TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS19/20

Christoph Anneser, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1920/grundlagen/

Blatt Nr. 05

Tool zum Üben von SQL-Anfragen: https://hyper-db.com/interface.html.

Hausaufgabe 1

Diese Aufgabe ist die zweite in einer Reihe von Aufgaben, in denen Sie lernen werden, einen gegebenen Sachverhalt zu analysieren, geeignete Modelle zu entwerfen, diese in ein Datenbankschema zu überführen, das Schema in einer Datenbank aufzusetzen und mit Daten zu füllen. Sie werden ebenfalls lernen, wie Sie die Datenbank für bestimmte Anfragen optimieren können.

Benutzen Sie im Folgenden Ihr ER-Modell des Onlineshops aus dem vorangegangenen Blatt. Falls Sie nicht im Besitz einer zufriedenstellenden Lösung sind, können Sie auch die am Freitag Nachmittag veröffentlichte Beispiellösung als Ausgangspunkt verwenden.

a) Übertragen Sie das ER-Modell in ein relationales Schema.

b) Verfeinern Sie das relationale Schema soweit möglich durch Eliminierung von Relationen.

c) Finden Sie für die Attribute sinnvolle und von Postgres unterstützte Datentypen. Beziehen Sie auch die Abhängigkeiten zwischen den einzelnen Relationen in Ihre Überlegungen ein. Eine Liste mit unterstützten Datentypen finden Sie hier: (https://www.postgresql.org/docs/current/datatype.html).

Lösung

a) Erstellen des relationalen Schemas

Die initiale Überführung ergibt folgende Relationen für die Entitytypen:

$$\begin{aligned}
\text{Kontinente} &: \{[\underline{\text{KontinentNr}}, \text{Name}]\} && (1)\\
\text{Länder} &: \{[\underline{\text{Ländercode}}, \text{Name}]\} && (2)\\
\text{Kunden} &: \{[\underline{\text{KundenNr}}, \text{Name}, \text{Adresse}, \text{Geschlecht}, \text{Geburtsdatum}]\} && (3)\\
\text{Bestellungen} &: \{[\underline{\text{BestellNr}}, \text{Datum}, \text{Gesamtstatus}, \text{Gesamtbetrag}]\} && (4)\\
\text{Bestellpositionen} &: \{[\underline{\text{BestellNr}, \text{Position}}, \text{Anzahl}, \text{Preis}, \text{Steuern}, \text{Status}]\} && (5)\\
\text{Artikel} &: \{[\underline{\text{ArtikelNr}}, \text{Name}, \text{Marke}, \text{Typ}, \text{UVP}]\} && (6)\\
\text{Lieferanten} &: \{[\underline{\text{LieferantenNr}}, \text{Adresse}, \text{Name}]\} && (7)
\end{aligned}$$

Für die Beziehungstypen werden folgende Relationen erstellt:

$$\text{liegt\_in} : \{[\underline{\text{Ländercode}}, \text{KontinentenNr}]\} \tag{8}$$
$$\text{wohnt\_in} : \{[\underline{\text{KundenNr}}, \text{KontinentenNr}]\} \tag{9}$$
$$\text{aufgeben} : \{[\underline{\text{BestellNr}}, \text{KundenNr}]\} \tag{10}$$
$$\text{enthalten} : \{[\underline{\text{BestellNr}, \text{Position}}]\}, \tag{11}$$
$$\text{bereitstellen} : \{[\underline{\text{LieferantenNr}, \text{ArtikelNr}}, \text{verfügbareAnz.}, \text{Lieferpreis}]\} \tag{12}$$
$$\text{liefert\_nach} : \{[\underline{\text{LieferantenNr}, \text{Ländercode}}]\} \tag{13}$$
$$\text{sitzt\_in} : \{[\underline{\text{LieferantenNr}}, \text{Ländercode}]\} \tag{14}$$

Ein Sonderfall stellt die Beziehung *liefern* dar. Nach ER-Diagramm gibt es zwei Möglichkeiten, sie in eine Relation zu übersetzen:

$$\text{liefern}_a : \{[\underline{\text{ArtikelNr}, \text{BestellNr}, \text{Position}}, \text{LieferantenNr}]\} \tag{15}$$
$$\text{liefern}_b : \{[\underline{\text{LieferantenNr}, \text{BestellNr}, \text{Position}}, \text{ArtikelNr}]\} \tag{16}$$

Wir haben allerdings die zusätzliche Bedingung an unser Modell, dass eine Bestellposition nur genau einen Artikel und genau einen Lieferanten enthalten darf (siehe Blatt 4, Aufgabe 1e). Da diese im ER-Diagramm nicht durch Funktionalitätsangaben darstellbar ist, wird sie auch in den beiden obigen Übersetzungen nicht modelliert. Hier könnte ein Lieferant pro Bestellposition mehrere Artikel bereitstellen (15) oder ein Artikel einer Bestellposition von mehreren Lieferanten geliefert werden (16).

