Wirtschaftsinformatik 2

Prof. Dr. Dr. L. Schmidt-Thieme
MSc. André Busche

Übung 8

1. letzte Übung
2. Übungsblatt 8
3. Übungsblatt 7 – Aufg. 2f und 2g
4. Übungsblatt 6

Klausurtermin

Am 24.07.2012

Um 10 Uhr

Im A9

Ihr könnt euch ab sofort im LSF anmelden. Bitte tut dies auch.

```
module bank{

    interface Account {
        Customer owner();
        double balance();
        boolean withdraw (double amount);
        boolean transfer(Account which_target, double amount,
string text);
    }

    interface Customer {
        Account openAccount();
        attribute name;
        attribute adress;
        Account[] accounts;
    }
}
```

```
module bank{

    interface Account {
        Customer owner();
        double balance();
        boolean withdraw (in double amount);
        boolean transfer(in Account which_target, in double
amount, in string text);
    };

    interface Customer {
        Account openAccount();
        attribute string name;
        attribute string adress;
        Account[] accounts;
    };
};
```

Und noch ein weiteres Problem, was wir nachher sehen ...

Aufgabe 1a (4 Punkte)

Entwerfen Sie ein XML Instanzdokument, welches folgenden Sachverhalt modelliert:

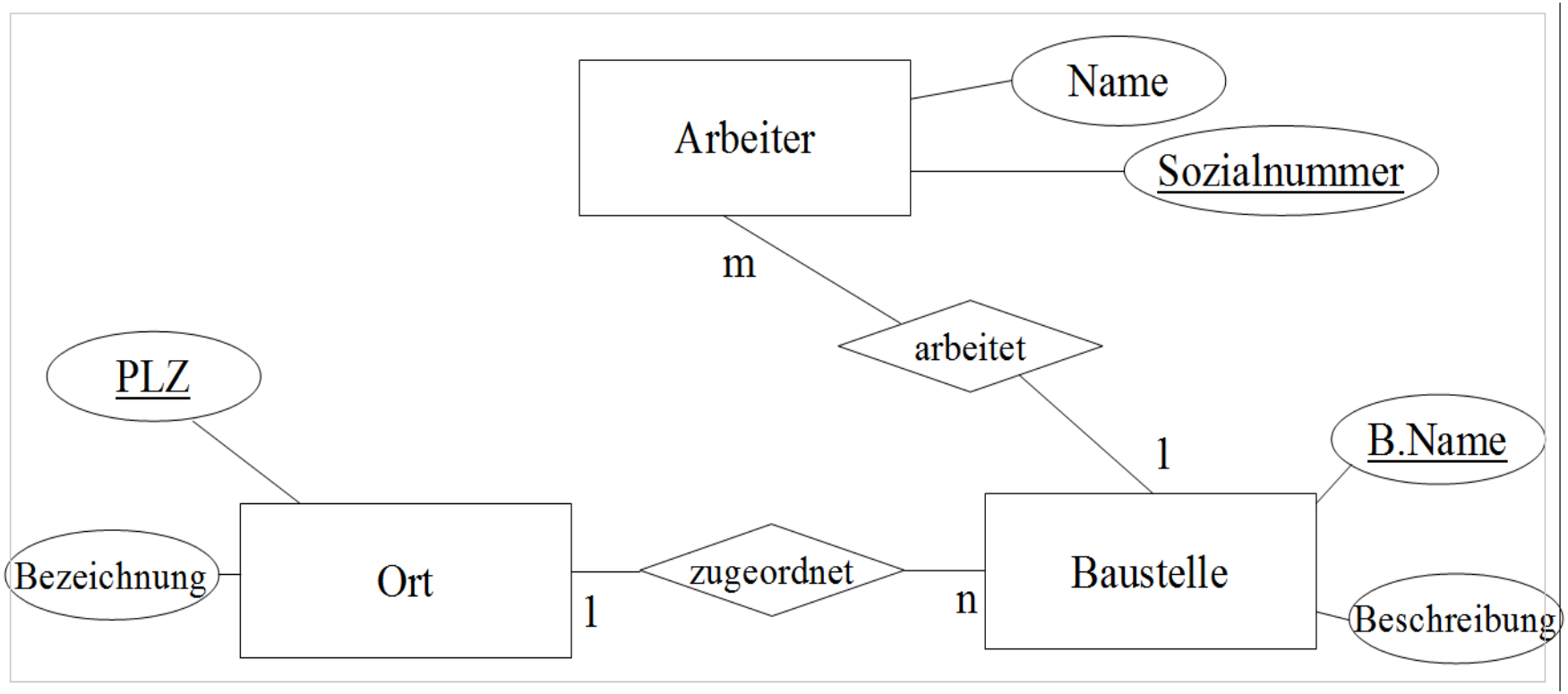
In Ort A gibt es zwei Baustellen. Auf der ersten Baustelle arbeiten die Arbeiter Meier, Müller und Schulze. Auf der anderen Baustelle arbeiten Dau-Schmidt, Jähne und Busche. In Ort B gibt es nur eine Baustelle, auf der die Arbeiter Meier, Busche, und Schmidt-Thieme beschäftigt sind. Die Arbeiter Meier und Busche können natürlich nicht zeitgleich auf den Baustellen A und B arbeiten...

