prof. Dr. U. Hoffmann
FH Wedel

Hibernate 4
Objekt–Relationales–Mapping 2

Gliederung

Mapping von Entity–Beziehungen

1:N–Assoziationen

Entity vs. Value

Beispiel

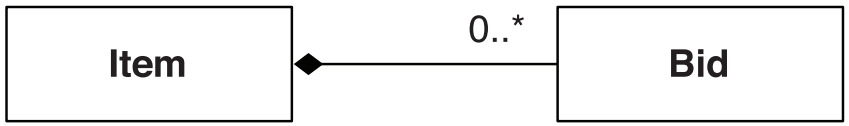
Kunden
Mietverträge
Fahrer
Punktekonto
Mietverträge/Fahrer

Mapping–Definitionen und Beispiele: *Java Persistence with Hibernate*, Bauer/King, 2007

1:N–Assioziationen

Kollektion mit Entity–Referenzen

- Sets, Bags oder Listen (ohne oder mit Reihenfolge)
- uni- oder bidirektional



Bisher betrachtetes Beispiel (Eltern/Kind–Beziehung):

- N:1–Assoziation (von `Bid` zu `Item`)
- inverse Assoziation realisiert als Java–`Set`

Statt `Set` jetzt `Bag` und `List`

1:N–Assoziationen

Beispiel (bidirektionale 1:N–Assoziation mit Bag)

```
<class name="Bid"
       table="BID">
    ...
    <many-to-one name="item"
                 column="ITEM_ID"
                 class="Item"
                 not-null="true"/>
</class>
<class name="Item"
       table="ITEM">
    ...
    <bag name="bids"
         inverse="true">
        <key column="ITEM_ID"/>
        <one-to-many class="Bid"/>
    </bag>
</class>
```

1:N–Assoziationen

Beispiel (bidirektionale 1:N–Assoziation mit Bag)

```
public class Item {
    ...
    private Collection bids = new ArrayList();
    public void setBids(Collection bids) {
        this.bids = bids;
    }
    public Collection getBids() {
        return bids;
    }
    public void addBid(Bid bid) {
        bid.setItem(this);
        bids.add(bid);
    }
    ...
}
```

Vorteil: keinen Aufwand für

- die Einhaltung der Reihenfolge und
- die Prüfung auf doppelte Elemente

unidirektionale 1:N–Assoziationen

Beispiel (unidirektionale 1:N–Assoziation mit Liste)

```
<class name="Item"
       table="ITEM">
    ...
    <list name="bids">
        <key column="ITEM_ID" not-null="true"/>
        <list-index column="BID_POSITION"/>
        <one-to-many class="Bid"/>
    </list>
</class>
```

- unidirektional: kein `inverse`
- Collection–Tabelle mit zusätzlicher Index–Spalte für Reihenfolge

BID

BID_ID	ITEM_ID	BID_POSITION	AMOUNT	CREATED_ON
1	1	0	99.00	19.04.08 23:11
2	1	1	123.00	19.04.08 23:12
3	2	0	433.00	20.04.08 09:30

unidirektionale 1:N–Assoziationen

Beispiel (unidirektionale 1:N–Assoziation mit List)

```
public class Item {
    ...
    private List<Bid> bids = new ArrayList();
    public void setBids(List<Bid> bids) {
        this.bids = bids;
    }
    public List<Bid> getBids() {
        return bids;
    }
    ...
}
```

bidirektionale 1:N–Assoziationen

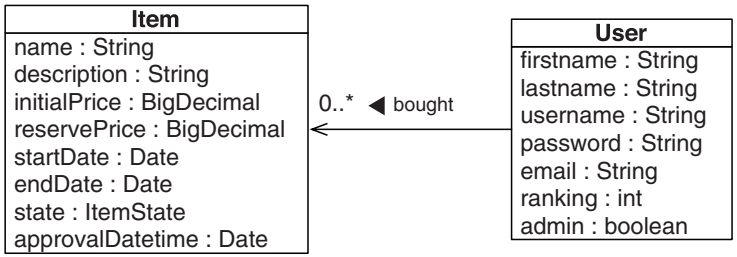
Welches ist die **inverse**–Assoziation?

- Hibernate ignoriert den Zustand der **inverse**–Assoziation
- 1:N-Assoziation ist `Set` oder `Bag`:
  1:N–Assoziation (Kollektion) ist **inverse**
- 1:N-Assoziation ist `List` oder `Map`:
  N:1–Assoziation ist “inverse”.
  statt **inverse**–Attribut: **insert,update**
  1:N–Assoziation darf nicht “inverse” sein, da die Kollektion zu sichernde Informationen enthält (Reihenfolge, Keys)

Beispiel (bidirektionale 1:N–Assoziation mit Liste)

```
<class name="Bid"
       table="BID">
    ...
    <many-to-one name="item"
                 column="ITEM_ID"
                 class="Item"
                 not-null="true"
                 insert="false"
                 update="false"/>
</class>
```

Optionale 1:N–Assoziationen

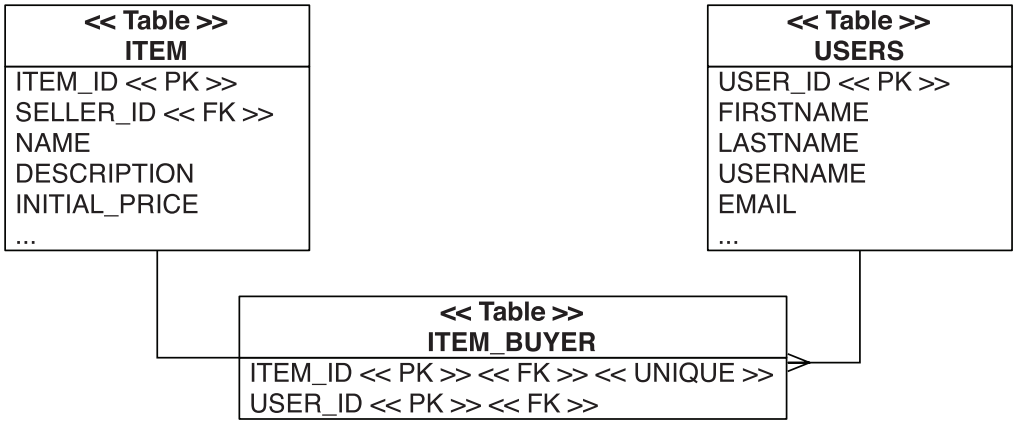


- optionale 1:N–Assoziation zwischen `User` und `Item`
- Anders als bei `Bid`, die immer einem `Item` zugeordnet sind
- ist ein `Item` nicht unbedingt einem `User` zugeordnet.

- Realisierung: Feld `buyer` in `Item`
- Feld `buyer` kann `null`–Wert annehmen.

- Alternative: Join-Tabelle

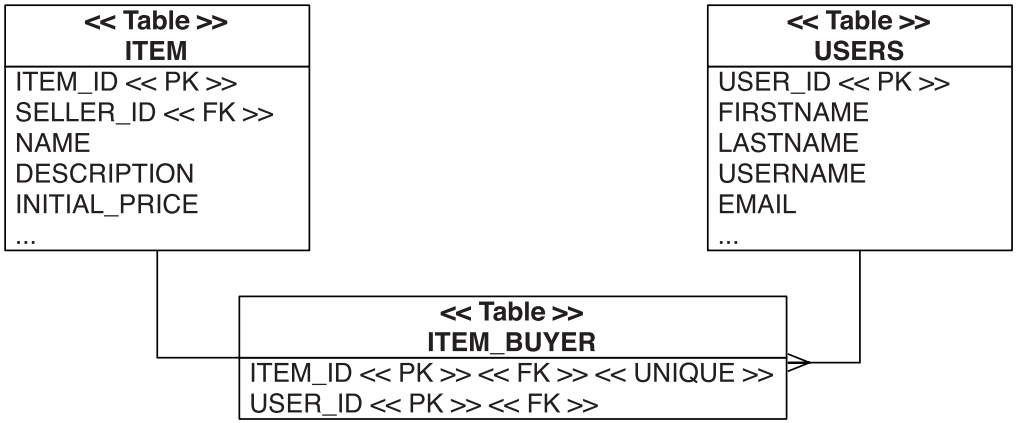
Optionale 1:N–Assoziationen



Beispiel (Optionale 1:N–Assoziation mit Join–Tabelle)

```
<class name="User" table="USERS">
...
<set name="boughtItems" table="ITEM_BUYER">
    <key column="USER_ID"/>
    <many-to-many class="Item"
                  column="ITEM_ID"
                  unique="true"/>
</set>
...
</class>
...
```