Da das relationale Modell (im Gegensatz zum ER-Diagramm) aber abbilden kann, dass eine Bestellposition nur genau einen Artikel von einem Lieferanten umfasst, übersetzen wir *liefern* entsprechend:

$$\text{liefern} : \{[\underline{\text{BestellNr}, \text{Position}}, \text{LieferantenNr}, \text{ArtikelNr}]\} \tag{17}$$

b) Verfeinerung des relationalen Schemas

Als Nächstes wird das relationale Schema verfeinert, indem Relationen zusammengefasst werden.

Dabei werden Relationen für binäre Beziehungstypen mit Relationen für Entitytypen zusammengefasst, falls diese gleiche Schlüssel besitzen und es sich dabei um 1:N, N:1 oder 1:1 Beziehungen handelt.

So kann die Relation *liegt_in* (8) in die *Länder*-Relation (2) aufgenommen werden. *wohnt_in* (9) wird mit *Kunden* (3) zusammengefasst. *liefern* (17) wird mit *Bestellpositionen* (5) zusammengefasst.

Auch die *sitzt_in*-Relation (14) wird mit der *Lieferanten*-Relation (7) zusammengefasst. Die Relationship *aufgeben* (10) wird zu *Bestellungen* (4) hinzugefügt. Des Weiteren kann die Relation *enthalten* (11) zu *Bestellpositionen* (5) hinzugefügt werden, da sie den gleichen Schlüssel besitzen und es sich um eine 1:N Beziehung handelt. Da *enthalten* keine weitere Information speichert, kann die Relation einfach entfernt werden.

Durch die oben genannten Änderungen erhält man folgendes Schema:

$$
\begin{aligned}
\text{Kontinente} &: \{[\underline{\text{KontinentenNr}}, \text{Name}]\} && (18)\\
\text{Länder} &: \{[\underline{\text{Ländercode}}, \text{Name}, \text{KontinentNr}]\} && (19)\\
\text{Kunden} &: \{[\underline{\text{KundenNr}}, \text{Name}, \text{Adresse}, \text{Geschlecht}, \text{Geburtsdatum}, \text{Ländercode}]\} && (20)\\
\text{Bestellungen} &: \{[\underline{\text{BestellNr}}, \text{Datum}, \text{Gesamtstatus}, \text{Gesamtbetrag}, \text{KundenNr}]\} && (21)\\
\text{Bestellpositionen} &: \{[\underline{\text{BestellNr}, \text{Position}}, \text{Anzahl}, \text{Preis}, \text{Steuern}, \text{Status}, \text{ArtikelNr}, \text{LieferantenNr}]\} && (22)\\
\text{Artikel} &: \{[\underline{\text{ArtikelNr}}, \text{Name}, \text{Marke}, \text{Typ}, \text{UVP}]\} && (23)\\
\text{Lieferanten} &: \{[\underline{\text{LieferantenNr}}, \text{Adresse}, \text{Name}, \text{Ländercode}]\} && (24)\\
\text{liefert\_nach} &: \{[\underline{\text{LieferantenNr}, \text{Ländercode}}]\} && (25)\\
\text{bereitstellen} &: \{[\underline{\text{LieferantenNr}, \text{ArtikelNr}}, \text{verfügbareAnz.}, \text{Lieferpreis}]\} && (26)
\end{aligned}
$$

c) Datentypen für die einzelnen Attribute

- **integer**:
  - Kontinente.KontinentenNr
  - Länder.Ländercode
  - Kunden.KundenNr
  - Bestellungen.BestellNr
  - Bestellpositionen.{Position,Anzahl}
  - Artikel.ArtikelNr
  - Lieferanten.LieferantenNr
  - bereitstellen.verfügbareAnz
- **date**:
  - Kunden.Geburtsdatum
- **timestamp**:
  - Bestellungen.Datum
- **text**:
  - Kontinente.Name
  - Länder.Name
  - Kunden.{Name, Adresse}
  - Bestellungen.Gesamtstatus
  - Bestellposition.Status
  - Artikel.{Name, Marke, Typ}
  - Lieferanten.{Name, Adresse}
- **money**:

- Bestellungen.Gesamtbetrag
- Bestellpositionen.{Preis, Steuern}
- Artikel.UVP
- bereitstellen.Lieferpreis

- **char(1)**:
  - Kunden.Geschlecht

Hausaufgabe 2

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse duplikatfrei aus.

