

Einführung in Datenbanken

Übung 5: Relationale Schemata

Prof. Dr. Stefan Brass

PD Dr. Alexander Hinneburg

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2024/25

<http://www.informatik.uni-halle.de/~brass/db24/>

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3
- 7 Präsenzaufg.

IndustrieTag Informationstechnologie (1)

- IT² am Dienstag, 12. November, 14:15, im Raum 3.07
Kaffeetrinken und Stände in Raum 509.
- Vorträge aus IT-Industrie und Uni
- Kaffeetrinken, „Get Together“ (Ende ca. 17:30)
- Stände von Firmen, die neue Mitarbeiter rekrutieren wollen.
- Man muss/sollte sich vorher anmelden:

<https://www.uni-halle.de/uzi/veranstaltungen/39it/>

Es ist natürlich kostenlos. Es wird ein Namensschild vorbereitet. Man wird bei Kaffee und Kuchen mitgezählt. Ohne vorherige Anmeldung muss man in einer längeren Schlange stehen (um ein „ad hoc“ Namensschild zu bekommen).

IndustrieTag Informationstechnologie (2)

- Es wird am nächsten Montag eine Hausaufgabe geben, die es mehr oder weniger nötig macht, zum IT² zu kommen.

PD Dr. Hinneburg ist einer der beiden Organisatoren (im Auftrag des Instituts). Momentan sind noch recht wenige Studierenden angemeldet. Dabei wird diese Veranstaltung „für die Studierenden“ organisiert.

- Auch relativ früh im Studium kann es günstig sein, sich über aktuelle Themen und Anforderungen in der Praxis zu informieren.

Eventuell beeinflusst das Ihre Auswahl für den Wahlpflicht-Bereich.

Erste Eindrücke von möglichen Arbeitgebern sind nützlich, selbst wenn man sich nicht schon nächste Woche bewirbt (Studentenjob?).

IndustrieTag Informationstechnologie (3)

Aufgabe von Übungsblatt 5 (nächste Woche):

- Wählen Sie einen Fachvortrag vom Informatik-Industrietag, der am 12.11.2024 von 14 bis 18 Uhr in der Informatik stattfindet.

Die Vorträge sind von 14 bis 16 Uhr.

- Modellieren Sie eine im gewählten Vortrag beschriebene Anwendung, einen relevanten Teil, oder eine Beispiel-Anwendung aus dem Vortrag als relationales Datenbankschema mit etwa 2–3 Tabellen.
- Geben Sie die Tabellen mit Primär- und Fremdschlüsseln in Kurznotation wie auf Folie 6-37 an.

IndustrieTag Informationstechnologie (4)

Aufgabe von Übungsblatt 5 (nächste Woche), Forts.:

- Geben Sie weiterhin für alle Tabellen Tupel an, die verdeutlichen, was der Inhalt der einzelnen Spalten ist. Der gesamte Datenbankzustand soll etwa 8-12 Tupel umfassen.
- Beschreiben Sie zum Schluss knapp und verständlich, was die Attribute der Tabellen bedeuten sollen.
- Beschreiben Sie weiterhin die Anwendungsprozesse, die die Daten in die Tabellen einfügen, lesen und ggf. aktualisieren.

IndustrieTag Informationstechnologie (5)

Anmerkung:

- Diese Aufgabe (und eine kleinere auf dem Übungsblatt diese Woche) erfordern, dass der Datenbank-Entwurf direkt im relationalen Modell gemacht wird.
- Eigentlich wäre das professionellere Vorgehen, den Datenbank-Entwurf zuerst im Entity-Relationship-Modell zu machen (ähnlich UML Klassendiagramm), und das ER-Schema dann ins relationale Modell zu übersetzen.
- Das ist im Moment noch nicht möglich, da das ER-Modell erst gegen Ende der Vorlesung behandelt wird.

Wir wollten schnell zu praktischen SQL-Anfragen kommen, die den Kern dieser Vorlesung ausmachen. Außerdem gibt es ein Henne-Ei-Problem:
Wenn man nichts über Anfragen weiß, bleiben Entwürfe rein theoretisch.

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3
- 7 Präsenzaufg.

Datenbank: Komponisten

- Die Aufgaben dieses Übungsblattes beziehen sich auf die Komponisten-Tabelle in einer Datenbank mit Informationen über Musik-CDs (klassische Musik).