Erweitern Sie, falls nötig, die Modellierung / das XML Dokument um weitere relevante Informationen, sofern Sie für die Modellierung des Sachverhalts notwendig sind.

Aufgabe 1b (6 Punkte)

Entwerfen Sie XPath / XQuery Ausdrücke, um die folgenden Informationen aus ihrem Beispieldokument aus Aufgabe 1a) abzufragen:

- Eine Liste aller Arbeiter
- Eine Liste aller Arbeiter, sortiert nach Namen, zusammen mit deren Sozialversicherungsnummer
- Eine Liste der Baustellen, zusammen mit der Anzahl der Arbeiter, die dort beschäftigt sind.
- Eine Liste der Baustellen pro Arbeiter (die Baustellen, auf denen ein Arbeiter beschäftigt ist).




```
<Ort>
   <Baustelle>
      <Arbeiter>

      </Arbeiter>
   </Baustelle>
</Ort>
```

Was ihr gemerkt habt: Probleme bei der Abfrage *„Alle Arbeiter“* (z. B. wenn diese doppelt sind...) und *„Baustellen pro Arbeiter“*

```
<Orte>
   <OrtA PLZ="11111" Bezeichnung="A">
      <Baustellen>
         <Baustelle1 B.Name="1" Beschreibung="BS1-OrtA">
            <Arbeiter name="Meier" Sozialnr="1"/>
            <Arbeiter name="Müller" Sozialnr="2"/>
            <Arbeiter name="Schulze" Sozialnr="3"/>
         </Baustelle1>
         <Baustelle2 B.Name="2" Beschreibung="BS2-OrtA">
            <Arbeiter name="Dau-Schmidt" Sozialnr="4"/>
            <Arbeiter name="Jähne" Sozialnr="5"/>
            <Arbeiter name="Busche" Sozialnr="6"/>
         </Baustelle2>
      </Baustellen>
   </OrtA>
   <OrtB PLZ="22222" Bezeichnung="B">
      <Baustelle3 B.Name="3" Beschreibung="BS-OrtB">
         <Arbeiter name="Meier" Sozialnr="1"/>
         <Arbeiter name="Busche" Sozialnr="6"/>
         <Arbeiter name="Schmidt-Thieme" Sozialnr="7"/>
```

- Eine Liste aller Arbeiter
- Eine Liste aller Arbeiter, sortiert nach Namen, zusammen mit deren Sozialversicherungsnummer
- Eine Liste der Baustellen, zusammen mit der Anzahl der Arbeiter, die dort beschäftigt sind.
- Eine Liste der Baustellen pro Arbeiter (die Baustellen, auf denen ein Arbeiter beschäftigt ist).

```
<baustellenplan>
   <ort bezeichnung="A" plz="31061" zeitraum="10.06 - 14.06">
      <baustelle b.name="haus" beschreibung="...">
         <arbeiter snummer="110">Meier</arbeiter>
         <arbeiter snummer="220">Schulze</arbeiter>
         <arbeiter snummer="330">Müller</arbeiter>
      </baustelle>
      <baustelle b.name="garage" beschreibung="...">
         <arbeiter snummer="440">Dau-Schmidt</arbeiter>
         <arbeiter snummer="550">Jähne</arbeiter>
         <arbeiter snummer="660">Busche</arbeiter>
      </baustelle>
   </ort>
   <ort bezeichnung="B" plz="31840" zeitraum="25.07-30.07">
      <baustelle b.name="keller" beschreibung="...">
         <arbeiter snummer="110">Meier</arbeiter>
         <arbeiter snummer="660">Busche</arbeiter>
         <arbeiter snummer="770">Schmidt-Thieme</arbeiter>
      </baustelle>
   </ort>
</baustellenplan>
```