(a) Finden Sie die *Studenten*, die Sokrates aus *Vorlesung(en)* kennen.

(b) Finden Sie die *Studenten*, die *Vorlesungen* hören, die auch Fichte hört.

(c) Finden Sie die *Assistenten* von *Professoren*, die den Studenten Carnap unterrichtet haben – z.B. als potentielle Betreuer seiner Bachelorarbeit.

(d) Geben Sie die Namen der *Professoren* an, die Theophrastos aus *Vorlesungen* kennt.

(e) Welche *Vorlesungen* werden von *Studenten* im Bachelorstudium (1. – 6. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

(f) Bestimmen Sie für jede Vorlesung wie viele Studenten diese hören. Geben Sie auch Vorlesungen ohne Hörer aus. Sortieren Sie das Ergebnis absteigend nach Anzahl der Hörer.

Lösung:

(a) Finden Sie die *Studenten*, die Sokrates aus *Vorlesung(en)* kennen.

```
select distinct s.Name, s.MatrNr
from Studenten s, hoeren h, Vorlesungen v,
   Professoren p
where s.MatrNr = h.MatrNr
  and h.VorlNr = v.VorlNr
  and v.gelesenVon = p.PersNr
  and p.Name ='Sokrates';
```

(b) Finden Sie die *Studenten*, die *Vorlesungen* hören, die auch Fichte hört.

```
select distinct s1.Name, s1.MatrNr
from Studenten s1, Studenten s2, hoeren h1, hoeren h2
where s1.MatrNr = h1.MatrNr
  and s1.MatrNr != s2.MatrNr
  and s2.MatrNr = h2.MatrNr
  and h1.VorlNr = h2.VorlNr
  and s2.Name ='Fichte';
```

(c) Finden Sie die *Assistenten* von *Professoren*, die den Studenten Carnap unterrichtet haben – z.B. als potentielle Betreuer seiner Bachelorarbeit.

```
select distinct a.Name, a.PersNr
from Assistenten a, Professoren p, Vorlesungen v,
   hoeren h, Studenten s
where a.Boss = p.PersNr
```

```
  and p.PersNr = v.gelesenVon
  and v.VorlNr = h.VorlNr
  and h.MatrNr = s.MatrNr
  and s.Name ='Carnap';
```

(d) Geben Sie die Namen der *Professoren* an, die Theophrastos aus *Vorlesungen* kennt.

```
select distinct p.PersNr, p.Name
from Professoren p, hoeren h, Vorlesungen v,
   Studenten s
where p.PersNr = v.gelesenVon
and v.VorlNr = h.VorlNr
and h.MatrNr = s.MatrNr
and s.Name ='Theophrastos';
```

(e) Welche *Vorlesungen* werden von *Studenten* im Bachelorstudium (1. – 6. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

```
select distinct v.Titel
from Vorlesungen v, hoeren h, Studenten s
where v.VorlNr = h.VorlNr
and h.MatrNr = s.MatrNr
and s.Semester between 1 and 6;
```

(f) Bestimmen Sie für jede Vorlesung wie viele Studenten diese hören. Geben Sie auch Vorlesungen ohne Hörer aus. Sortieren Sie das Ergebnis absteigend nach Anzahl der Hörer.

```
select v.VorlNr, v.Titel, count(h.MatrNr) as hoerer
from
  Vorlesungen v left outer join
  hoeren h on (v.VorlNr = h.VorlNr)
group by v.VorlNr, v.Titel
order by hoerer desc;
```

Hausaufgabe 3

Bestimmen Sie für alle Studenten eine gewichtete Durchschnittsnote ihrer Prüfungen. Die Gewichtung der einzelnen Prüfungen erfolgt gemäß dem Vorlesungsumfang (SWS). Dies entspricht dem Verfahren der Durchschnittsnotenberechnung für Ihr Bachelor-Zeugnis.

HINWEIS: Wenn Sie das Ergebnis Ihrer Anfrage besser testen wollen, können Sie eine größere *prüfen*-Relationenausprägung benutzen. Fügen Sie dazu das folgende SQL-Statement vor ihrer Anfrage ein. Verwenden Sie dann `pruefenxl` statt `pruefen`:

```
with pruefenxl(matrnr,vorlnr,persnr,note) as (
    select * from pruefen
    union all
    values
        (25403, 5049, 2126, 1),
        (26120, 5001, 2137, 1),
        (26120, 5043, 2126, 3),
        (26120, 5052, 2126, 4),
        (26120, 4630, 2137, 1)
)
```

Lösung:

```
select
    s.matrnr,
    s.name,
    sum(p.note * v.sws) / sum(v.sws) as durchschnitt
from studenten s, pruefenxl p, vorlesungen v
where s.matrnr = p.matrnr and p.vorlnr = v.vorlnr
group by s.matrnr, s.name
```

