Einführung in Datenbanken

Übung 5: Relationaler DB-Entwurf, Einfache Anfragen

Prof. Dr. Stefan Brass
PD Dr. Alexander Hinneburg
Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Inhalt

DBMS auf eigenem Rechner?

- Haben Sie PostgreSQL auf einem eigenen Rechner installiert?

  Ja/Nein-Umfrage.

- Gab es Schwierigkeiten bei der Installation?
- Haben Sie ein anderes DBMS auf einem eigenen Rechner?

  A. Oracle, Microsoft SQL Server oder IBM DB2
  B. MySQL/MariaDB
  C. SQLite
  D. Sonstiges (welches?)
  E. Keins

  Bei mehreren kreuzen Sie das erste an.

Fremdschlüssel in MySQL/MariaDB (1)

- In MySQL/MariaDB muss man die Spalten der referenzierten Relation angeben (auch für Verweis auf Primärschlüssel):

```
CREATE TABLE BEWERTUNGEN(
                SID      NUMERIC(3)    NOT NULL,
                ATYP     CHAR(1)       NOT NULL,
                ANR      NUMERIC(2)    NOT NULL,
                PUNKTE   NUMERIC(4,1)  NOT NULL,
                PRIMARY KEY(SID, ATYP, ANR),
                FOREIGN KEY(SID) REFERENCES
                        STUDENTEN(SID),
                FOREIGN KEY(ATYP,ANR) REFERENCES
                        AUFGABEN(ATYP,NR))
```

- Nach dem SQL-Standard und in den anderen mir bekannten Systemen kann man das tun, muss es aber nicht.

Fremdschlüssel in MySQL/MariaDB (2)

- `create table r(a numeric(1) not null,`
  `primary key(a));`
- `insert into r values (1);`
- `create table s(a numeric(1) not null,`
  `foreign key(a) references r);`
  `ERROR 1005 (HY000): Can't create table 'sb.s'`
  `(errno: 150)`
- `create table s(a numeric(1) not null,`
  `foreign key(a) references r(a));`
  Das funktioniert.
- Was könnte man jetzt tun, um die Wirkung des Fremdschlüssels auszuprobieren?

Fremdschlüssel in MySQL/MariaDB (3)

- r(a)
  s(a → r)
- r enthält eine Zeile, und s ist bisher leer.

r
a
1

- Wie prüfen Sie, ob der Fremdschlüssel überwacht wird?

A. `insert into s values (1);`

B. `insert into s values (2);`

C. `insert into r values (2);`

D. `delete from r where a = 1;`

Fremdschlüssel in MySQL/MariaDB (4)

- `insert into s values (2);`
  `ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (‘sb‘.‘s‘, CONSTRAINT ‘s_ibfk_1‘ FOREIGN KEY (‘a‘) REFERENCES ‘r‘ (‘a‘))`

  In MySQL werden Delimited Identifier mit „Backticks“ geschrieben: ‘...‘.

- Es hängt an der Storage Engine, ob Fremdschlüssel überwacht werden.

  Inzwischen ist InnoDB Default, die Fremdschlüssel unterstützt.

- Wenn man die kürzere „Column Constraint“ Syntax für Fremdschlüssel wählt ($\rightarrow$ Sommersemester) ignoriert MariaDB 5.5.68 stillschweigend den Fremdschlüssel!

  `create table s(a numeric(1) not null references r(a));`

  Dann gibt es die obige Fehlermeldung nicht. FS fehlt in `show create table s`

Delimited Identifier in MySQL/MariaDB

- `create table "r"("a" numeric(1));`

  ```
  ERROR 1064 (42000): You have an error in your
  SQL syntax; check the manual that corresponds
  to your MariaDB server version for the right
  syntax to use near '"r"("a" numeric(1))' at
  line 1
  ```

  In MySQL/MariaDB wird normalerweise `"..."` als Zeichenkette interpretiert. Man kann die Option („SQL Mode“) `ANSI_QUOTES` setzen, damit sich MySQL/MariaDB in dieser Beziehung standard-konform verhält.

- Z.B. in PostgreSQL funktioniert das problemlos.

  PostgreSQL versteht die „Backticks“ `` `...` `` nicht, die sind aber auch im Standard nicht vorgesehen.