- Liste Arbeiter
- Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer
- Liste Baustellen, mit der Anzahl der Arbeiter.
- Liste Baustellen pro Arbeiter.

```
<?xml version="1.0" encoding="UTF-8"?>
<Orte>
   <Ort PLZ="31141" Bezeichnung="Ort A">
      <Baustelle B.name="Baustelle 1A" Beschreibung="erste Baustelle in Ort A"> </Baustelle>
      <Baustelle B.name="Baustelle 2A" Beschreibung="zweite Baustelle in Ort A"> </Baustelle>
   </Ort>
   <Ort PLZ="27318" Bezeichnung="Ort B">
      <Baustelle B.Name="Baustelle 1B" Beschreibung="erste Baustelle in Ort B"> </Baustelle>
   </Ort>
   <Arbeiters>
      <Arbeiter Name="Meier" Sozialnummer="123456"/>
      <Arbeiter Name="Müller" Sozialnummer="234567"/>
      <Arbeiter Name="Schulze" Sozialnummer="345678"/>
   </Arbeiters>
   <arbeitet>
      <Arbeiter1>
         <SozialNummer>123456</SozialNummer>
         <Baustelle>Baustelle 1A</Baustelle>
         <Baustelle>Baustelle 2A</Baustelle>
         <ZurZeitTätig>Baustelle 1A</ZurZeitTätig>
      </Arbeiter1>
      <Arbeiter2>
         <SozialNummer>234567</SozialNummer>
         <Baustelle>Baustelle 1A</Baustelle>
         <ZurZeitTätig
      </Arbeiter2>
   </arbeitet>
</Orte>
```

- Liste Arbeiter
- Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer
- Liste Baustellen, mit der Anzahl der Arbeiter.
- Liste Baustellen pro Arbeiter.

```
<?xml version="1.0" encoding="UTF-8"?>
<Orte>
   <Ort>
      <Baustelle bn="A" >
         <Arbeiter sn="1" n="Meier"/>
         <Arbeiter sn="2" n="Schulze"/>
      </Baustelle>
      <Baustelle bn="B" >
         <Arbeiter sn="3" n="Busche"/>
         <Arbeiter sn="4" n="Jähne"/>
      </Baustelle>
   </Ort>
   <Ort>
      <Baustelle bn="C" >
         <Arbeiter sn="1" n="Meier"/>
      </Baustelle>
   </Ort>
</Orte>
```

- Liste Arbeiter
- Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnum
- Liste Baustellen, mit der Anzahl der Arbeiter.
- Liste Baustellen pro Arbeiter.

- Liste Arbeiter
  distinct-values(//Arbeiter/@n)

- Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer

- Liste Baustellen, mit der Anzahl der Arbeiter.

- Liste Baustellen pro Arbeiter.

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer

```
for $a in distinct-values(//Arbeiter/@n)

return $a
```

```
<?xml version="1.0" encoding="UTF-8"?>Meier Schulze Busche Jähne
```

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer

```
for $a in distinct-values(//Arbeiter/@n)

order by $a

return $a
```

```
<?xml version="1.0" encoding="UTF-8"?>Busche Jähne
Meier Schulze
```

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer

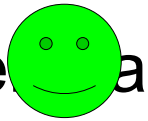
```
for $a in distinct-values(//Arbeiter/@n)

order by $a

return $a/../@sn
```

$a ist ein atomarer Wert,
Kein Knoten (ab dem man sich
Im Baum weiter bewegen kann...

… Macht auch keinen Sinn, da **`distinct-values`** Duplikate entfernen soll … welcher (von mehreren) Knoten sollte also in $a adressiert werden????

Glückwunsch: Modellierungsfehler erkannt

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer



```
for $a in //Arbeiter/@n

order by $a

return $a/string()
```

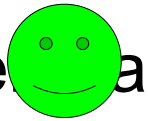
<?xml version="1.0" encoding="UTF-8"?>Busche Jähne Meier Meier Schulze

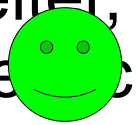
Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer

```
for $a in //Arbeiter/@n
let $sn:=$a/../@sn

order by $a


return concat($a/string(), " ", $sn)
```

```
<?xml version="1.0" encoding="UTF-8"?>Busche 3 Jähne
4 Meier 1 Meier 1 Schulze 2
```

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer



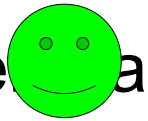


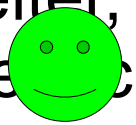
```
for $a in //Arbeiter/@n
let $sn:=$a/../@sn

order by $a


return <MA>{$a}{$sn}</MA>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<MA n="Busche" sn="3"/>
<MA n="Jähne" sn="4"/>
<MA n="Meier" sn="1"/>
<MA n="Meier" sn="1"/>
<MA n="Schulze" sn="2"/>
```

Not yet well-formed ...