KOMPONIST				
<u>KNr</u>	Name	Vorname	geboren	gestorben
11	Händel	Georg Friedrich	1685	1759
⋮	⋮	⋮	⋮	⋮

- Sie finden Sie im **Adminer**, Schema „komponist_public“:

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

Übungsblatt 2, Aufgabe 1 (1)

- Schreiben Sie eine SQL-Anfrage, die den Vornamen des Komponisten „Vivaldi“ bestimmt.
- Versuchen Sie, sich dabei an die eingeschränkte Syntax aus **Kapitel 3** der Vorlesung zu halten.
- Das erwartete Ergebnis ist:

vorname

Antonio

- Schreiben Sie Ihre Bereitschaft, vorzurechnen, in einen Kommentar (ebenso eventuelle weitere Angaben zu Quellen und Arbeitszeit).

Da die Fähigkeit, Kommentare in SQL zu schreiben, wichtig ist, wird ein Punkt abgezogen, wenn die Angabe fehlt. Dies gilt auch für die anderen Aufgaben.

Übungsblatt 2, Aufgabe 1 (2)

- Mögliche Lösung:

```
SELECT Vorname
FROM   Komponist
WHERE  Name = 'Vivaldi'
```

- Die Groß-/Kleinschreibung von Schlüsselworten wie „**SELECT**“ und Tabellen- und Spaltennamen ist egal:

```
select vorname
from   komponist
where  name = 'Vivaldi'
```

- Bei String-Konstanten ist die Groß-/Kleinschreibung dagegen wichtig. Mit '**vivaldi**' wäre das Ergebnis leer.

Übungsblatt 2, Aufgabe 1 (3)

- RTF, PDF, oder Word als Abgabeformat ist falsch.
- SQL ist eine Programmiersprache wie Java.

Natürlich ist SQL speziell für Datenbanken gemacht, und deklarativ im Gegensatz zu Java, was eher imperativ ist. Aber wenn es um die Codierung von „Programmen“ geht, sind beide im wesentlichen Folgen von ASCII-Zeichen (oder UTF-8, wenn Sie auch Umlaute verwenden wollen). Es gibt z.B. keinen Fettdruck in SQL-Anfragen oder die Möglichkeit, einen Teil in einer größeren Schrift anzuzeigen. Ein Editor, der die SQL-Syntax kennt, wird automatisch Schlüsselworte wie „SELECT“ in einer anderen Farbe anzeigen als z.B. die Tabellennamen. In der Datei gibt es die Angaben zu Farben aber nicht. Sie können die Farbe auch nicht einzeln für jedes Wort wählen.

- Verwenden Sie einen Editor, wie Sie ihn auch für Java verwenden würden (z.B. **Notepad++**, **Visual Studio Code**).

Übungsblatt 2, Aufgabe 1 (4)

- Sie brauchen Quellen wie W3Schools nur dann anzugeben, wenn die Anfragen nichttrivial waren und Ihnen die Quelle für die konkrete Hausaufgabe mehr genutzt hat als ein allgemeines Datenbank-Lehrbuch (oder mein Skript).

Dass Sie mein Skript verwenden, weiß ich ja (oder hoffe ich).

- Bei einer komplexen Anfrage gibt es eine ganze Reihe von Möglichkeiten, sie zu formulieren. Wenn dann zwei Studierende von der gleichen Anfrage ausgegangen sind, und nur Tabellen- und Spaltennamen ausgetauscht haben, könnte vielleicht der Eindruck eines Plagiats entstehen.
- Bei dieser Aufgabe gibt es aber im wesentlichen nur eine einzige Lösung.

Bis auf Groß-/Kleinschreibung und Zeilenaufteilung/Formatierung (man kann z.B. alles in eine Zeile schreiben). Leerzeichen links/rechts von „=“: optional.

Übungsblatt 2, Aufgabe 1 (5)

- Verwenden Sie nur Daten, die in der Aufgabenstellung genannt werden.

Man muss natürlich wissen, dass Vivaldi Nachname eines Komponisten ist.

