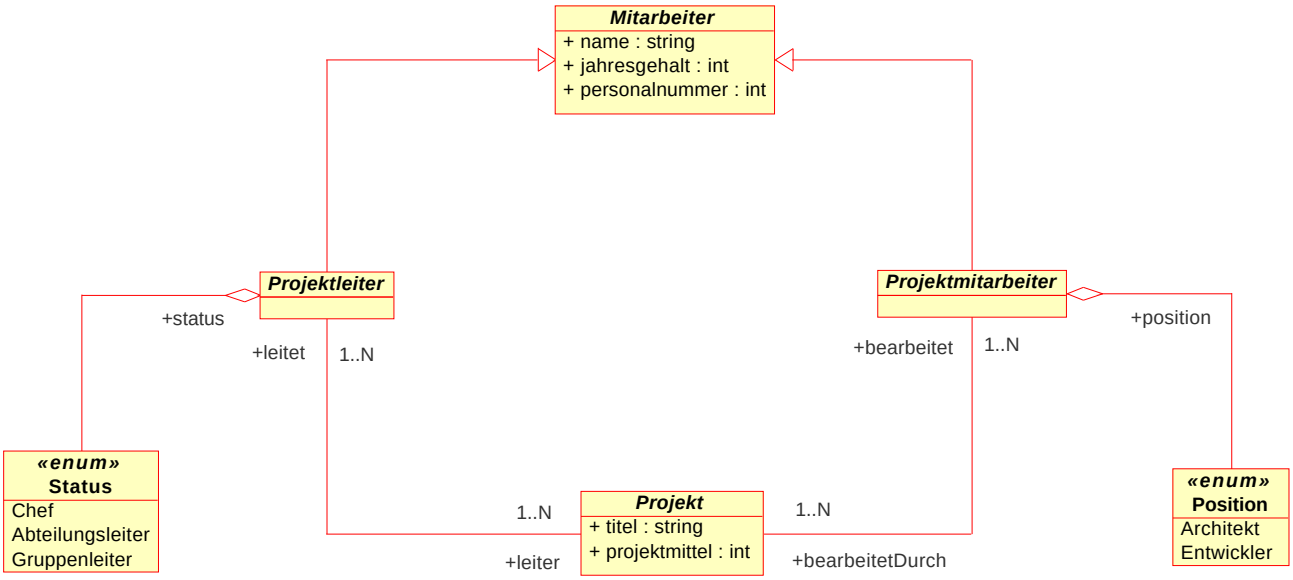
Informationssysteme II
Übungsblatt 2

Christine Pries, Mareike Wagner, Felix Oppermann (Gruppe 18)

3. Mai 2005

Aufgabe 1 - OODBMS I (30 Punkte)

a)



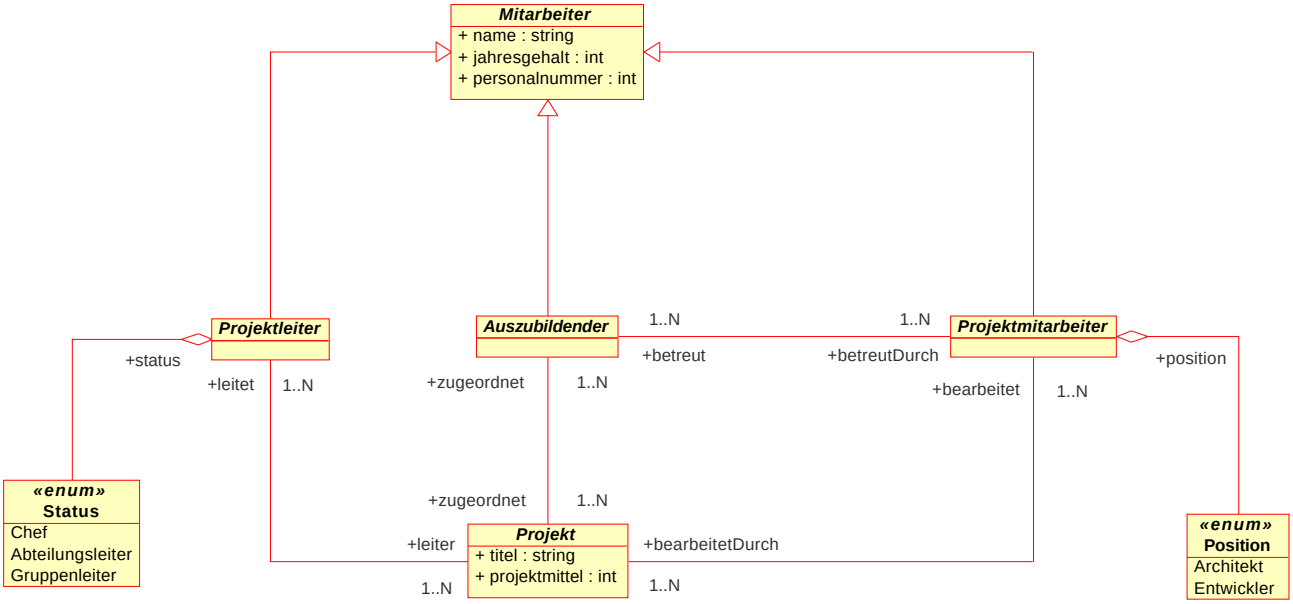
b) b-1)
```
select struct (name: l.name, personalnummer: l.personalnummer,  projekt: p)
    from Projektleiter l,  l.leitet p
    where l.status = Gruppenleiter
```