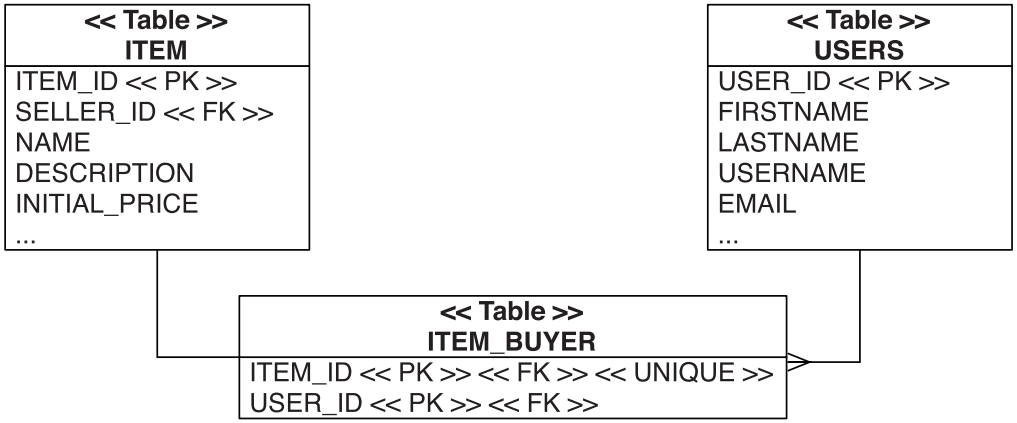
Optionale 1:N–Assoziationen



Beispiel (Optionale 1:N–Assoziation mit Join–Tabelle)

```
<class name="Item" table="ITEM">

<join table="ITEM_BUYER"
      optional="true"
      inverse="true">
  <key column="ITEM_ID" unique="true"
       not-null="true"/>
  <many-to-one name="buyer" column="USER_ID"/>
</join>
...
</class>
```

Optionale 1:N–Assoziationen



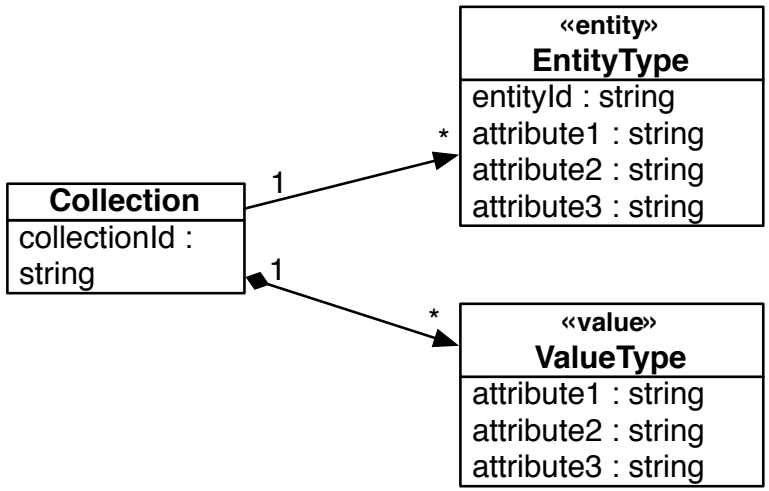
Beispiel (Herstellen der Assoziation im Programm)

```
aUser.getBoughtItems().add(anItem);
anItem.setBuyer(aUser);
```

Wahl der `inverse`–Assoziation:

- `<join>` ist `inverse`:
  Kollektion ist maßgeblich, `Item.buyer` wird ignoriert.
- Kollektion ist `inverse`:
  Kollektion darf nicht indiziert sein (Reihenfolge geht verloren)
  Kollektion wird ignoriert, `Item.buyer` ist maßgeblich.
- Java–Zugriffscode bleibt gleich.

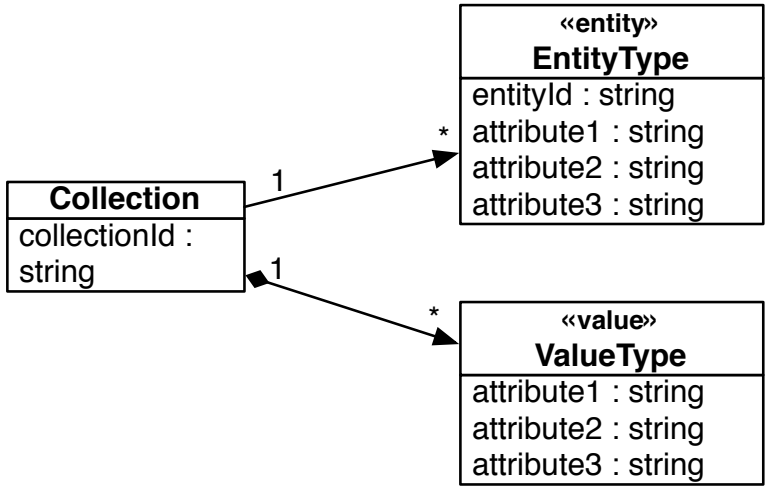
Unterschied zwischen Entitäts– und Wert–Kollektionen



Java

```
public class Collection {
    private String collectionId;
    private Set<EntityType> entities = new HashSet<EntityType>();
    private Set<ValueType> values = new HashSet<ValueType>();
    ...
}
```

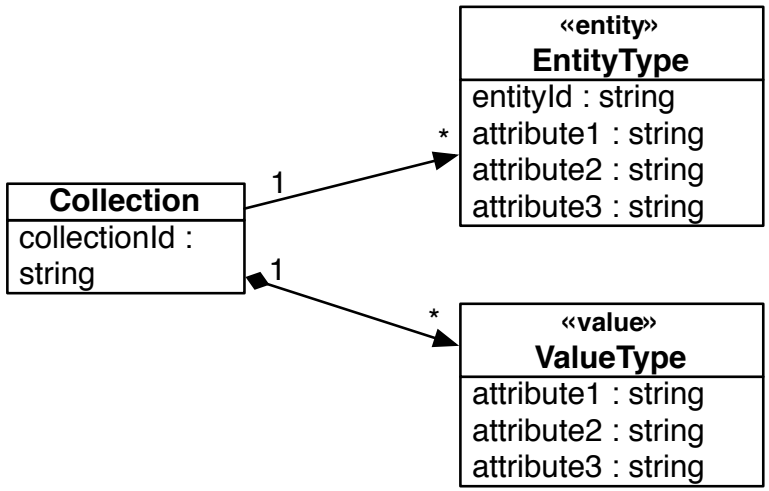
Unterschied zwischen Entitäts– und Wert–Kollektionen



Java

```
public class EntityType {

    private String entityId;

    private String attribute1;
    private String attribute2;
    private String attribute3;
    ...
}
```

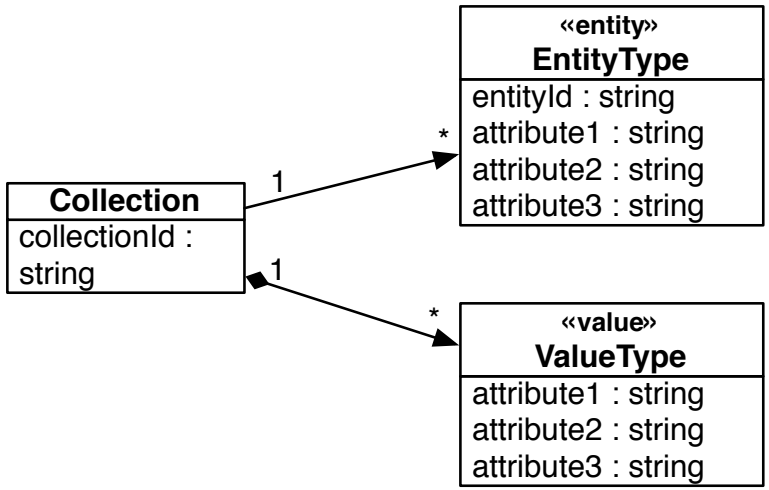
Unterschied zwischen Entitäts– und Wert–Kollektionen