Inhalt

Hausaufgabe 3: Relationaler DB-Entwurf (1)

- Entwickeln Sie ein relationales Datenbank-Schema für Raumreservierungen in der Universität.
- Für jeden Raum (z.B. Hörsaal)
  - Gebäude (Kürzel), z.B. `VSP1`.
  - Raum-Nummer, z.B. 328.
  - Anzahl Plätze unter Corona-Bedingungen (Kapazität), z.B. 33.
  - Gibt es ein automatisches Aufzeichnungssystem (ja/nein)?

  Räume werden identifiziert über Gebäude und Raum-Nummer.
- `RAUM(GEB, RNR, PLAETZE, AUFZEICHNUNG)`

Hausaufgabe 3: Relationaler DB-Entwurf (2)

Alternativen:

- `Raum(RaumNr,Gebaeude,AnzahlPlaetze,autoRecord)`
  - Ich würde die größere Einheit „`Gebaeude`“ vor der untergeordneten Einheit „`RaumNr`“ anordnen.
  - Die von Java bekannte „Kamelhöcker-Schreibweise“ (Gross/Klein) funktioniert bei SQL nicht gut:
    Im Systemkatalog steht alles in Kleinbuchstaben.

    Bei PostgreSQL. Bei Oracle alles in Großbuchstaben. Besser „_“.
- `Räume(Gebäude, RaumNummer, Kapazität, Aufzeichnungssystem)`
  - Ich würde in den Tabellen- und Spaltennamen besser nur ASCII-Zeichen verwenden, keine Umlaute.

    Aber vielleicht ist das eine Frage des Alters. Inzwischen müsste es ja gehen. Eventuell Probleme mit ISO Latin 1 vs. UTF-8?

Hausaufgabe 3: Relationaler DB-Entwurf (3)

- Dozenten und Dozentinnen (Lehr-Personal) als Ansprechpartner für Lehrveranstaltungen:
  - Nummer zur eindeutigen Identifizierung
  - Anrede (Herr/Frau/...) (kann auch leer bleiben) (oder null)
  - Titel (z.B. „Prof. Dr.“)
  - Vorname
  - Name
  - EMail-Adresse
  - Telefon (optional)
  - Institut
- `DOZENT(DNR, ANREDE, TITEL, VORNAME, NACHNAME, EMAIL, TEL°, INSTITUT)`

Hausaufgabe 3: Relationaler DB-Entwurf (4)

Alternativen:

- `DozentIn(Nummer, Anrede, Titel, Vorname, Name, E-Mail-Adresse, Telefon°, Institut)`
  - Die Datenbank macht daraus `dozentin` oder `DOZENTIN` (nicht geschlechtsneural)
  - `DOZENT_IN`, `"DozentIn"`, `"DOZENT*IN"`, `"DOZENT/IN"`, `LEHRENDE` (Plural!), `DOZ`, `LEHRPERSONAL`, `LEHRPERSON`

