Datenbanken TI 3, 11.10.2023

Fragen, Antworten, Kommentare

Kurz zur Programmierung auf dem ersten Aufgabenblatt: Sie sollten bei Collections immer über den passenden Typ nachdenken, hier war Map eine sehr gute Wahl, List noch ok. Erinnerung Map ist Key-Value (auch Dictionary), Set ist ohne doppelte (braucht equals und hashCode in Java), List ist mit doppelten und Reihenfolge. Es gibt noch mehr über Bibliotheken, wie MultiSet ist ohne Reihenfolge aber mit Doppelten.

Bei der Angabe von Typen immer den allgemeinsten auf der linken Seite, also Map nicht HashMap, List nicht ArrayList. Die Idee ist, dass so der konkrete Objekttyp leicht austauschbar ist. Nur wenn sie bewusst z. B. eine ArrayList und keine LinkedList haben wollen, wird der konkrete Typ angegeben.

Wichtig ist, dass Datenströme wieder geschlossen werden, da sonst schleichende Speicherlecks entstehen können. Java bietet mit try-with-Resources eine Möglichkeit, dass ein übergebenes Objekt garantiert geschlossen wird. Z. B.:

```
try (XMLDecoder in = new XMLDecoder(
        new BufferedInputStream(new FileInputStream(DATEI)))) {
    this.studis = (HashMap<Integer, Student>) in.readObject();
} catch (FileNotFoundException e) {
  //hier gehoert was Sinnvolles rein
}
```

Frage: Unsere zweite Lösung zu Aufgabe 8 würde zu NULL-Werten führen, die ja eigentlich vermieden werden sollen, ist das ok?

Antwort: Ist hier kein Problem. Es gibt insgesamt drei sinnvolle Lösungen, eine davon erzwingt NULL-Werte und wird trotzdem in der Praxis sehr häufig genutzt.

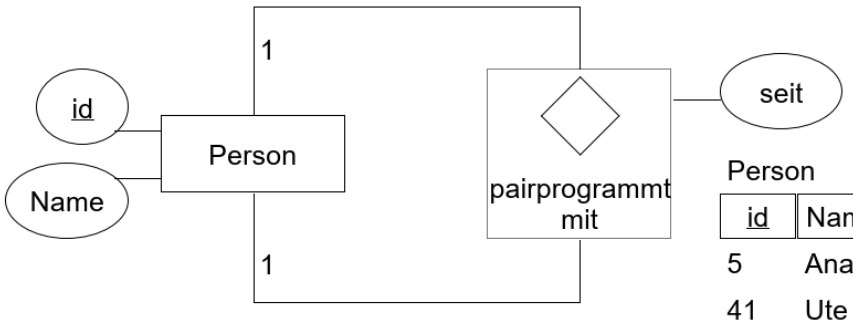
Frage: Wie wird eine reflexive 1:C-Relation umgesetzt?

Antwort: Gar nicht, da es sie in dieser Form nicht geben kann. Gehen wird von einer Menge M= {m1, m2, …, mn} und einer Relation $M \times M$ aus. Zu jedem Element auf der linken Seite soll es ein Element aus M auf der rechten Seite geben, dabei darf kein Element rechts doppelt vorkommen. Das geht, wenn rechts eine Permutation von M steht. Aber jetzt soll mindestens ein Element nicht rechts vorkommen, das geht nicht, da jedes Element benötigt wird oder eines auf der rechten Seite doppelt vorkommen müsste.

Frage: Wie wird eine reflexive 1:1-Relation umgesetzt?

Antwort: Der Schlüssel(kandidat) kommt einmal als Identifikator und einmal als Fremdschlüssel vor. Jede Person im Team hat genau eine andere Person mit der sie Pairprogramming macht. Da hier ein

Attributwert redundant doppelt auftritt, wäre hier zu überlegen einen neuen Entitätstypen Pair mit zwei Relationen zu Person zu ergänzen. Hier könnte wie im Ursprungsdiagramm aber nicht sichergestellt werden, dass eine Person nicht mit sich selbst ein Pair bildet. Es werden also formale Randbedingungen (Constraints) benötigt.



Person

id	Name	pair_id	seit
5	Ana	41	20
41	Ute	5	20