Java

```
public class ValueType {
    private String attribute1;
    private String attribute2;
    private String attribute3;
    ...
}
```

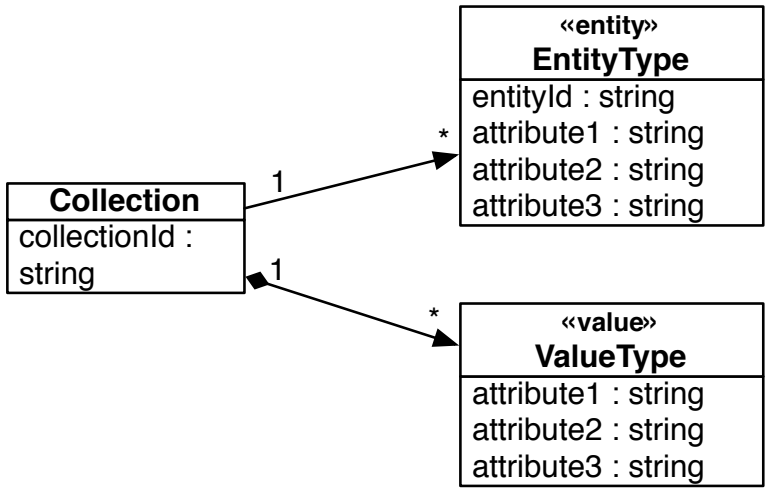
Unterschied zwischen Entitäts– und Wert–Kollektionen



Hibernate Mapping: EntityType

```
<hibernate-mapping package="de.fhwedel.oodb">
  <class name="EntityType">
      <id name="entityId" ><generator class="native"/></id>
      <property name="attribute1"/>
      <property name="attribute2"/>
      <property name="attribute3"/>
  </class>
</hibernate-mapping>
```

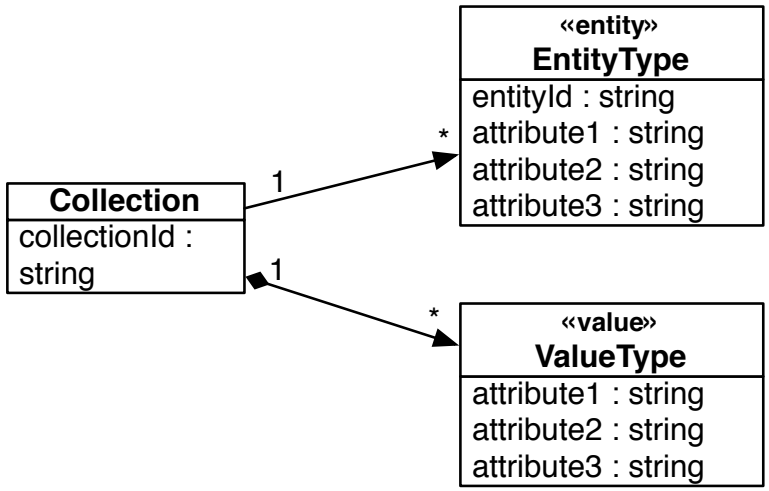
Unterschied zwischen Entitäts– und Wert–Kollektionen



Hibernate Mapping: Collection

```
<hibernate-mapping package="de.fhwedel.oodb">
  <class name="Collection">
    <id name="collectionId" ><generator class="native"/></id>

    <set name="entities"
         table="COLLECTION_ENTITYTYPE">
       <key column="collectionId"/>
       <one-to-many class="EntityType"/>
    </set>
    ...
  </class>
</hibernate-mapping>
```

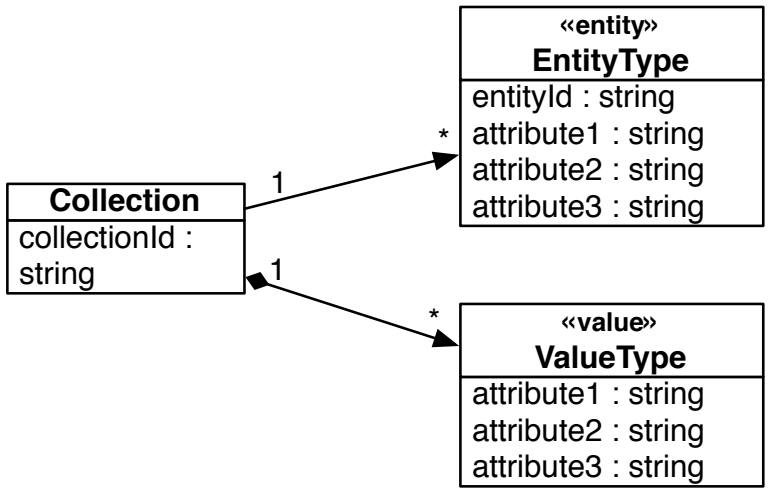
Unterschied zwischen Entitäts– und Wert–Kollektionen



Hibernate Mapping: Collection

```
<class name="Collection">
  <id name="collectionId" ><generator class="native"/></id>
  ...
  <set name="values" table="COLLECTION_VALUETYPE">
    <key column="collectionId"/>
    <composite-element class="ValueType">
      <property name="attribute1" not-null="true"/>
      <property name="attribute2" not-null="true"/>
      <property name="attribute3" not-null="true"/>
    </composite-element>
  </set>
</class>...
```

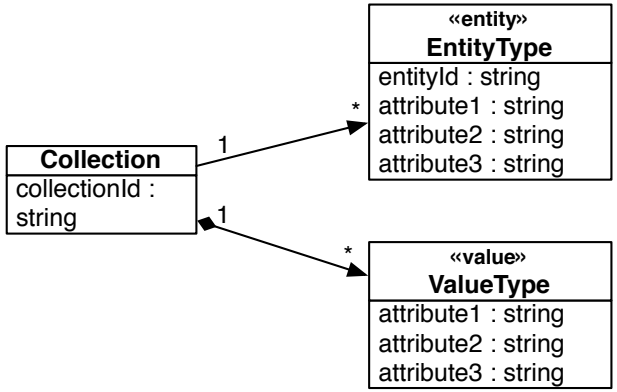
Resultierende Tabellen



SQL: Collection

```
create table Collection (
    collectionId varchar(255) not null auto_increment,
    primary key (collectionId)
) type=MyISAM
```

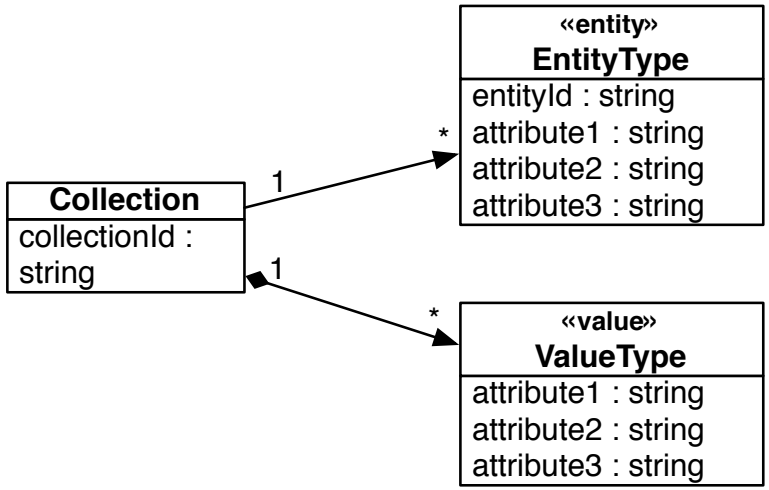
Resultierende Tabellen



SQL: EntityType

```
create table EntityType (
    entityId varchar(255) not null auto_increment,
    attribute1 varchar(255),
    attribute2 varchar(255),
    attribute3 varchar(255),
    collectionId varchar(255),
    primary key (entityId)
) type=MyISAM

alter table EntityType
    add index FK1E6D19DD44629127 (collectionId),
    add constraint FK1E6D19DD44629127
    foreign key (collectionId)
    references Collection (collectionId)
```