    Was würden Sie in Java als Klassennamen verwenden? DB vs. UI.
  - ```
    COMMENT ON TABLE DOZENT IS
    'Gemeint sind Dozentinnen und Dozenten ' ||
    '(generisches Maskulinum)';
    ```

    Kann man im Systemkatalog nachschauen. Nicht im SQL Standard.
  - Bindestrich ginge nicht in SQL-Bezeichnern
    ```

Hausaufgabe 3: Relationaler DB-Entwurf (5)

- Für jede Lehrveranstaltung:
  - Nummer der Lehrveranstaltung (zur eindeutigen Identifikation).
  - Name der Lehrveranstaltung
  - Dozent/Dozentin der Lehrveranstaltung (Ansprechpartner)
- LV(LNR, NAME, DNR $\rightarrow$ DOZENT)

Hausaufgabe 3: Relationaler DB-Entwurf (6)

Alternativen:

- `Lehrveranstaltung(Nummer_Lehrveranstaltung, Personal_Nummer→Lehr_Personal, Bezeichnung)`
  - Etwas lang.
  - Besser konsistent „`Nummer`“ vorne oder hinten.
- `lehrveranstaltung(LEHRNO, name, ANSPRECHPARTNER°→dozenten)`
  - „`NO`“: nur in Englisch übliche Abkürzung (lat. „numero“).
  - Obwohl Groß-/Kleinschreibung letztendlich vom DBMS eliminiert wird: In der Dokumentation bitte einheitlich.
  - Singular/Plural für Tabellennamen konsistent verwenden.
  - Nullwert für Ansprechpartner ist unzulässige Verbesserung.

Hausaufgabe 3: Relationaler DB-Entwurf (7)

- Die Reservierung der Räume. In dieser Aufgabe geht es nur um regelmäßige Reservierungen für alle Semesterwochen, z.B. immer montags, 10:30–12:00.
  - Raum
  - Wochentag (Mo, Di, . . . , Sa, So)
  - Start-Uhrzeit

    Die Reservierung erfolgt dann immer für 90 min, deswegen braucht keine End-Uhrzeit gespeichert zu werden.
  - Lehrveranstaltung.
  - Optionaler Kommentar, z.B. „Übungsgruppe 2“.
- BELEGUNG((GEB, RNR) $\rightarrow$ RAUM, TAG, ZEIT, LNR $\rightarrow$ LV, KOMMENTAR$^\circ$)

Hausaufgabe 3: Relationaler DB-Entwurf (8)

Beispiel-Abgaben:

- Reservierung((facility,room) → Raum,
  startTime, dayOfWeek,
  class → Lehrveranstaltung,
  comment°)
  - Englisch ist prinzipiell möglich, aber dann bitte einheitlich.

    In einem Softwareprojekt macht die Internationalisierung wohl Sinn, aber wenn der Kunde/Anwender deutsch spricht, haben Sie eventuell Probleme, Fachbegriffe zu übersetzen. Vielleicht sollte man die deutschen Worte wenigstens in der Dokumentation nennen, wenn es nicht offensichtlich ist.
  - Welche Möglichkeit gibt dieser Schlüssel?

Hausaufgabe 3: Relationaler DB-Entwurf (9)

Beispiel-Abgaben, Forts.:

- RESERVIERUNG((GEBÄUDE,NR) → Raum,
  TAG, STARTZEIT,
  LEHRVERANSTALTUNG→LEHRVERANSTALTUNG,
  KOMMENTAR°)
  - Warum ist der Schlüssel falsch?
  - Im Prinzip ist möglich, dass Spalten wie Tabellen heißen. Man sollte das aber besser vermeiden.
  - Es ist nicht verlangt, dass die Namen von Fremdschlüssel-Attributen mit den Namen der Primärschlüssel-Attribute übereinstimmen.

    Es ist weder in SQL noch in dieser Notation verlangt, dass die Spalten beim Fremdschlüssel angegeben werden (bei Referenz auf Primärschlüssel).

Hausaufgabe 3: Relationaler DB-Entwurf (10)

- „Eine Lehrveranstaltung kann mehrere wöchentliche Termine haben.“
  - Deswegen kann die LV-Nummer nicht Schlüssel sein.
- „Ein Raum kann zur gleichen Zeit nur durch eine Lehrveranstaltung belegt sein.“
  - Deswegen reicht Raum-Identifikation (Gebäude, Nr), Wochentag und Zeit zur eindeutigen Identifikation.
- „Ein Raum kann zu verschiedenen Zeiten natürlich von unterschiedlichen Veranstaltungen belegt sein.“
  - Deswegen ist die Raum-Identifikation alleine nicht Schlüssel.
- Es versteht sich von selbst, dass zur gleichen Zeit in unterschiedlichen Räumen Lehrveranstaltungen möglich sind.

Hausaufgabe 3: Relationaler DB-Entwurf (11)

- Es wäre noch zu klären, ob eine Lehrveranstaltung gleichzeitig Termine in verschiedenen Räumen haben kann.

  Etwa mehrere parallele Übungsgruppen — aber wäre das die gleiche Lehrveranstaltung, oder ist jede Übungsgruppe eine eigene Lehrveranstaltung?

- Falls nicht, wären auch die Lehrveranstaltungs-Nummer, Tag und Zeit zusammen ein Schlüssel.

  Da kein Schlüssel den anderen logisch impliziert (keiner ist eine Obermenge des anderen), müsste man beide Schlüssel angeben. Natürlich kann nur einer Primärschlüssel sein.

- Leider sagt die Aufgabe nichts dazu.

  Sie hätten natürlich fragen können.

Hausaufgabe 3: Relationaler DB-Entwurf (12)

- Mit Schlüsseln kann man überlappende Termine nicht ausschließen.

  Der Schlüssel garantiert nur, dass es nicht zwei Einträge für den gleichen Raum am gleichen Tag mit der gleichen Start-Uhrzeit gibt. Eine Start-Uhrzeit eine Minute später wäre aber möglich. Das könnte man z.B. mit Triggern verhindern, die in der Fortsetzungs-Vorlesung im Sommer besprochen werden. Trigger sind Programmstücke, die bei einem auslösenden Ereignis (z.B. einer Einfügung in die Reservierungs-Tabelle) automatisch ausgeführt werden.

- Bei der Aufgabe wird angenommen, dass durch das feste Zeitraster und die Standard-Dauer von 90min partiell überlappende Termine ausgeschlossen werden.

  Es war nicht verlangt, das Zeitraster zu überwachen. Das wäre aber möglich mit einem Fremdschlüssel auf eine Tabelle `ZEIT_RASTER(ZEIT)`, in der nur die erlaubten Zeiten eingetragen sind. Alternative: `CHECK`-Constraint ($\rightarrow$ Sommer).

Hausaufgabe 3: CREATE TABLE (1)

