Übungen zum JOIN-Befehl

David Tyndall

1 Übungen

1.1 Beziehungstypen

Aufgabe 1: In einer kleinen Schul-Datenbank seien folgende Tabellen vorhanden:

- Hausaufgaben (z. B. „Buch S. 27 lesen“)
- Fächer (Deutsch, Mathe, usw.)
- Klassen (1a, 1b, 2a, usw.)
- Schüler (Ueli, Peter, Nadine, usw.)

a) Welcher Typ Beziehung (1:1, 1:N, N:1 M:N) ist zwischen folgenden Tabellen zu erwarten?

   i. Hausaufgaben und Fächer

   ii. Fächer und Klassen

   iii. Klassen und Schüler

   iv. Schüler und Fächer

b) Kannst du selber eine Tabelle hinzufügen, mit der eine 1:1-Verbindung aus irgendeiner der obigen Tabellen Sinn machen würde?

1.2 JOIN-Befehl

Die folgenden Aufgaben beziehen sich auf die uni.db-SQLite-Datenbank.

Falls du noch keinen Zugriff auf einen SQLite-Editor hast, kannst du die Datei z. B. online auf `https://sqliteonline.com/` öffnen. Alternativ kannst du den „DB Browser for SQLite“ auf `https://sqlitebrowser.org/dl/` herunterladen und installieren.

Die Datenbank besteht aus folgenden Tabellen. In Klammern sind die Attribute angegeben, die Primary Keys sind unterstrichen.

- student (studno, name, semester): Liste der Studierenden inkl. ihrem Studiensemester
- lecture (courseno, title, credits, givenby): Vorlesungen
- professor (persno, name, rank, office): Liste der Professoren
- assistant (persno, name, area, worksfor): Assistenten der Professoren, inkl. ihrem Forschungsfeld
- attends (studno, courseno): Liste der Vorlesungensteilnehmenden
- examines (studno, courseno, persno, grade): Noten der Studierenden pro Kurs, inkl. Examinatoren
- requires (prereq, advanced): Vorlesungen, welche Vorbedingung für andere Vorlesungen sind

Aufgabe 2: Verbinden zweier Tabellen

Entwirf selber SQL-Abfragen, die Folgendes ausgeben. Verwende dabei Aliasse.

a) Alle Vorlesungen, die einen Professor haben. Ausgabe: Die Namen der Vorlesungen sowie die Namen ihrer Professoren

b) Der Name des Professors, der die Vorlesung „Logics“ hält

c) Alle Professoren, die Assistierende haben. Ausgabe: Namen der Professoren und Assistierenden

Aufgabe 3: Verbinden mehrerer Tabellen

In SQL kann man mehr als zwei Tabellen einfach durch Aneinanderhängen von mehreren `JOIN`s zusammenfügen. Dies ist z. B. bei M:N-Verbindungen immer nötig. Entwirf selber SQL-Abfragen, die Folgendes ausgeben:

a) Eine Liste der Namen der Studierenden mit allen Vorlesungstiteln, die sie besuchen

b) Eine Liste der Namen der Studierenden, die ein Professor unterrichtet (Weshalb treten gewisse Studenten-Professor-Verbindungen mehrfach auf?)

c) Name der Studierenden, Vorlesungstitel, Note

Aufgabe 4: Mehrdeutigkeit

Folgende Abfrage sollte eigentlich eine Liste der Studierenden mit ihren Vorlesungen generieren (14 Einträge). Leider ist aber etwas schief gelaufen; die Abfrage liefert 1232 Zeilen. Kannst du begründen, weshalb genau 1232 Zeilen ausgegeben werden? Was fehlt bei der Abfrage?

```
SELECT l.title, s.name
FROM  lecture l
        JOIN attends a
        JOIN student s
```

Aufgabe 5: NATURAL JOIN

Vereinfache alle Anfragen von Aufgabe 3 mit `NATURAL` so weit als möglich.

Aufgabe 6: LEFT/RIGHT JOIN

a) Gib *alle* Vorlesungen zusammen mit den Namen des Professors aus.

b) Gib eine Liste mit den Namen der Professoren sowie ihren Assistierenden aus. Jeder Professor soll erscheinen, egal, ob er Assistierende hat oder nicht.

c) Gib eine Liste der Studierenden aus, welche aktuell keine Vorlesung besuchen.

d) Gegeben sei folgende Abfrage. Sie gibt eine Liste der Vorlesungen aus, welche eine andere Vorlesung voraussetzen. Erweitere diese Abfrage dahingehend, dass in der linken Spalte *alle* Vorlesungen angezeigt werden, und nicht nur diejenigen, die eine Voraussetzung sind. Begründe, weshalb welche Ergänzungen notwendig sind.

```
SELECT lp.title AS Voraussetzung, la.title AS Folgevorlesung
FROM  lecture lp
        JOIN requires r ON lp.courseNo = r.prereq
        JOIN lecture la ON la.courseNo = r.advanced
```

e) Gegeben sei folgende Abfrage, welche 16 Zeilen ausgibt. Sie gibt eine Liste der Vorlesungen und mit allen Studierenden aus, die sie besuchen. Schreibe die Abfrage so um, dass kein `RIGHT JOIN` mehr verwendet wird.

```
SELECT l.title, s.name
FROM  student s
        RIGHT JOIN attends a ON s.studNo = a.studNo
        RIGHT JOIN lecture l ON l.courseNo = a.courseNo
```