Resultierende Tabellen

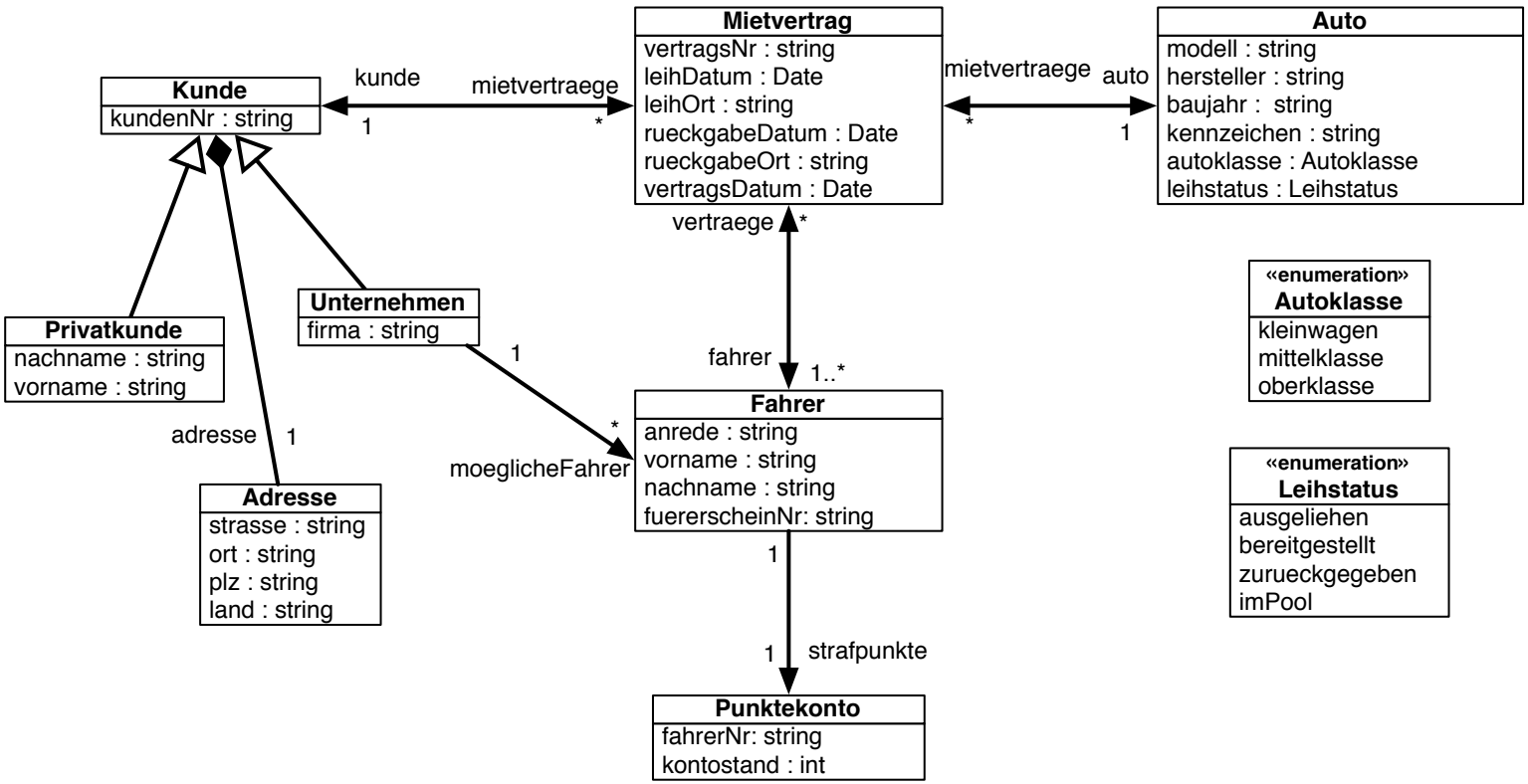


SQL: Collection–Tabellen

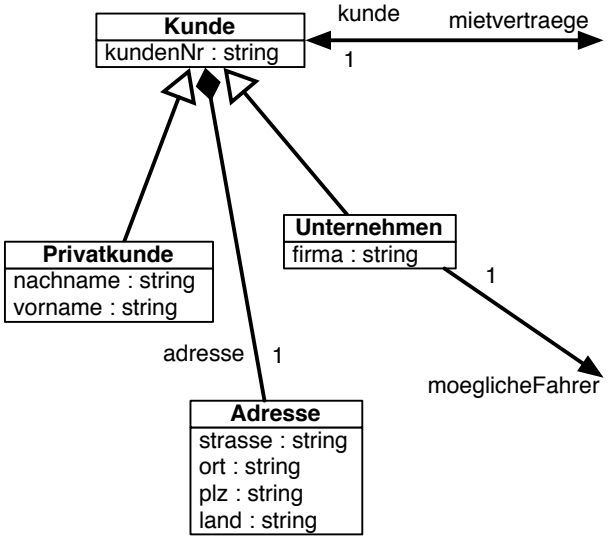
```
create table COLLECTION_VALUETYPE (
    collectionId varchar(255) not null,
    attribute1 varchar(255) not null,
    attribute2 varchar(255) not null,
    attribute3 varchar(255) not null,
    primary key (collectionId, attribute1, attribute2, attribute3)
) type=MyISAM

alter table COLLECTION_VALUETYPE
    add index FKB2D177AA44629127 (collectionId),
    add constraint FKB2D177AA44629127
    foreign key (collectionId)
    references Collection (collectionId)
```

Beispiel: RENTaC–Autovermietung

RENTaC — RENT a Car



RENTaC–Kunden — Implementierung in Java



Java

```
public class Adresse {
    private String strasse;
    private String ort;
    private String plz;
    private String land;
... }

public class Kunde {
    private String kundenNr;
    private Adresse adresse;
    private Set<Mietvertrag>
                mietvertraege;
... }
```

Java

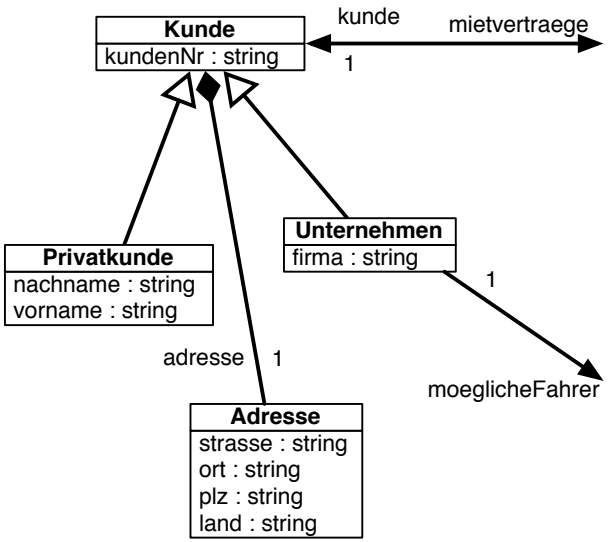
```
public class Privatkunde
       extends Kunde {
 private String nachname;
 private String vorname;
}
```

Java

```
public class Unternehmen
       extends Kunde {
 private String firma;
 private List<Fahrer> moeglicheFahrer;
}
```

RENTaC–Kunden — Mapping

Attribute von `Kunde` abbilden.



Hibernate Mapping: `hibernate.cfg.xml`

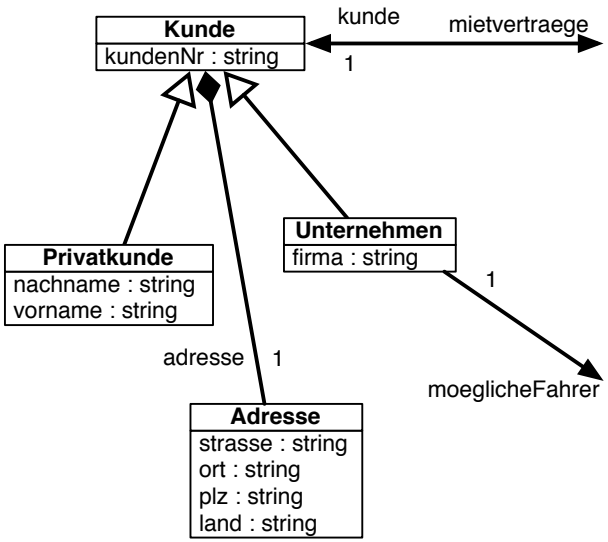
```
<hibernate-configuration>
  <session-factory>
  ...
    <mapping resource=
      ".../Kunde.hbm.xml" />
  ...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Kunde.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
  <class name="Kunde" table="kunde" >
    <id name="kundenNr" ><generator class="native"/></id>

    ...
  </class>
</hibernate-mapping>
```

RENTaC–Kunden — Mapping

Adresse als **Value-Typ** realisieren.