```
CREATE TABLE RESERVIERUNG(
        GEBAUEDE  VARCHAR(10)    NOT NULL,
        RAUMNR    NUMERIC(4)     NOT NULL,
        WOCHENTAG CHAR(2)        NOT NULL,
        STARTZEIT NUMERIC(4)     NOT NULL,
        LVNR      NUMERIC(7)     NOT NULL,
        KOMMENTAR VARCHAR(255),
        PRIMARY KEY
            (GEBAUEDE, RAUMNR, WOCHENTAG, STARTZEIT),
        FOREIGN KEY (GEBAUEDE,RAUMNR)
            REFERENCES RAUM,
        FOREIGN KEY (LVNR)
            REFERENCES LEHRVERANSTALTUNG,
        CHECK(STARTZEIT IN (0730, 0900, 1030, 1200,
                            1330, 1500, 1630, 1800)));
```

Hausaufgabe 3: CREATE TABLE (2)

- Es war dazu geschrieben:

  ```
  // Kommentar: Startzeit im Format HHMM
  ```

  Das ist ein Syntaxfehler. Kommentar in SQL: `--`.

- Der `CHECK`-Constraint war natürlich nicht verlangt.

  Er stellt sicher, dass Startzeit nur einen der explizit aufgelisteten Werte annehmen kann. SQL verwendet hier `(...)` als Mengenklammern $\{...\}$. `IN`-Bedingungen werden in dieser Vorlesung noch für Anfragen besprochen.

- Man könnte auch den Typ `CHAR(5)` mit Konstanten der Form `'09:00'` verwenden.

  Man muss die Stunden dann immer zweistellig schreiben, damit die Sortierung klappt: `'9:00' > '18:00'` gilt in PostgreSQL (wegen `'9' > '1'`) und auch `'1:00' > '12:00'` („:“ steht im ASCII-Code nach den Ziffern).

Hausaufgabe 3: CREATE TABLE (3)

- Besser ist natürlich der Datentyp `TIME`.

  [https://www.postgresql.org/docs/9.3/datatype-datetime.html]

- Er enthält bei PostgreSQL Stunden, Minuten, Sekunden, und Microsekunden (6 Nachkommastellen).
  Gültige Werte sind von 00:00:00 bis 24:00:00.

  Man kann die Anzahl der Nachkommastellen für die Sekunden angeben, z.B. auch `TIME(0)`, wenn nur ganze Sekunden gespeichert werden sollen. Nach dem SQL-92 Standard sollte das der Default sein, aber bei PostgreSQL bekommt man 6 Nachkommastellen, wenn man nichts angibt. Es gibt auch `TIME(`p`) WITH TIME ZONE`.

- Konstanten werden offiziell geschrieben in der Form
  `TIME 'hh:mm:ss'` bzw. `TIME 'hh:mm:ss.xxxxxx'`
  Bei PostgreSQL kann man das Schlüsselwort weglassen, auch bei den Strings akzeptiert es mehr.

Inhalt

Präsenzaufgabe: PQ-Formel in SQL (1)

- Es seien viele Übungsaufgaben zur Bestimmung der Nullstellen von Parabeln der folgenden Form gegeben:

$$y = x^2 + px + q$$

- Die Aufgaben stehen in folgender Tabelle:

PARABEL		
ID	P	Q
1	-3	2
2	2	2
3	2	1
4	1	-12

- Schreiben Sie eine SQL-Anfrage, die für jede Parabel die Nullstellen bestimmt, soweit möglich (zwei Spalten für x_1, x_2).

  Verwenden Sie die bekannte „pq-Formel“: $x_{1,2} = -\left(\frac{p}{2}\right) \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$.

  Form der Anfrage: `SELECT ID, ... AS X1, ... AS X2 FROM ... WHERE ...`

Präsenzaufgabe: PQ-Formel in SQL (2)

- Lösung:

```
SELECT ID,
       -(P/2) + SQRT((P/2)^2 - Q) AS X1,
       -(P/2) - SQRT((P/2)^2 - Q) AS X2
FROM   PARABEL
WHERE  (P/2)^2 - Q >= 0
```

- PostgreSQL hat viele Operatoren definiert, u.a. „^“ für x^y.
  [https://www.postgresql.org/docs/9.3/functions-math.html] Sogar |/ für $\sqrt{}$.
- Standard-konform müsste man `POWER(P/2, 2)` schreiben.
- Natürlich ist auch `(P/2)*(P/2)` möglich.
- `P` und `Q` haben den Typ `DOUBLE PRECISION`. Wenn sie vom Typ `INTEGER` wären, müsse man `P/2.0` schreiben.

Präsenzaufgabe: PQ-Formel in SQL (3)

Anmerkungen:

- Es gibt keine Funktion `SQUARE` zum Quadrieren in PostgreSQL.
- SQL hat natürlich das monadische Minus: `-(P/2)`. Einige haben `-1 * (P/2)` geschrieben.
- Wenn man die `WHERE`-Klausel weglässt, bekommt man:
  `ERROR: cannot take square root of a negative number`

  Die Parabel mit der ID 2 hat keine Nullstelle und der Wert von `(P/2)^2 - Q` ist negativ. Es tritt sozusagen eine Exception bei der Anfrage-Ausführung auf, wenn die Funktion `SQRT` für diesen Wert aufgerufen wird.

Inhalt

Einfache SQL-Anfragen (1)

- Einfache SQL-Anfrage, Stufe 0:

```
SELECT *
FROM   ⟨Tabelle⟩
```

- Einfache SQL-Anfrage, Stufe 1a („Projektion“):

```
SELECT ⟨Spalte⟩, ..., ⟨Spalte⟩
FROM   ⟨Tabelle⟩
```

- Einfache SQL-Anfrage, Stufe 1b („Selektion“):

```
SELECT *
FROM   ⟨Tabelle⟩
WHERE  ⟨Spalte⟩ = ⟨Wert⟩
```

- Stufe 1: Alles zusammen, auch Prädikate `<`, `<=`, `>`, `>=`, `<>`, sowie `LIKE`

Einfache SQL-Anfragen (2)

- Was sind die Ergebnisse der folgenden Anfrage?

