

# Datenbanksysteme II

## SS 2017 – Übungsblatt 3

### Teillösung

Universität Leipzig, Institut für Informatik

Abteilung Datenbanken

Prof. Dr. E. Rahm,

V. Christen, M. Franke

# Aufgabe 1a

Abteilung												
Abtnr	MgrNr	Budget	Mitarbeiter				Ausstattung					
1	49	500.000	MaNr	Funktion	Projekt		Anzahl	Typ				
					PNr	PName						
						39582	Entwickler	17	XYZ	8	Sun	
								23	ABC	12	Telefon	
						69011	Sekretärin	17	XYZ			
									18	MN		
									32	STU		
			...	...	...	...						
2	48	600.000										

# Aufgabe 1a - Redundanzen

Abteilung												
Abtnr	MgrNr	Budget	Mitarbeiter				Ausstattung					
1	49	500.000	MaNr	Funktion	Projekt		Anzahl	Typ				
					PNr	PName						
						39582	Entwickler	17	XYZ	8	Sun	
								23	ABC	12	Telefon	
						69011	Sekretärin	17	XYZ			
									18	MN		
									32	STU		
			...	...	...	...						
2	48	600.000										

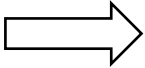
# Aufgabe 1b –SET/ROW Konstruktoren

```
CREATE Table Abteilung(  
    Abtnr      INT,  
    MgrNr      INT,  
    Budget     FLOAT,  
    Mitarbeiter SET(ROW(MaNr  INT,  
                        Funktion VARCHAR(20),  
                        Projekt  SET(ROW(PNR  INT  
                        PName VARCHAR(20)  
                        ))  
    ),  
    Ausstattung SET(ROW(Anzahl INT,  
                        Typ      VARCHAR(20)  
                        ))  
)
```

# Aufgabe 2a – NF2-Operationen

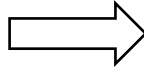
A	D	
	B	C
1	5	8
	6	9
1	2	3
7	1	4

R

  $\text{Unnest}_D(R)$

A	B	C
1	5	8
1	6	9
1	2	3
7	1	4

R'

  $\text{Nest}_{B,C:D}(R')$

A	D	
	B	C
1	5	8
	6	9
	2	3
7	1	4

R''≠R

# Aufgabe 3 - Komplexe Objekte/Typkonstruktoren

```
class ZWEIRAD (  
  Marke: String,  
  Gewicht: integer,  
  Rahmen: REF (RAHMEN),  
  Räder: ARRAY[2]( REF (LAUFRAD ));
```

```
class LAUFRAD (  
  Größe: integer,  
  SpeichenAnzahl: integer );
```