Hibernate Mapping: `hibernate.cfg.xml`

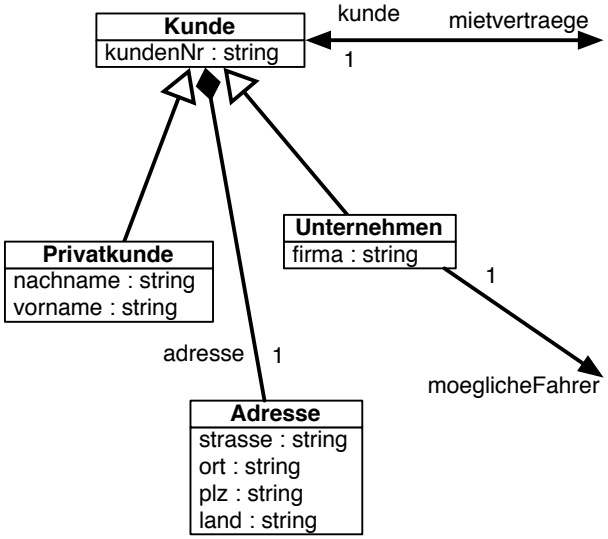
```
<hibernate-configuration>
  <session-factory>
  ...
    <mapping resource=
      ".../Kunde.hbm.xml" />
  ...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Kunde.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
 <class name="Kunde" table="kunde" >
 ...
   <component name="address" class="Adresse">
    <property name="strasse" column="str" not-null="true"/>
    <property name="ort" type="string" not-null="true"/>
    <property name="plz" not-null="true"/>
    <property name="land" not-null="true"/>
...</component>  </class>  </hibernate-mapping>
```

RENTaC–Kunden — Mapping

Viele gemeinsame Attribute: Table per Class Hierarchy



Hibernate Mapping: hibernate.cfg.xml

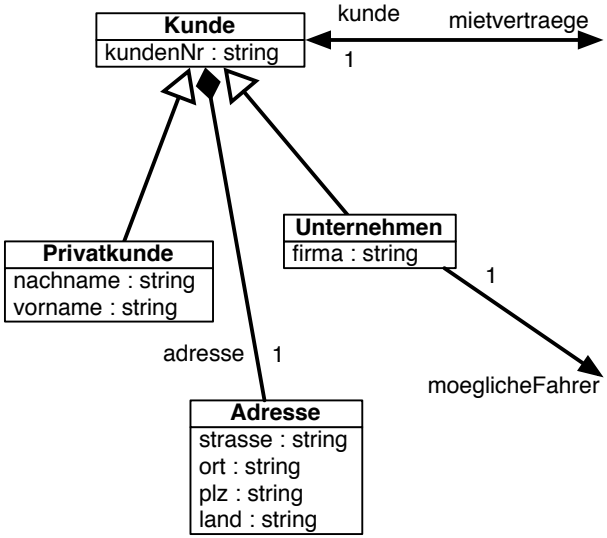
```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
      ".../Kunde.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: Kunde.hbm.xml

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
 <class name="Kunde" table="kunde" >
    ...
    <discriminator column="CUSTOMER_TYPE" type="string"/>
    ...
    <subclass name="Privatkunde" discriminator-value="PRIVAT">
      <property name="nachname" column="PRIVATKUNDE_NACHNAME"/>
      <property name="vorname" column="PRIVATKUNDE_VORNAME"/>
    </subclass> ...
 </class>
</hibernate-mapping>
```

RENTaC–Kunden — Mapping

Viele gemeinsame Attribute: Table per Class Hierarchy



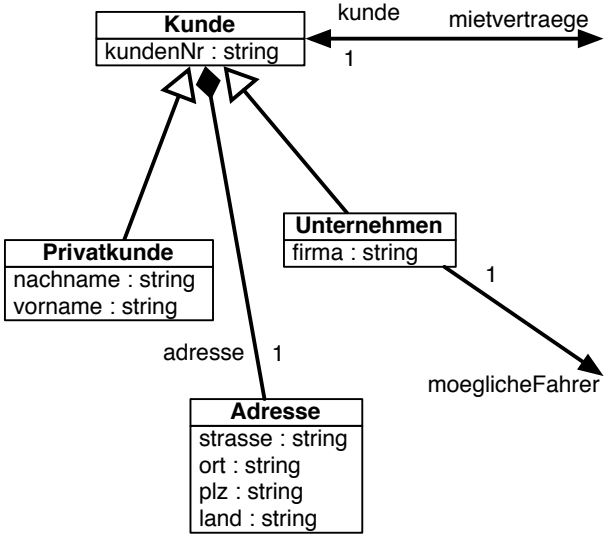
Hibernate Mapping: `hibernate.cfg.xml`

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
     ".../Kunde.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Kunde.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
 <class name="Kunde" table="kunde" >
    ...
    <subclass name="Unternehmen" discriminator-value="UNTERNEHMEN">
      <property name="firma" column="UNTERNEHMEN_FIRMA"/>
      ...
    </subclass>
 </class>
</hibernate-mapping>
```

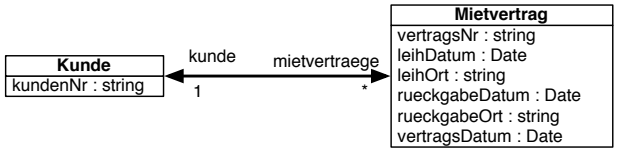
RENTaC–Kunden — Resultierende Tabelle (soweit)



SQL: Kunde

```
create table kunde (
      kundenNr varchar(255) not null auto_increment,
      CUSTOMER_TYPE varchar(255) not null,
      str varchar(255) not null,
      ort varchar(255) not null,
      plz varchar(255) not null,
      land varchar(255) not null,
      PRIVATKUNDE_NACHNAME varchar(255),
      PRIVATKUNDE_VORNAME varchar(255),
      UNTERNEHMEN_FIRMA varchar(255),
      primary key (kundenNr)
 ) type=MyISAM
```

RENTaC–Kunden — 1:N–Assoziation zu Mietverträgen



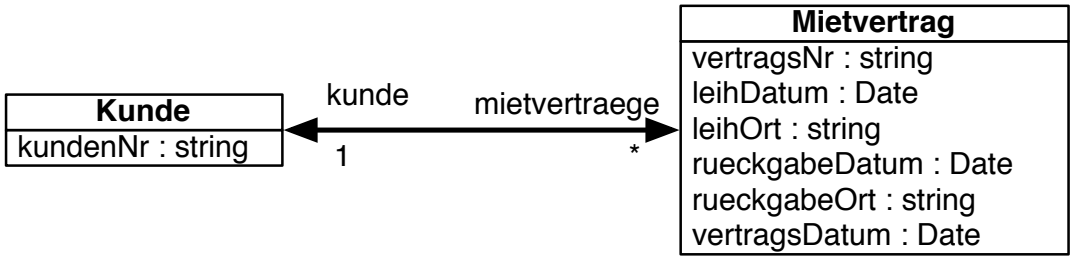
Hibernate Mapping: hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
  ".../Mietvertrag.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: Mietvertrag.hbm.xml

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
  <class name="Mietvertrag">
    <id name="vertragsNr" ><generator class="native"/></id>
    <property name="leihOrt"/>
    ...
    <many-to-one name="kunde" column="kundenNr"
                 class="Kunde" not-null="true"/>
    ...
  </class>
</hibernate-mapping>
```

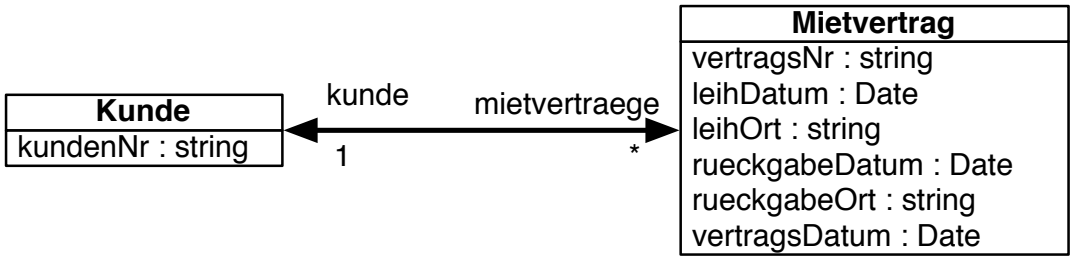
RENTaC–Mietvertrag — Resultierende Tabelle