b-2)
```
select m.personalnummer
    from Mitarbeiter m
    where m.jahresgehlt > 50000
```

b-3)
```
select distict(*)
    from
    (
      (
        select personalnummer: m.personalnummer
          from Projektmitarbeiter m, m.bearbeitet p
          where p.title = "Tolles Informationsystem!"
      )
      union
      (
        select m.personalnummer
          from Projektleiter m, m.leitet p
          where p.title = "Tolles Informationsystem!"
      )
    )
```

b-4)
```
select m.personalnummer
    from Projektleiter l, l.leitet p, p.leiter m
    where l.personalnummer = 35553
    and.status = Abteilungsleiter
    and m.status = Gruppenleiter
```

c)




```
interface Mitarbeiter {
        attribute string name;
        attribute integer jahresgehalt;
        attribute integer personalnummer;
}

interface Projektleiter : Mitarbeiter {
       enum Status {Chef, Abteilungsleiter, Gruppenleiter};
        attribute Status status;
        relationship Set<Projekt> leitet inverse Projekt::leiter;
}

interface Projektmitarbeiter : Mitarbeiter {
       enum Position {Architekt, Entwickler};
        attribute Position position;
        relationship Set<Projekt> bearbeitet inverse Projekt::bearbeitetDurch;
        relationship Set<Auszubildender> betreut inverse Auszubildender::betreutDurch;
}

interface Projekt {
        attribute string titel;
        attribute integer projektmittel;
        relationship Set<Projektmitarbeiter> bearbeitetDurch inverse
                     Projektmitarbeiter::bearbeitet;
        relationship Set<Prjektleiter> leiter inverse
                     Projektleiter::leitet;
        relationship Set<Auszubildender> zugeordnet inverse
                     Auszubildender::zugeordnet;
}

interface Auszubildender : Mitarbeiter {
        relationship Set<Projekt> zugeordnet inverse
                     Projekt::zugeordnet;
        relationship Set<Projektmitarbeiter> betreutDurch inverse
                     Projektmitarbeiter::betreut
}
```

Aufgabe 2 - OODBMS II (25 Punkte)

Kriterien, die ein Datenbanksystem erfüllen muss/sollte, damit es die Bezeichnung OODBMS zu Recht tragen darf. Unterstützt werden müssen:

- Komplexe Objekte
- Objektidentität
- Verschachtelung von Objekten
- Typen oder Klassen
- Klassen/Typen sollten von ihren Vorgängern erben / Klassen- oder Typenhierarchie
- Späte Bindung
- Rechnerische Vollständigkeit
- Erweiterbarkeit
- Persistenz der Daten gewährleisten
- Sehr große Datenbanken
- Benutzung durch mehrere Benutzer ermöglichen
- Wiederherstellung von Daten nach Soft- und Hardwarefehlern
- Bereitstellung von Abfragemöglichkeiten

„Lieblingskriterien"

