Prüfungsprotokoll zu MMDB/OODB im WS12/13

Atmosphäre: sehr entspannt
Note/Aufwand: Benotung erfolgt durchaus fair. Der Lernaufwand steht allerdings nicht wirklich in Relation zu dem, was letztlich gefragt wird. Das ist meist nur ein minimaler Teil.

(Es entsteht der Eindruck der Prüfer erwartet bei Antwortet spezielle Schlüsselwörter.
Diese sind bei den Antworten **fett** hervorgehoben.)

F: Möchten Sie mit OODB oder MMDB anfangen.
A: OODB.

F: Das ODMG Modell war ja doch ein Schwerpunkt der Veranstaltung. Beschreiben Sie es doch kurz.
A:

- Aufteilung: Objektmodell, ODL, OQL, Anbindung an OO-Programmiersprache
- es gibt **Objekte**, **Literale** und **Relationships**
- Sammlungsobjekte, atomare Objekte, strukturierte Literale, atomare Literale, benutzerdefinierte Literale, Sammlungsliterale
- Relationships sind immer **binär**, verbinden genau 2 Objekte miteinander
- Relationships sind immer **bidirektional**, 1:1, 1:N sowie N:M Beziehungen lassen sich mit Hilfe von Sammlungsobjekten (Set, Bag, List, Array) abbilden
- es gibt **Extensionen** (Menge aller Typinstanzen in der DB) + Schlüssel (eindeutig innerhalb der Extension)

F: Skizzieren Sie doch mal eine solche Beziehung an einem Beispiel.
A:

```
interface Person (extent Personen key Personenname) {
  …
  relationship Set<Kunde> Kunden inverse Kunde::Person;
  …
}
```

Die Relation wird immer eindeutig durch ihre Endpunkte bestimmt (**Traversierungspfad**!).

F: Was ist denn das spezielle an der bidirektionalen Beziehung?
A:

- Das Schema garantiert immer eine **Gegenreferenz**
- Sonderfall beim Löschen: Referenziertes und referenzierendes Objekt wissen durch Bidirektion voneinander; null-Referenzen werden daher vermieden
- Sonderfall beim Lesen: Vorgesetzter eines Vorgesetzten
  `Mitarbeiter m; m->Vorgesetzter->Vorgesetzter;`

F: Sie müssen einen Kunden davon überzeugen das ODMG Modell einzusetzen. Wie würden Sie das tun, denken Sie dabei speziell an den lesenden Zugriff.
A:

- das ODMG Modell unterstützt eigentlich jegliche Art von lesendem Zugriff
- iterativ: `select p.Kinder from Person p where …`
- **Extensionen können direkt verwendet werden: `Personen;`**
- Schachtelungen sind zugelassen (select im select, select im from, …)

F: Gehen wir zu MMDB über. Gehen Sie davon aus, Sie haben eine im Funktionsumfang vollständig ausgeführte relationale Multimediadatenbank die den neusten Anforderungen genügt. Ein Kunde möchte nun in dieser Röntgenbilder verwalten. Wie würden Sie dazu das Schema modellieren?
A: Neuste Anforderungen bedeutet Datentypen **TEXT und IMAGE sind zugelassene SQL Datentypen** und können daher verwendet werden.

```
relation Röntgenbilder(
  BildNr    integer primary key,
  PersonNr  integer foreign key,
  Bild      image);
```

F: Welche Operationen könnten Sie sich für den Datentyp IMAGE vorstellen?
A: Bei IMAGE handelt es sich ja um ein Rasterbild. Daher z.B.

- getHeight(), getWidth()
- getPixelmatrix()
- getColorMap()
- getSize()
- replacePart(from_x, from_y, to_x, to_y, newPart)
- …

F: Wie speichern Sie denn ein solches Bild in dieser Tabelle?

A:

- interaktiv: `insert into Röntgenbilder values(1, 1, IMAGE(dateiname.jpg));`
- eingebettet (Programm): `insert into Röntgenbilder values(1, 1, IMAGE(:bild));`

F: Wie würden Sie ein Bild lesen?

A:

- `select r.Bild.asJPG() from Röntgenbilder r where …`
- **Bild.asJPG aus Gründen der Formatunabhängigkeit**
- **internes Format soll und muss dem Anwender verborgen bleiben!**