SQL: Mietvertrag

```
create table Mietvertrag (
    vertragsNr varchar(255) not null auto_increment,
    leihOrt varchar(255),
    ...
    kundenNr varchar(255) not null,
    primary key (vertragsNr)
) type=MyISAM

alter table Mietvertrag
    add index FKA0E6FB7C70C53B4A (kundenNr),
    add constraint FKA0E6FB7C70C53B4A
    foreign key (kundenNr)
    references kunde (kundenNr)
```

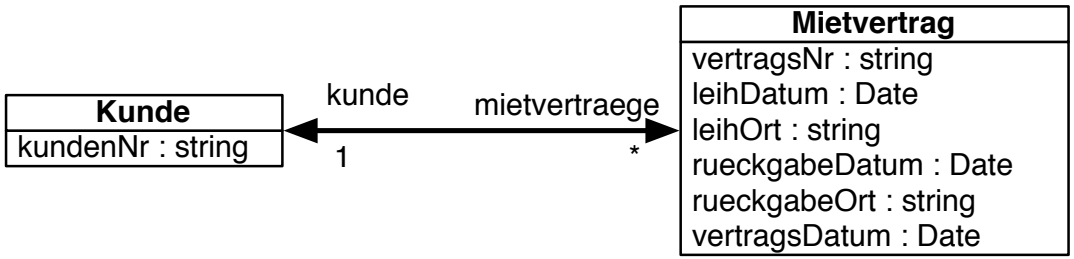
RENTaC–Mietvertrag — Accessor-Methoden



Java

```
public class Mietvertrag {

    private String vertragsNr;
    ...
    private Kunde kunde;

    public Kunde getKunde() {
        return kunde;
    }

    public void setKunde(Kunde kunde) {
        this.kunde = kunde;
        kunde.addMietvertrag(this);
    }
    ...
}
```

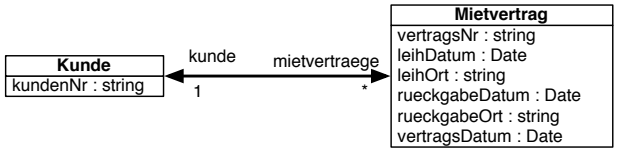
RENTaC–Kunden — Accessor-Methoden



Java

```
public class Kunde {
  private Set<Mietvertrag> mietvertraege =
      new HashSet<Mietvertrag>();
  ....
  public Set<Mietvertrag> getMietvertraege() {
      return mietvertraege;
  }
  ...
  public Set<Mietvertrag> addMietvertrag(Mietvertrag mv) {
      if (!mietvertraege.contains(mv)) {
        mv.setKunde(this);
        mietvertraege.add(mv);
      }
      return mietvertraege;
  }
...
}
```

RENTaC–Kunden — 1:N–Assoziation zu Mietverträgen



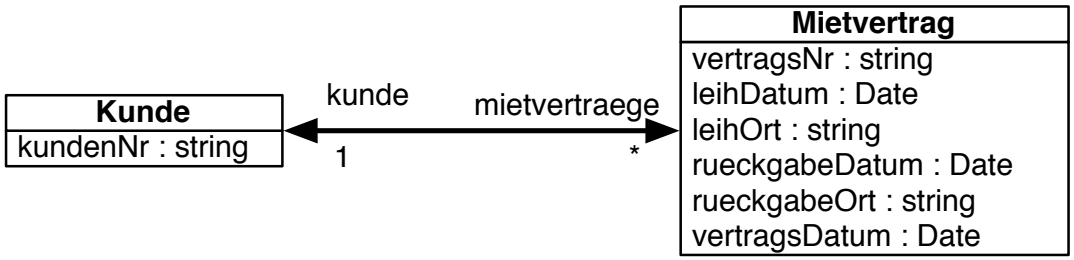
Hibernate Mapping: hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
      ".../Kunde.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: Kunde.hbm.xml

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
 <class name="Kunde" table="kunde" >
    ...
    <set name="mietvertraege" inverse="true">
        <key column="kundenNr"/>
        <one-to-many class="Mietvertrag"/>
    </set>
    ...
 </class>
</hibernate-mapping>
```

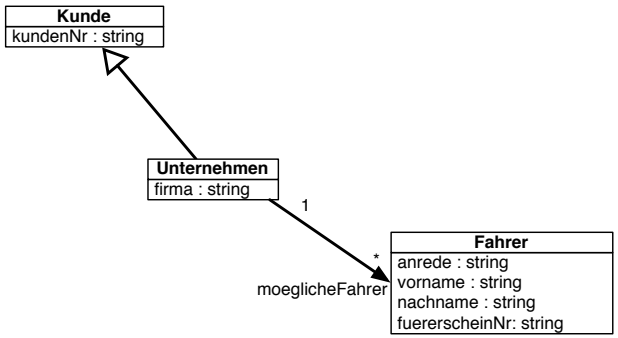
RENTaC–Mietvertrag — Resultierende Tabelle bleibt gleich



SQL: Mietvertrag

```
create table Mietvertrag (
    vertragsNr varchar(255) not null auto_increment,
    leihOrt varchar(255),
    ...
    kundenNr varchar(255) not null,
    primary key (vertragsNr)
) type=MyISAM

alter table Mietvertrag
    add index FKA0E6FB7C70C53B4A (kundenNr),
    add constraint FKA0E6FB7C70C53B4A
    foreign key (kundenNr)
    references kunde (kundenNr)
```

RENTaC–Unternehmen — 1:N–Assoziation zu Fahrern



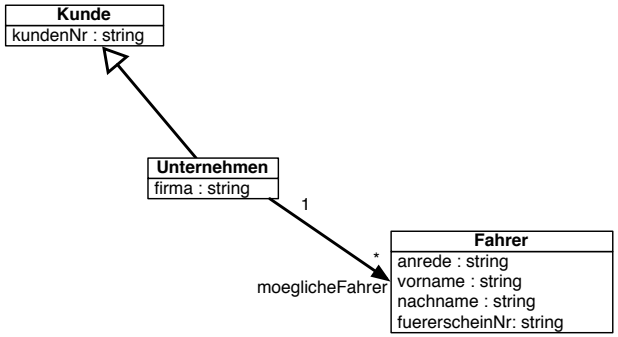
Hibernate Mapping: hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
  ".../Kunde.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: Kunde.hbm.xml

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
   <class name="Kunde">
     ...
     <subclass name="Unternehmen" discriminator-value="UNTERNEHMEN">
      <property name="firma" column="UNTERNEHMEN_FIRMA"/>
      <list name="moeglicheFahrer">
        <key column="kundenNr"/>
        <list-index column="fahrer_index">
        <one-to-many class="Fahrer"/>
      </list>
  ...</subclass>
...</class>
</hibernate-mapping>
```

RENTaC–Unternehmen — 1:N–Assoziation zu Fahrern



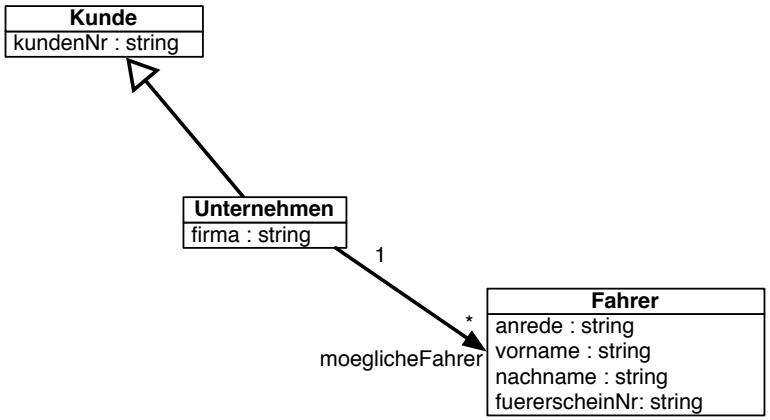
Hibernate Mapping: hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
  ".../Fahrer.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: Fahrer.hbm.xml