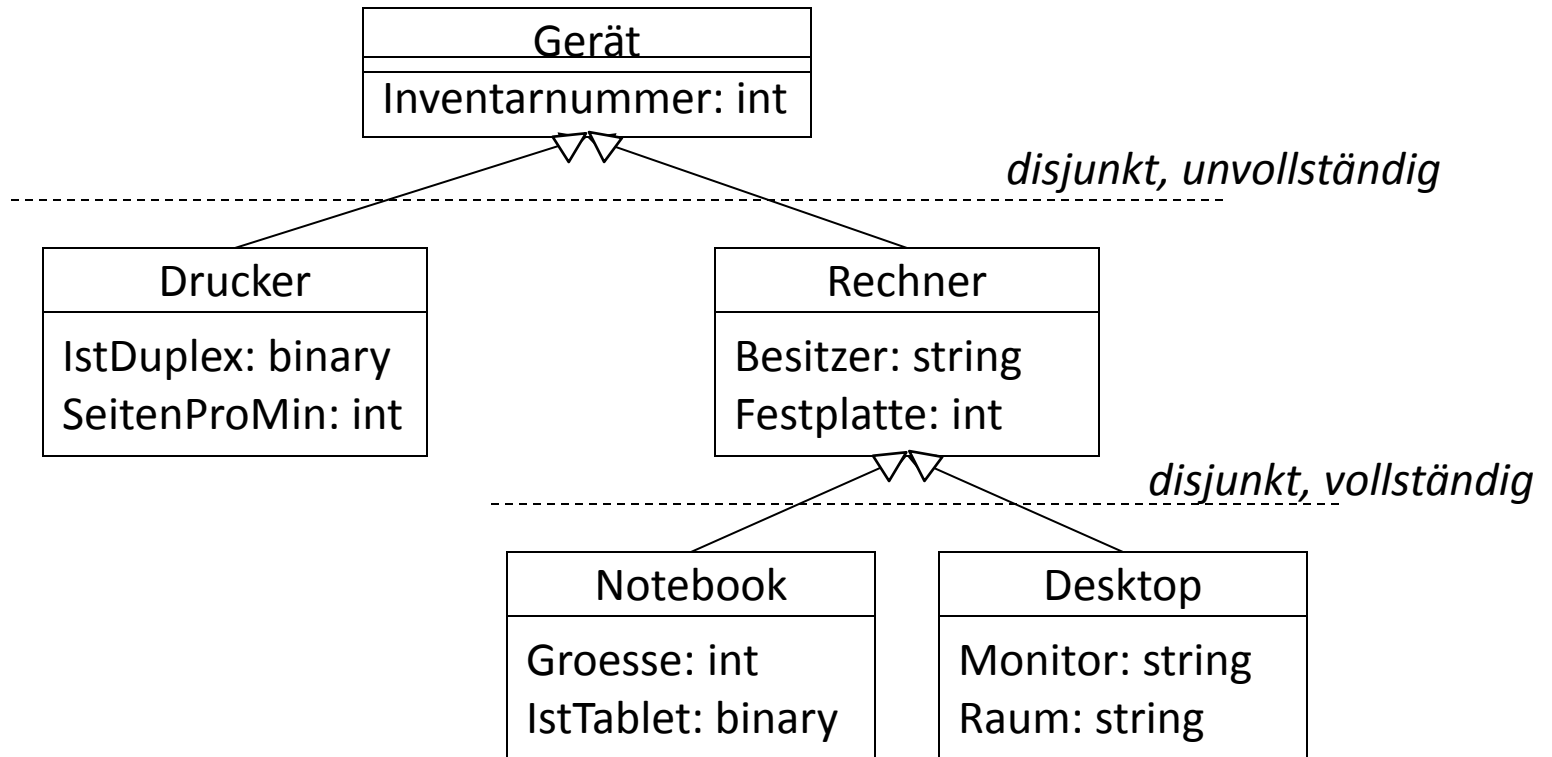
```
class RAHMEN (  
  Material: String,  
  Größe: integer );
```

```
class FUSSBALLMANNSCHAFT (  
  Spieler: ARRAY[11]( REF( SPIELER ))  
);
```

```
class SPIELER (  
  Spielername:  
    TUPLE( Vorname: String, Nachname: String ),  
  Spielernummer: integer,  
  Spieleralter: integer,  
  Mannschaft: REF( FUSSBALLMANNSCHAFT )  
);
```

```
class SPIEL (  
  Gegner: ARRAY[2]( REF( FUSSBALLMANNSCH. )),  
  Torschützen: BAG( REF( SPIELER ))  
);
```

# Aufgabe 4 - Objekt-Relationales Mapping (Vererbung)



# Table-per-(concrete-)class: horizontale Part.

- Je Instanz nur in ihrer “Spezialklasse” mit allen Attributen
- Keine Redundanz in den Instanzdaten (lediglich in den Metadaten)
- Tabelle Geraet notwendig, da Spezialisierung unvollständig
- Tabelle Rechner nicht notwendig, da Spezialisierung vollständig

Geraet
<u>InvNr</u>
104

Drucker		
<u>InvNr</u>	IstDupl	SpM
101	Ja	12

Notebook				
<u>InvNr</u>	Besitzer	Festplatte	Groesse	IstTablet
102	Rahm	100	12	ja

Desktop				
<u>InvNr</u>	Besitzer	Festplatte	Monitor	Raum
103	Thor	80	21	4-31

- + Anfragen mit allen Attributen einer Instanz
- – Anfragen über alle Instanzen eines Typs (UNION notwendig)