Hausaufgabe 4

Folgender Ausdruck im Tupelkalkül gibt alle Studenten aus, die alle von ihnen gehörten Vorlesungen bestanden haben.

$$\{s \mid s \in \text{Studenten} \wedge \\ \forall h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \Rightarrow \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

Übersetzen Sie diese Anfrage nun in SQL. Da SQL keine Allquantoren und Implikationen unterstützt, müssen Sie sie dazu zunächst umformen.

a) Formen Sie den Ausdruck in einen Äquivalenten um, der keine Implikationen oder Allquantoren verwendet.

b) Übersetzen Sie den so erlangten Ausdruck in SQL. Testen Sie ihn in der Webschnittstelle.

Lösung:

a) Wir formen zunächst die innere Implikation um, denn $A \Rightarrow B \iff \neg A \vee B$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \forall h \in \text{hoeren}(h.\text{MatrNr} \neq s.\text{MatrNr} \vee \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

Wir ersetzen nun den Allquantor durch einen negierten Existenzquantor, denn $\forall x(P(x)) \iff \neg\exists x(\neg P(x))$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \neg\exists h \in \text{hoeren}(\neg(h.\text{MatrNr} \neq s.\text{MatrNr} \vee \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4)))\}$$

Zuletzt wenden wir De Morgans Regel an, um die Negation nach innen zu ziehen, denn $\neg(A \vee B) \iff \neg A \wedge \neg B$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \neg\exists h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \wedge \\ \neg\exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

b) Die Anfrage kann nun eins zu eins in SQL übersetzt werden, wobei jeder logische Operator einfach durch seine SQL-Entsprechung ersetzt wird:

```
select * from Studenten s
where not exists (select * from hoeren h
    where h.MatrNr = s.MatrNr
    and not exists (select * from pruefen p
        where p.MatrNr = s.MatrNr
        and p.VorlNr = h.VorlNr
        and p.Note <= 4
    )
)
```

Zusatzaufgabe 5

Diese Aufgabe wird nicht in der Übung besprochen. Bei Installationsschwierigkeiten besuchen Sie bitte die Tutorsprechstunde am Mittwoch nach der Vorlesung.

In dieser Aufgabe richten Sie ein Datenbanksystem auf Ihrem eigenen Rechner ein, um später auch komplexe Anfragen an unseren Onlineshop aus den vorherigen Aufgaben auswerten zu können. Wir empfehlen hierfür Postgres, da es die gleiche Syntax wie HyPer benutzt, quellenoffen ist und vergleichsweise einfach installiert werden kann.

Die folgende Installationsanweisungen beziehen sich auf Ubuntu, wobei Postgres auch auf Windows und MacOS installiert werden kann. **Bei Installationsschwierigkeiten besuchen Sie bitte die Tutorsprechstunde am Mittwoch nach der Vorlesung**.

```
$ sudo apt install postgresql
```

Den Datenbankserver kann man nun mit folgenden Befehlen steuern:

```
$ sudo systemctl (start|stop|restart|reload|status) postgresql
```

Wir legen nun einen neuen Benutzer an, mit dem wir uns künftig in `psql` einloggen werden. Als Nutzernamen benutzen Sie am besten Ihren Username, mit dem Sie auch in Ubuntu eingeloggt sind.

```
sudo -u postgres createuser -P <nutzername>
```

Im nächsten Schritt wechseln wir wieder zum User `postgres` und können nun unsere erste Datenbank anlegen, die wir `universitaet` nennen. Außerdem müssen wir den neuen Benutzer berechtigen, diese Datenbank verwenden zu dürfen:

```
$ sudo -u postgres psql

postgres=# CREATE DATABASE universitaet;

postgres=# GRANT ALL ON DATABASE universitaet to <nutzername>;
```

Die `psql`-Umgebung kann mit `\q` wieder verlassen werden.

Wir importieren nun das aus der Vorlesung bekannte Unischema. Dazu verlassen wir `psql` und laden zunächst die folgende SQL Datei von der Vorlesungswebsite herunter:

```
$ wget https://db.in.tum.de/teaching/ws1920/grundlagen/uni_mysql.sql
```

Nun erstellen wir ein neues Schema in der Datenbank und importieren alle Tabellen und Daten, die in der Datei `uni_mysql.sql` angegeben sind.

```
$ psql universitaet < uni_mysql.sql
```

Ab sofort können wir uns über `psql` mit der Datenbank verbinden:

```
$ psql universitaet
```

Nun können wir in der Datenbank neue Tabellen erstellen, diese mit Daten befüllen und Anfragen ausführen.