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
  <class name="Fahrer" table="fahrer" >
    <id name="fuehrerscheinNr" />
    <property name="anrede" type="string" />
    <property name="vorname" type="string" />
    <property name="nachname" type="string" />
    ...
  </class>
</hibernate-mapping>
```

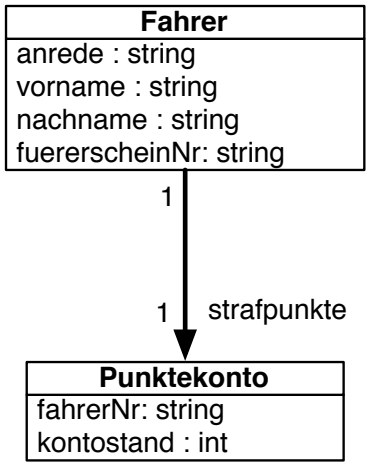
RENTaC–Fahrer — Resultierende Tabelle



SQL: Fahrer

```
create table fahrer (
     fuehrerscheinNr varchar(255) not null,
     anrede varchar(255),
     vorname varchar(255),
     nachname varchar(255),
     kundenNr varchar(255),
     fahrer_index integer,
     primary key (fuehrerscheinNr)
) type=MyISAM

alter table fahrer
     add index FKB396367299C9D12 (kundenNr),
     add constraint FKB396367299C9D12
     foreign key (kundenNr)
     references kunde (kundenNr)
```

RENTaC–Fahrer — 1:1–Assoziation zu Punktekonten



Hibernate Mapping: hibernate.cfg.xml

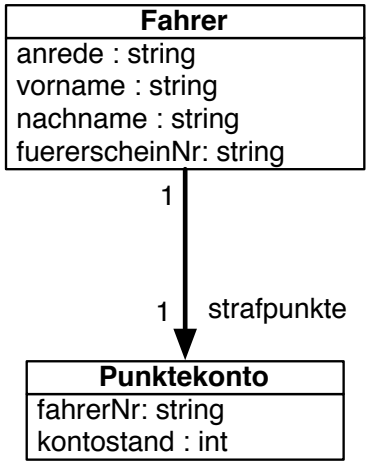
```
<hibernate-configuration>
  <session-factory>
  ... <mapping class=
      "... .Punktekonto" />...
  </session-factory>
</hibernate-configuration>
```

Java

```
@Entity
@Table (name="punktekonto")
public class Punktekonto {

    @Id
    private String fahrerNr;

    @Column
    private int kontostand;
...
}
```

RENTaC–Fahrer — 1:1–Assoziation zu Punktekonten

Mit eindeutigem Foreign-Key in `Fahrer` realisieren



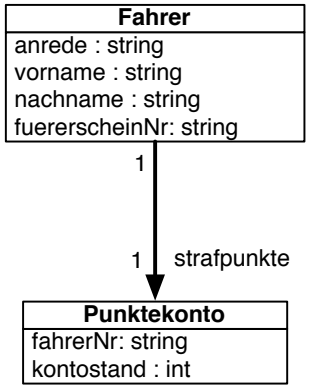
Hibernate Mapping: `hibernate.cfg.xml`

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
      ".../Fahrer.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Fahrer.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
  <class name="Fahrer" table="fahrer" >
    ...
    <many-to-one name="strafpunkte"
                 class="Punktekonto"
                 cascade="save-update"
                 unique="true"/>
    ...
  </class>
</hibernate-mapping>
```

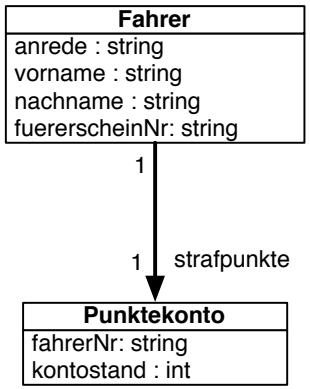
RENTaC–Fahrer — Resultierende Tabelle



SQL: Fahrer

```
create table punktekonto (
      fahrerNr varchar(255) not null,
      kontostand integer,
      primary key (fahrerNr)
 ) type=MyISAM
create table fahrer (
      fuehrerscheinNr varchar(255) not null,
      anrede varchar(255),
      vorname varchar(255),
      nachname varchar(255),
      strafpunkte varchar(255) unique,
      kundenNr varchar(255),
      fahrer_index integer,
      primary key (fuehrerscheinNr)
 ) type=MyISAM
 ...
```

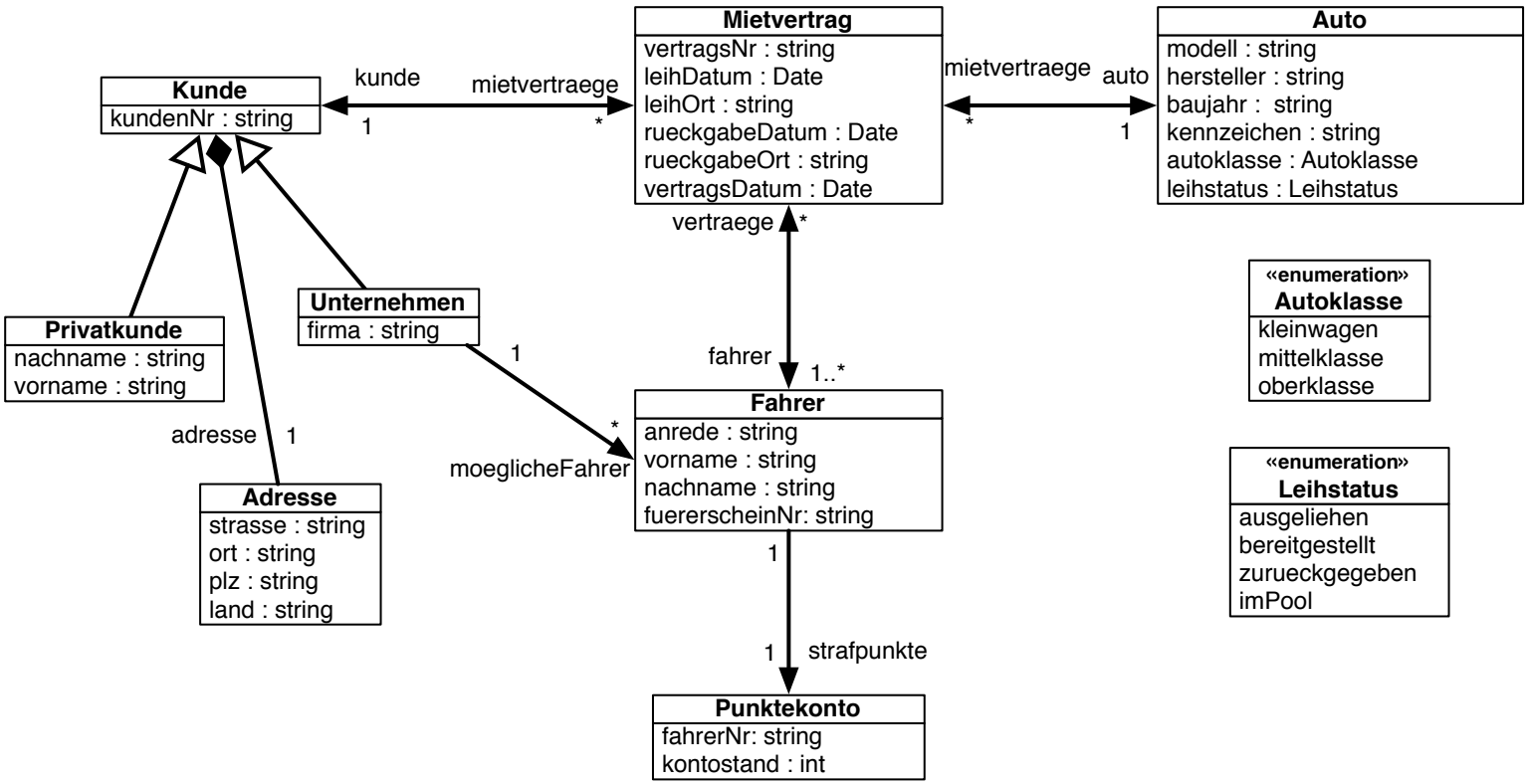
RENTaC–Fahrer — Resultierende Tabelle



SQL: Fahrer

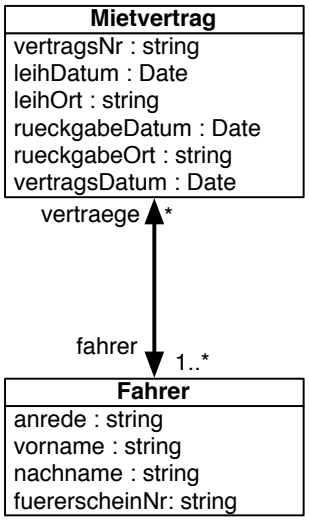
```
...
alter table fahrer
    add index FKB396367299C9D12 (kundenNr),
    add constraint FKB396367299C9D12
    foreign key (kundenNr)
    references kunde (kundenNr)

alter table fahrer
    add index FKB3963672962D55B3 (strafpunkte),
    add constraint FKB3963672962D55B3
    foreign key (strafpunkte)
    references punktekonto (fahrerNr)
```

