TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)
http://db.in.tum.de/teaching/ss23/ei2/

Blatt Nr. 10
Dieses Blatt wird am Montag, den 10.07.2023 besprochen.

SQL-Webschnittstelle: `http://hyper-db.com/interface.html`.

Aufgabe 1: SQL Gruppierung

Formulieren Sie die folgenden Anfragen auf dem bekannten Universitätsschema in SQL:

(a) Bestimmen Sie das durchschnittliche Semester der Studenten der Universität.

(b) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.

(c) Bestimmen Sie, wie viele Vorlesungen im Schnitt pro Student gehört werden. Beachten Sie, dass Studenten, die keine Vorlesung hören, in das Ergebnis einfließen müssen.

Aufgabe 2: SQL Exists

Formulieren Sie die folgenden Anfragen auf dem bekannten Universitätsschema in SQL:

(a) Finden Sie die Namen der *Studenten*, die in keiner *Prüfung* eine bessere Note als 3.0 hatten.

(b) Finden Sie alle Professoren die einen Assistenten haben, aber dennoch keine Vorlesung geben.

Aufgabe 3: Bekanntheitsgrad

Formulieren Sie eine SQL-Anfrage, um den Bekanntheitsgrad von Studenten zu ermitteln. Der Bekanntheitsgrad eines Studenten ist definiert als die Anzahl an Studenten die ihn kennen. Gehen Sie dabei davon aus, dass Studenten sich aus gemeinsam besuchten Vorlesungen kennen. Sortieren Sie das Ergebnis absteigend nach Bekanntheitsgrad!

Bonus: Überlegen Sie sich warum Studenten die niemanden kennen nicht im Ergebnis auftauchen. Wie könnte man dieses Problem lösen.

Aufgabe 4: Programmieraufgabe: Linear Hashing ⌨

Implementieren Sie in Java ein Hashset, das "linear hashing" zur Kollisionsbehandlung verwendet und die in der Zentralübung besprochene Invariante nach dem Löschen von Elementen wiederherstellt. Benutzen Sie **keine** Tombstones (Grabsteine), sondern verschieben Sie nach dem Löschen die Elemente sukzessive näher zu deren optimaler Position, so wie es in der ZÜ besprochen wurde.

Das Hashset muss wachsen und schrumpfen können, wenn jeweils der maximale oder der minimale load factor über- bzw. unterschritten wird. In diesen Fällen verdoppeln oder halbieren Sie die Anzahl an Buckets.

Tipp: implementieren Sie die dynamische Größenanpassung erst zum Schluss und benutzen Sie die Übungsaufgabe auf Blatt 8 mit einer festen Größe von $m = 13$, um Ihren Code zu überprüfen.

Sie finden den Code und einige Testfälle auf `https://gitlab.db.in.tum.de/DropTableGrades/linearhashing`.