```
SELECT ID, S
FROM   TEST
WHERE  S LIKE 'A%B_'
```

- Die Tabelle ist:

TEST	
ID	S
1	A%B_
2	ABC
3	ABBBC
4	AXYZB
5	ACB␣␣

Ja/Nein-Abstimmung pro Zeile. „␣“ symbolisiert ein Leerzeichen.

Einfache SQL-Anfragen (3)

- Einfache SQL-Anfrage, Stufe 2: Wertausdrücke statt Spalten!

  `SELECT` ⟨Wertausdruck⟩`,` `...,` ⟨Wertausdruck⟩
  `FROM` ⟨Tabelle⟩
  `WHERE` ⟨Wertausdruck⟩ `=` ⟨Wertausdruck⟩

- Wertausdrücke („Terme“ in der Logik) können sein:
  - Konstanten (Datentyp-Literale), z.B. `1`, `'abc'`
  - Spaltennamen
  - Mit Datentyp-Operationen zusammengesetzte Ausdrücke:

    ⟨Wertausdruck⟩ `+` ⟨Wertausdruck⟩

    Binäre Operatoren: `+`, `-`, `*`, `/`, `||`, monadisches Minus: `-`⟨Wertausdruck⟩
  - Aufruf einer Datentyp-Funktion (teils spezielle Syntax):

    ⟨Funktion⟩`(` ⟨Wertausdruck⟩`,` `...,` ⟨Wertausdruck⟩ `)`

Einfache SQL-Anfragen (4)

- Umbenennung von Ausgabe-Spalten:

  `SELECT` ⟨Wertausdruck⟩ `AS` ⟨Spalte⟩, ...

- Funktionen z.B. `UPPER`, `TRIM`, `SQRT`, `POWER`, `ROUND`, `SIN`.
- SQL kennt die Standard-Vorrangregeln, z.B. bedeutet `A+B*C`:
  - `A+(B*C)`, und nicht: `(A+B)*C`.
- Klammern `(...)` können verwendet werden, um eine bestimmte Struktur zu erzwingen.
- Übung: Was ist das Ergebnis von `7+3*2-4-1`?

  Es kann nützlich sein, einen Operator-Baum zu zeichnen.

  "`-`" ist links-assoziativ (von links ausgewertet).

Einfache SQL-Anfragen (5)

- Einfache SQL-Anfragen, Stufe 3: Logische Verknüpfungen

  `SELECT` ⟨Wertausdruck⟩, ..., ⟨Wertausdruck⟩
  `FROM` ⟨Tabelle⟩
  `WHERE` ⟨Bedingung⟩

- Bedingungen können z.B. sein:
  - Vergleichsoperator (Prädikat), angewendet auf Wertausdrücke:
    ⟨Wertausdruck⟩ ⟨Vergleichsoperator⟩ ⟨Wertausdruck⟩
  - ⟨Bedingung⟩ `AND` ⟨Bedingung⟩
  - ⟨Bedingung⟩ `OR` ⟨Bedingung⟩
  - `NOT` ⟨Bedingung⟩
- Einfache SQL-Anfragen, Stufe 4: Mehrere Tabellen („Join“).

Inhalt

Präsenzaufgabe: System-Katalog (1)

- Der Systemkatalog eines DBMS enthält „Metadaten“, also „Daten über Daten“, und insbesondere das DB-Schema.

  Das „Information Schema“ ist Teil des SQL-Standards.

  PostgreSQL hat daneben noch seinen eigentlichen „nativen“ Systemkatalog.

- In der Tabelle `information_schema.columns` sind alle Spalten von allen Tabellen der Datenbank eingetragen.

  [https://www.postgresql.org/docs/9.3/infoschema-columns.html]

- Schreiben Sie eine Anfrage, die alle Spalten liefert von einem Typ `numeric(p,s)` mit Nachkommastellen ($s \neq 0$) und höchstens 4 Dezimalstellen ($p \leq 4$).

  Spalten: `data_type` („`numeric`“), `numeric_precision` (p), `numeric_scale` (s).

- Geben Sie `table_schema`, `table_name`, `column_name` und den Datentyp aus, z.B. `NUMERIC(4,1)`.

  Die String-Konkatenation `||` kann auch auf Zahlen angewendet werden.

Präsenzaufgabe: System-Katalog (2)

- Erwartetes Ergebnis:

table_schema	table_name	column_name	datentyp
student...	bewertungen	punkte	NUMERIC(4,1)
sakila_public	film	rental_rate	NUMERIC(4,2)
sakila_public	film_list	price	NUMERIC(4,2)
sakila_public	nicer_...	price	NUMERIC(4,2)

[https://dbs.informatik.uni-halle.de/edb?pgsql=db&username=student_gast&db=postgres&ns=]

Die Zugangsdaten unserer Installation stehen in StudIP, Reiter „Adminer“.

- Anfragen müssen nicht nur mit dem aktuellen Zustand funktionieren, sondern mit beliebigen DB-Zuständen.
- Sie können sich also nicht darauf verlassen, dass alle Ergebnis-Typen `numeric(`p`,`s`)` die Bedingung $p = 4$ erfüllen. Die Bedingung der Aufgabe ist $p \leq 4$.