RENTaC–Autovermietung



- 1:N–Assoziation zwischen Auto und Mietvertrag
  analog zu Kunde $\leftrightarrow$ Mietvertrag

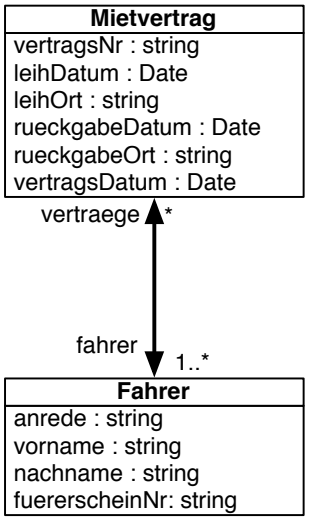
RENTaC–Fahrer — N:M–Assoziation zu Mietverträgen



Java

```
public class Mietvertrag {
   ...
   private Set<Fahrer> fahrer =
                    new HashSet<Fahrer>();

   public Set<Fahrer> getFahrer()
      return fahrer;
   }
   public void setFahrer(Set<Fahrer> fahrer) {
      this.fahrer = fahrer;
   }
   public void addFahrer(Fahrer einFahrer) {
      this.fahrer.add(einFahrer);
      einFahrer.add(this);
   }
   ...
}
```

RENTaC–Fahrer — N:M–Assoziation zu Mietverträgen

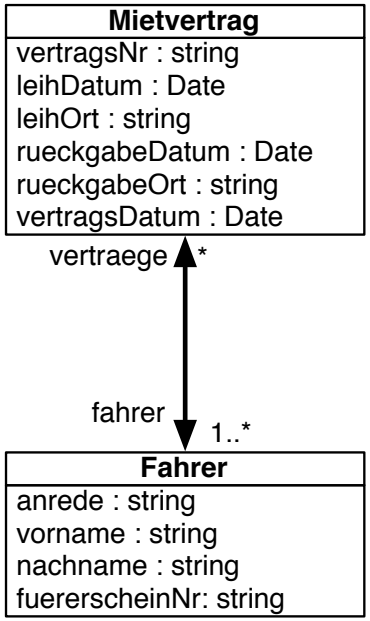


Java

```
public class Fahrer {
    ...
    private Set<Mietvertrag> vertraege =
          new HashSet<Mietvertrag>();

    public Set<Mietvertrag>
           getMietvertraege() {
       ...
    }

    public void setMietvertraege(
           Set<Mietvertrag> vertraege) {
       ...
    }
    ...
    public void addMietvertrag(
              Mietvertrag vertrag) {
       ...
    }
    ...
}
```

RENTaC–Fahrer — N:M–Assoziation zu Mietverträgen

Realisierung durch eine **Join-Tabelle**



Hibernate Mapping: `hibernate.cfg.xml`

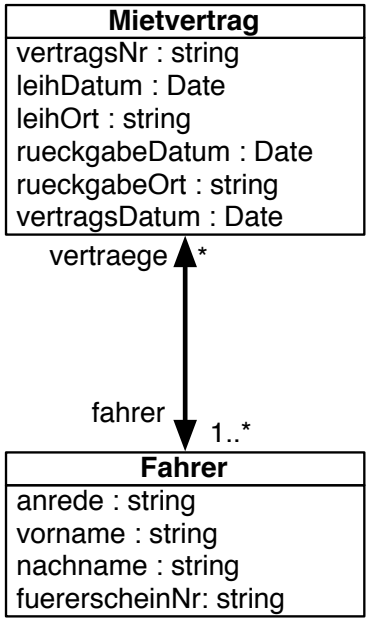
```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
     ".../Fahrer.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Fahrer.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
   <class name="Fahrer" table="fahrer" >
    ...
    <set name="vertrag"
         table="VERTRAG_FAHRER" cascade="save-update">
        <key column="fuehrerscheinNr"/>
        <many-to-many class="Mietvertrag" column="vertragsNr"/>
    </set>
...</class>
</hibernate-mapping>
```

RENTaC–Fahrer — N:M–Assoziation zu Mietverträgen

Realisierung durch eine **Join-Tabelle** — Rückrichtung



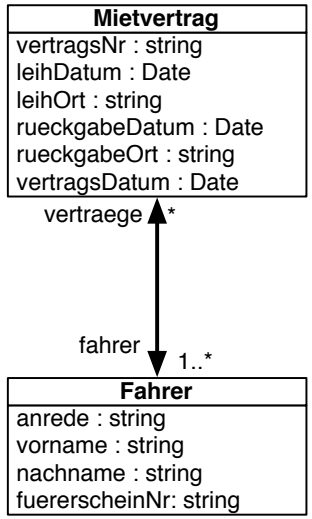
Hibernate Mapping: `hibernate.cfg.xml`

```
<hibernate-configuration>
  <session-factory>
  ... <mapping resource=
     ".../Mietvertrag.hbm.xml" />...
  </session-factory>
</hibernate-configuration>
```

Hibernate Mapping: `Mietvertrag.hbm.xml`

```
<hibernate-mapping package="de.fhwedel.oodb.rentac.model">
   <class name="Mietvertrag">
    ...
    <set name="fahrer" inverse="true"
         table="VERTRAG_FAHRER" cascade="save-update">
        <key column="vertragsNr"/>
        <many-to-many class="Fahrer" column="fuehrerscheinNr"/>
    </set>
...</class>
</hibernate-mapping>
```

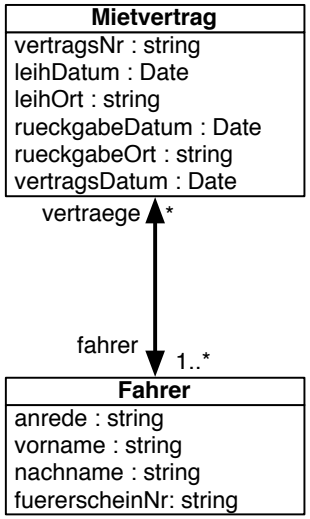
RENTaC–Fahrer — Resultierende Tabelle



SQL: Join-Tabelle VERTRAG_FAHRER

```
create table VERTRAG_FAHRER (
    vertragsNr varchar(255) not null,
    fuehrerscheinNr varchar(255) not null,
    primary key (fuehrerscheinNr, vertragsNr)
) type=MyISAM

...
```

RENTaC–Fahrer — Resultierende Tabelle



SQL: Join-Tabelle VERTRAG_FAHRER

```
...

alter table VERTRAG_FAHRER
    add index FK4CB7208A8DBE4DDD (fuehrerscheinNr),
    add constraint FK4CB7208A8DBE4DDD
    foreign key (fuehrerscheinNr)
    references fahrer (fuehrerscheinNr)

alter table VERTRAG_FAHRER
    add index FK4CB7208A89F6FF24 (vertragsNr),
    add constraint FK4CB7208A89F6FF24
    foreign key (vertragsNr)
    references Mietvertrag (vertragsNr)
```

Ausblick

Mapping von Entity–Beziehungen

1:N–Assoziationen

Entity vs. Value

Beispiel

Kunden
Mietverträge
Fahrer
Punktekonto
Mietverträge/Fahrer

- Fragen?
- nächste Woche: mehr über Hibernate: Bottom Up-Entwicklung