Liste Arbeiter, sortiert nach Namen, mit Sozialversicherungsnummer





```
<MAs>
{
for $a in //Arbeiter/@n
let $sn:=$a/../@sn
order by $a
return <MA>{$a}{$sn}</MA>
}
</MAs>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<MAs>
  <MA n="Busche" sn="3"/>
  <MA n="Jähne" sn="4"/>
  <MA n="Meier" sn="1"/>
  <MA n="Meier" sn="1"/>
  <MA n="Schulze" sn="2"/>
</MAs>
```

- Liste Baustellen, mit der Anzahl der Arbeiter

```
for $b in //Baustelle

return $b
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Baustelle bn="A">
        <Arbeiter sn="1" n="Meier"/>
        <Arbeiter sn="2" n="Schulze"/>
</Baustelle>
```

- Liste Baustellen, mit der Anzahl der Arbeiter

```
for $b in //Baustelle

let $c:=count($b/Arbeiter)

return concat("Anzahl Arbeiter in ", $b/@bn, ":",$c)
```

<?xml version="1.0" encoding="UTF-8"?>Anzahl Arbeiter in A:2 Anzahl Arbeiter in B:2 Anzahl Arbeiter in C:1

- Liste Baustellen pro Arbeiter

```
for $a in distinct-values(//Arbeiter/@n)

return $a
```

```
<?xml version="1.0" encoding="UTF-8"?>Meier Schulze
Busche Jähne
```

- Liste Baustellen pro Arbeiter

```
for $a in distinct-values(//Arbeiter/@n)

let $c :=  //Baustelle[Arbeiter/@n=$a]

return concat($a, count($c))
```

<?xml version="1.0" encoding="UTF-8"?>Meier2 Schulze1 Busche1 Jähne1

```
<?xml version="1.0" encoding="UTF-8"?>
<ER>
    <Arbeiter sn="1" n="Meier"/>
    <Arbeiter sn="2" n="Schulze"/>
    <Arbeiter sn="3" n="Busche"/>
    <Arbeiter sn="4" n="Jähne"/>
    <Baustelle bn="A" />
    <Baustelle bn="B" />
    <Baustelle bn="C" />
    <arbeitet sn="1" bn="A"/>
    <arbeitet sn="1" bn="C"/>
    <arbeitet sn="2" bn="A"/>
    <arbeitet sn="3" bn="B"/>
    <arbeitet sn="4" bn="B"/>
</ER>
```

- Eine Liste aller Arbeiter
- Eine Liste aller Arbeiter, sortiert nach Namen, zusammen mit deren Sozialversicherungsnummer
- Eine Liste der Baustellen, zusammen mit der Anzahl der Arbeiter, die dort beschäftigt sind.
- Eine Liste der Baustellen pro Arbeiter (die Baustellen, auf denen ein Arbeiter beschäftigt ist).

- Liste Arbeiter

```
//Arbeiter
```

- Liste Arbeiter, sortiert Namen, mit SN

```
for $a in //Arbeiter
order by $a/@n
return $a
```

- Liste Baustellen, mit der Anzahl der Arbeiter.

```
for $b in //Baustelle
let $c := count(//arbeitet[@bn=$b/@bn])
return concat($b/@bn, $c)
```

→ A2 B2 C1

- Liste Baustellen pro Arbeiter.

```
<?xml version="1.0"
encoding="UTF-8"?>
<ER>
   <Arbeiter sn="1" n="Meier"/>
   <Arbeiter sn="2" n="Schulze"/>
   <Arbeiter sn="3" n="Busche"/>
   <Arbeiter sn="4" n="Jähne"/>
   <Baustelle bn="A" />
   <Baustelle bn="B" />
   <Baustelle bn="C" />
   <arbeitet sn="1" bn="A"/>
   <arbeitet sn="1" bn="C"/>
   <arbeitet sn="2" bn="A"/>
   <arbeitet sn="3" bn="B"/>
   <arbeitet sn="4" bn="B"/>
</ER>
```

Aufgabe 2a (3 Punkte)

Welche Aussagen zu Netzwerktopologien sind richtig? Falls Aussagen falsch sind, begründen und korrigieren Sie die Aussagen.