- **Erweiterbarkeit**
  Das System sollte neben den vordefinierten Typen, die Möglichkeit bieten eigene Typen anlegen zu können. Diese sollten im System nicht anders benutzt werden als die vom System definierten. Wie diese Typen im System unterstützt werden, soll vor der Applikation sowie dem Programmier versteckt bleiben
- **Persistenz der Daten gewährleiste**n
  Es wird eine dauerhafte Speicherung aller Daten erwünscht, d.h. die Speicherung soll unabhängig von dem Typ der Daten erfolgen. Weiterhin soll diese Speicherung automatisch erfolgen, ohne dass die Daten explizit persistent gespeichert werden müssen. Es dürfen keine gespeicherten Daten verloren gehen.
- **Benutzung durch mehrere Benutzer ermöglichen**
  Es sollen mehrere Benutzer zugelassen werden, die gleichzeitig am System arbeiten können. Das System muss daher ein kontrolliertes Teilen der Daten zulassen und unterstützen. Hierbei sollte die ACID-Kriterien erfüllt sein.
- **Bereitstellung von Abfragemöglichkeiten**
  Das System soll ermöglichen, dass die Benutzer einfache Abfragen direkt an die Datenbank stellen können. Dabei sollen nicht-triviale Anfragen in wenigen Worten ausgedrückt werden können. Um effizient zu sein, sollte es eine Anfrageobtimierung beinhalten. Weiterhin sollte es unabhängig von der Applikation eingesetzt werden können, so z.B. auf verschiedenen Datenbanken arbeiten können.

Aufgabe 3 - ORDBMS (25 Punkte)

Bedeutung des relationalen Modells für ORDBMS: Beim ORDBMS handelt es sich im Gegensatz zum OODBMS um eine Erweiterung des relationalen Modells. Eine Datenbank der dritten Generation unterstützt immernoch alle Funktionen einer rein relationalen Datenbank. Die durch das ORDBMS-Modell angebotenen Konzepte stellen somit eine Obermenge da, welche das relationale Modell umfasst. Hierbei wird das relationale Modell um Konzepte aus der objektorientierten Programmierung erweitert, wie z.B. komplexe Typen und Typenhirachien.

Aufgabe 4 - Query-Shipping & Data-Shipping (20 Punkte)

Query-Shipping

Query-Shipping wird besonders im Zusammenhang mit RDBMS verwendet. Hierbei werden vom Client Anfragen (Queries) an den Server gesendet, der daraufhin die Ergebnisse zurückliefert. Die Verarbeitung des Querys wird hierbei zum Server verlagert. Der Server bearbeitet die Anfrage einschließlich enthaltener Agregationen. Durch die Verarbeitung auf dem Server kann zudem der Kommunikationsaufwand verringert werden, da nur die Anfrage und das entgültige Ergebnis der Anfrage übertragen werden müssen. Beim Query-Shipping werden spezielle Anfragesprache, meist SQL, verwendet. Dies vereinfacht zudem die Kommunikation mit verschiedenen Datenbanksystemen.

Data-Shipping

Data-Shipping wird vor allem bei OODBMS verwendet und kann in Object-Server- und Page-Server-Realisierungen unterschieden werden. Beim Data-Shipping liefert der Server nur die unverarbeiteten Daten. Die Verarbeitung übernimmt der Client. Diese Architektur verbessert die Skalierbarkeit, da der zentrale Server weniger belastet wird.
Bei der Object-Server-Realisierung sendet der Client einen Object-Request an den Server, woraufhin dieser als Ergebnis Objekte zurücksendet. Diese Variante zeichnet sich besonders durch eine gute Nutzung CLientseitiger Caches aus. Darüberhinaus kann bei machen Anwendungen der Kommunikationsaufwand reduziert werden. Ein weiter Vorteil ist die relativ einfache Implementierung eines Object-Locking-Mechanismus auf dem Server.
Bei der Page-Server-Realisierung fragt der Client Seiten oder Segmente an, die vom Server zurückgeliefert werden. Auch bei dieser Variante kann bei bestimmten Anwendungen der Kommunikationsaufwand reduziert werden. Zudem wird durch eine stärker verlagerung von Funktionalität auf den Client der Server stärker entlastet. Die Page-Server-Realisierung ist im Systemdesign leichter umzusetzen als die Object-Server-Realisierung.