2 Lösungen

Aufgabe 1:

a) Welcher Typ Beziehung (1:1, 1:N, N:1 M:N) ist zwischen folgenden Tabellen zu erwarten?

i. 1:N. Jede Hausaufgabe ist eindeutig einem Fach zugeordnet. Von einem Fach können aber mehrere Hausaufgaben gegeben werden.

ii. M:N. Jede Klasse hat mehrere Fächer, und jedes Fach wird in mehreren Klassen unterrichtet.

iii. 1:N. Jede Schülerin und jeder Schüler ist genau einer Klasse zugeteilt. Die Klasse enthält aber mehrere Schülerinnen und Schüler.

iv. M:N. Jede Schülerin und jeder Schüler besucht mehrere Fächer. Jedes Fach wird von mehreren Schülerinnen und Schülern besucht.

b) Ein paar Beispiele:

a) Legi: Jeder Schülerin und jedem Schüler wird genau eine Legi zugeordnet.

b) Spind: Jeder Schülerin und jedem Schüler wird genau ein Spind zugeordnet.

c) Klassenlehrer: Jeder Klasse wird genau eine Klassenlehrperson zugeordnet.

d) Klassenzimmer: Jeder Klasse wird genau ein Klassenzimmer zugeordnet.

Aufgabe 2: Verbinden zweier Tabellen

a)
```
SELECT l.title, p.name
FROM lecture l
        JOIN professor p ON l.givenBy = p.persNo
```

b)
```
SELECT p.name
FROM professor p
        JOIN lecture l ON l.givenBy = p.persNo
WHERE l.title="Logics"
```

c)
```
SELECT p.name, a.name
FROM professor p
        JOIN assistant a ON a.worksFor = p.persNo
```

Aufgabe 3: Verbinden mehrerer Tabellen

a)
```
SELECT s.name, l.title
FROM student s
        JOIN attends a ON a.studNo = s.studNo
        JOIN lecture l ON l.courseNo = a.courseNo
```

b)
```
SELECT s.name, l.title, p.name
FROM student s
        JOIN attends a ON a.studNo = s.studNo
        JOIN lecture l ON l.courseNo = a.courseNo
        JOIN professor p ON p.persNo = l.givenBy
```

Die mehrfach auftretenden Verbindungen resultieren daraus, dass die Studierenden ihre Professoren in verschiedenen Vorlesungen sehen.

c)
```
SELECT s.name, l.title, e.grade
FROM examines e
        JOIN lecture l ON l.courseNo = e.courseNo
        JOIN student s ON s.studNo = e.studNo
```

Aufgabe 4: Es sind 11 Vorlesungen, 14 Verbindungen Vorlesung-Studierende (Tabelle `attends`), und 8 Studierende. Dies gibt insgesamt $11 \cdot 14 \cdot 8 = 1232$ Kombinationsmöglichkeiten. Alle diese werden ausgegeben. Es fehlen die `ON`-Bedingungen bei den beiden `JOIN`s (vgl. Aufgabe 3a)).

Aufgabe 5: NATURAL JOIN

a)
```
SELECT s.name, l.title
FROM student s
        NATURAL JOIN attends
        NATURAL JOIN lecture l
```

b)
```
SELECT s.name, l.title, p.name
FROM student s
        NATURAL JOIN attends a
        NATURAL JOIN lecture l
        JOIN professor p ON p.persNo = l.givenBy
```

c)
```
SELECT student.name, lecture.title, examines.grade
FROM examines
        NATURAL JOIN lecture
        NATURAL JOIN student
```

Aufgabe 6: LEFT/RIGHT JOIN

a)
```
SELECT l.title, p.name
FROM lecture l
        LEFT JOIN professor p ON l.givenBy = p.persNo
```

b)
```
SELECT p.name, a.name
FROM professor p
        LEFT JOIN assistant a ON a.worksFor = p.persNo
```

c)
```
SELECT s.name
FROM student s
        NATURAL LEFT JOIN attends a
WHERE a.courseNo IS NULL
```

d)
```
SELECT lp.title AS Voraussetzung, la.title AS Folgevorlesung
FROM  lecture lp
        LEFT JOIN requires r ON lp.courseNo = r.prereq
        LEFT JOIN lecture la ON la.courseNo = r.advanced
```

Falls nur in einer Zeile `LEFT` ergänzt wird und nicht in beiden, sorgt die andere Zeile dafür, dass nur die Schnittmenge mit der Tabelle `requires` ausgegeben wird. Da diese nur die Verbindungsinformation für Vorlesungen enthält, die eine Voraussetzung haben oder eine Voraussetzung sind, werden jeweils alle anderen Vorlesungen aus der Ausgabe entfernt.

e)
```
SELECT l.title, s.name
FROM   lecture l
         LEFT JOIN attends a ON l.courseNo = a.courseNo
         LEFT JOIN student s ON s.studNo = a.studNo
```