- In einer Client-Server Netzwerkarchitektur (Sternnetz) gibt es einen zentralen Vermittlungsknoten, an dem jeder andere Knoten direkt durch eine physikalische Verbindung angeschlossen ist.
- Ein Peer-to-Peer Netzwerk bietet den Vorteil einer großen Ausfallsicherheit, erfordert aber einen großen Verkabelungsaufwand.
- In einer Client-Server Netzwerkarchitektur ist es einfacher, Datenkonsistenz zu erhalten.

Aufgabe 2b (2+3 Punkte)

Ein verteiltes Baustellen-Informationssystem soll entwickelt werden. Sie entscheiden sich für CORBA, um die Kommunikation zwischen einzelnen logischen Rechner zu realisieren. Sie entscheiden sich weiterhin für oben stehende E/R Modellierung, um die Realität abzubilden.

Entwerfen Sie eine IDL, die oben stehende Entitäten als jeweils eigenständiges Interface definiert.

Erweitern Sie ihre IDL, sodass Sie über weitere Interfaces die Relationen der beteiligten Entitäten abfragen können. Diese Interfaces sollten beispielsweise Anfragen á lá „Welche Mitarbeiter arbeiten auf Baustelle B?“ ermöglichen.

Aufgabe 2b (2+3 Punkte)

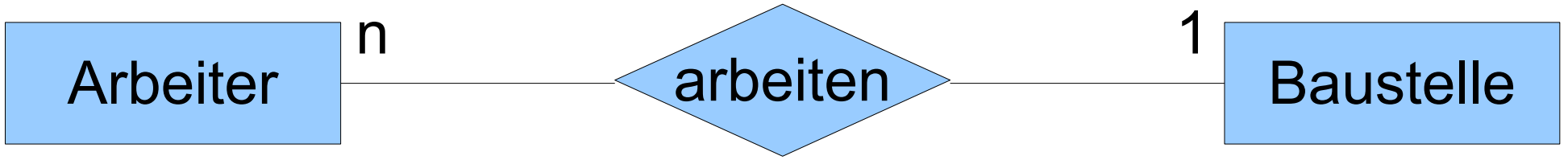
Entwerfen Sie eine IDL, die oben stehende Entitäten als jeweils eigenständiges Interface definiert.

```
module BauInf{
   interface Arbeiter{
      attribute string name;
      readonly attribute long sozialnummer;
      string einsatzort();//gibt den einsatzort eines Arbeiters an
   };
   interface Baustelle{
      attribute string BName;
      attribute string Beschreibung;
   };
   interface Ort{
      attribute long plz;
      readonly attribute string bezeichnung;
   };
};
```

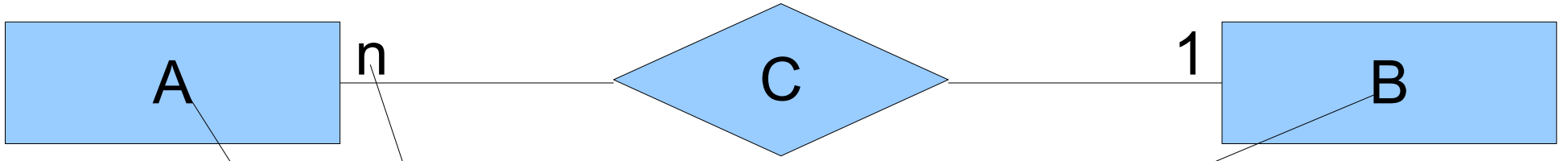
Aufgabe 2b (2+3 Punkte)

Erweitern Sie ihre IDL, sodass Sie über weitere Interfaces die Relationen der beteiligten Entitäten abfragen können. Diese Interfaces sollten beispielsweise Anfragen á lá „Welche Mitarbeiter arbeiten auf Baustelle B?“ ermöglichen.

Welche Mitarbeiter arbeiten auf Baustelle B



Aufgabe 2b (2+3 Punkte)



Welche A C B?

```
module ismll_commerce {

   interface A { };
   interface B { };

   typedef sequence<A> aseq;

   interface Accessor {
      aseq c(in B arg);
      B c2(in A arg);
   };
};
```

Accessor ist neu und gibt eine Sequence aus A-Objekten, gegeben einem B, zurück