# Table-per-(concrete-)class: horiz. Part.: XML

```
<hibernate-mapping>
  <class name="Geraet" table="GERAET">
    <id name="inventarNummer" type="int" column="INVENTARNUMMER"/>
    <union-subclass name="Drucker" table="DRUCKER">
      <property name="istDuplex" column="ISTDUPLEX" />
      <property name="seitenProMin" column="SEITENPROMIN" />
    </union-subclass>
    <union-subclass name="Rechner" table="RECHNER" abstract="true">
      <property name="besitzer" column="BESITZER" />
      <property name="festPlatte" column="FESTPLATTE" />
      <union-subclass name="Notebook" table="NOTEBOOK">
        <property name="groesse" column="GROESSE" />
        <property name="istTablet" column="ISTTABLET" />
      </union-subclass>
      <union-subclass name="Desktop" table="DESKTOP">
        <property name="monitor" column="MONITOR" />
        <property name="raum" column="RAUM" />
      </union-subclass>
    </union-subclass>
  </class>
</hibernate-mapping>
```

# Table-per-(concrete-)class: horiz. Part.: Annotationen

```
import javax.persistence.*;

@Entity
@Table(name = "Geraet")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Geraet{

    @Id
    @Column(name = "InvNr")
    private int invNr;

    // ...
}

@Entity
@Table(name = "Drucker")
public class Drucker extends Geraet{

    @Column(name = "IstDupl")
    private boolean istDupl;

    @Column(name = "SpM")
    private int SpM;
}

@Entity
public abstract class Rechner extends Geraet{

    @Column(name = "Besitzer")
    private boolean besitzer;

    @Column(name = "Festplatte")
    private int festplatte;

    // ...
}

@Entity
@Table(name = "Notebook")
public class Notebook extends Rechner{
    // ...
}

@Entity
@Table(name = "Desktop")
public class Desktop extends Rechner{
    // ...
}
```

# Table-per-subclass: vertikale Partitionierung

- Attribute zu Instanzen werden jeweils in Spezialklasse gespeichert
- Id-Attribut wird dupliziert (evtl. Fremdschlüssel)

Geraet
<u>InvNr</u>
101
102
103
104

Drucker		
<u>InvNr</u>	IstDupl	SpM
101	Ja	12

Rechner		
<u>InvNr</u>	Besitzer	Festplatte
102	Rahm	100
103	Thor	80

Notebook		
<u>InvNr</u>	Groesse	IstTablet
102	12	ja

Desktop		
<u>InvNr</u>	Monitor	Raum
103	21	4-31

- + Anfragen über alle Instanzen eines Typs
- – Anfragen mit allen Attributen einer Instanz (JOIN notwendig)

# Table-per-subclass: vertikale Part.: XML

```
<hibernate-mapping>
  <class name="Geraet" table="GERAET">
    <id name="inventarNummer" type="int" column="INVENTARNUMMER"/>
    <joined-subclass name="Drucker" table="DRUCKER">
      <key column="INVENTARNUMMER"/>
      <property name="istDuplex" column="ISTDUPLEX" />
      <property name="seitenProMin" column="SEITENPROMIN" />
    </joined-subclass>
    <joined-subclass name="Rechner" table="RECHNER">
      <key column="INVENTARNUMMER"/>
      <property name="besitzer" column="BESITZER" />
      <property name="festPlatte" column="FESTPLATTE" />
      <joined-subclass name="Notebook" table="NOTEBOOK">
        <key column="INVENTARNUMMER"/>
        <property name="groesse" column="GROESSE" />
        <property name="istTablet" column="ISTTABLET" />
      </joined-subclass>
      <joined-subclass name="Desktop" table="DESKTOP">
        <key column="INVENTARNUMMER"/>
        <property name="monitor" column="MONITOR" />
        <property name="raum" column="RAUM" />
      </joined-subclass>
    </joined-subclass>
  </class>
</hibernate-mapping>
```

# Table-per-subclass: vertikale Part.: Annotationen

```
import javax.persistence.*;

@Entity
@Table(name = "Geraet")
@Inheritance(strategy = InheritanceType.JOINED)
public class Geraet{

    @Id
    @Column(name = "InvNr")
    private int invNr;

    // ...

}

@Entity
@Table(name = "Drucker")
@PrimaryKeyJoinColumn(name = "InvNr")
public class Drucker extends Geraet{

    @Column(name = "IstDupl")
    private boolean istDupl;

    @Column(name = "SpM")
    private int spM;

}

@Entity
@Table(name = "Rechner")
@Inheritance(strategy = InheritanceType.JOINED)
@PrimaryKeyJoinColumn(name = "InvNr")
public class Rechner extends Geraet{

    @Column(name = "Besitzer")
    private boolean besitzer;

    @Column(name = "Festplatte")
    private int festplatte;

    // ...

}

@Entity
@Table(name = "Notebook")
@PrimaryKeyJoinColumn(name = "InvNr")
public class Notebook extends Rechner{

    // ...

}

@Entity
@Table(name = "Desktop")
@PrimaryKeyJoinColumn(name = "InvNr")
public class Desktop extends Rechner{

    // ...

}
```

# Table-per-class-hierarchy

- Nur eine (große) Tabelle pro Hierarchie (“wide table”)
- Instanz wird durch Diskriminator-Wert (hier: Typ) typisiert
- (viele) NULL-Werte, falls Attribut für Typ nicht relevant
- bei vielen Klassen/Attributen (sehr) unübersichtlich

Geraet (frei=NULL)									
<u>InvNr</u>	Typ	IstDupl	SpM	Besitzer	Festplatte	Groesse	IstTablet	Monitor	Raum
101	Drucker	Ja	12						
102	Notebook			Rahm	100	12	ja		
103	Desktop			Thor	80			21	4-31
104	Geraet								

- + Anfragen mit allen Attributen einer Instanz
- ○ Anfragen über alle Instanzen eines Typs (Disjunktion von Typwerten nötig)

# Table-per-class-hierarchy: XML

```
<hibernate-mapping>
  <class name="Geraet" table="GERAET">
    <id name="inventarNummer" type="int" column="INVENTARNUMMER"/>
    <discriminator column="TYP" type="string"></discriminator>
    <subclass name="Drucker" discriminator-value="DRUCKER">
      <property name="istDuplex" column="ISTDUPLEX" />
      <property name="seitenProMin" column="SEITENPROMIN" />
    </subclass>
    <subclass name="Rechner" discriminator-value="RECHNER">
      <property name="besitzer" column="BESITZER" />
      <property name="festPlatte" column="FESTPLATTE" />
      <subclass name="Notebook" discriminator-value="NOTEBOOK">
        <property name="groesse" column="GROESSE" />
        <property name="istTablet" column="ISTTABLET" />
      </subclass>
      <subclass name="Desktop" discriminator-value="DESKTOP">
        <property name="monitor" column="MONITOR" />
        <property name="raum" column="RAUM" />
      </subclass>
    </subclass>
  </class>
</hibernate-mapping>
```

# Table-per-class-hierarchy: Annotationen

```
import javax.persistence.*;

@Entity
@Table(name = "Geraet")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(
    name = "Typ",
    discriminatorType = DiscriminatorType.STRING
)
@DiscriminatorValue(value = "Geraet")
public class Geraet{

    @Id
    @Column(name = "InvNr")
    private int invNr;

    // ...

}

@Entity
@Table(name = "Geraet")
@DiscriminatorValue("Drucker")
public class Drucker extends Geraet{

    @Column(name = "IstDupl")
    private boolean istDupl;

    @Column(name = "SpM")
    private int SpM;

}
```

```
@Entity
@Table(name = "Geraet")
@DiscriminatorValue("Rechner")
public class Rechner extends Geraet{
    // ...
}

@Entity
@Table(name = "Geraet")
@DiscriminatorValue("Notebook")
public class Notebook extends Rechner{
    // ...
}

@Entity
@Table(name = "Geraet")
@DiscriminatorValue("Desktop")
public class Desktop extends Rechner{
    // ...
}
```