- Es ist z.B. nicht ok, auch die Geburtsdaten in der Anfrage zu verwenden:

```
SELECT Vorname
From   komponist
WHERE  name = 'Vivaldi'
AND    geboren = '1678' AND gestorben = '1741'
```

- Außerdem ist falsch, dass die Geburtsdaten hier als Strings angegeben sind, nicht als Zahlen.

Die meisten Datenbanksysteme sind sehr großzügig mit Typ-Umwandlungen, so dass es keine Fehlermeldung gibt. Wir ziehen aber dennoch Punkte für einen Typfehler ab. Die Spalten geboren und gestorben sind numerisch.

Übungsblatt 2, Aufgabe 1 (6)

- Es ist insbesondere falsch, das erwartete Ergebnis direkt in die Anfrage zu schreiben:

```
SELECT vorname FROM komponist
WHERE vorname='Antonio'
```

- Auch folgende Anfrage würde ja das richtige Ergebnis liefern, wäre aber trotzdem falsch:

```
SELECT 'Antonio'
```

- Ebenso auch die Bedingung `WHERE KNr = 15`.
- Anfragen müssen in allen möglichen DB-Zuständen das richtige Ergebnis liefern.

Natürlich wird sich der Vorname von Vivaldi nicht ändern, aber falls in der DB ein falscher Vorname eingetragen wäre, müsste dieser geliefert werden. Wenn man den Vornamen schon weiß, braucht man keine DB-Anfrage.

Übungsblatt 2, Aufgabe 1 (7)

- Es war verlangt, Anfragen auszuprobieren.
- Die folgende Anfrage bekommt voraussichtlich 0 Punkte:

```
select vorname
from   komponist
where  nachname = 'vivaldi'
```

- Diese Anfrage liefert eine Fehlermeldung:

Error in query (7):

ERROR: column "nachname" does not exist

LINE 3: where nachname = 'vivaldi'

„(7)“ ist vermutlich ein Fehlercode. Eventuelle Hinweise wären willkommen.

- Korrigiert man den Fehler, bekommt man ein leeres Ergebnis wegen der Kleinschreibung von 'vivaldi'.

Dafür würde es nur einen Punkt Abzug geben.

Übungsblatt 2, Aufgabe 1 (8)

- Die folgende Anfrage bekommt auch 0 Punkte:

```
SELECT vorname
FROM   komponisten
WHERE  name = 'Vivaldi'
```

- Diese Anfrage liefert:

Error in query (7):

ERROR: relation "komponisten" does not exist

LINE 2: FROM komponisten

- Die Tabelle heißt „Komponist“ (Singular).
- Probieren Sie Ihre Anfragen im Adminer aus und verwenden Sie Copy&Paste (von dort in die .sql-Datei oder von einer .sql-Datei in den Adminer).

Übungsblatt 2, Aufgabe 1 (9)

Warum so streng?

- Es ist für die Tutoren viel leichter, die volle Punktzahl zu geben, statt die Anfrage zu debuggen und Fehler zu erklären.

Das Übungsblatt hatte 5 Aufgaben mit ca. 70 Abgaben pro Aufgabe.

Außerdem ist eine Präsenzübung mit ca. 50 Abgaben zu korrigieren, die wegen der Papierabgaben aufwändiger ist (manuelle Erfassung der Namen).

Wir haben drei Tutoren. Die Tutoren werden für 25 Stunden im Monat bezahlt, wovon wöchentliche Besprechungen von jeweils ca. einer Stunde abgehen. Es bleiben also ca. 5 Stunden pro Tutor pro Woche. Wenn man zur einfacheren Rechnung einen Tutor ganz für die Präsenzaufgabe reserviert, bleiben 10 Stunden, d.h. 600 Minuten, für 350 Lösungen. Insgesamt also ca. 2 Minuten pro Lösung.

- Oft verdecken Syntaxfehler nur weitere Fehler.
- In der Klausur müssen Sie mit Syntaxfehlern umgehen können.

Übungsblatt 2, Aufgabe 1 (10)

- Datenbank-Spalten sollten nicht in Anführungszeichen "... " geschrieben werden, wenn die **CREATE TABLE**-Anweisungen nicht auch diese Anführungszeichen enthalten:

```
SELECT "vorname"  
FROM   komponist  
WHERE  vorname = 'Antonio'
```

- Die Anführungszeichen erzwingen eine bestimmte Groß-/Kleinschreibung. Wozu?

Das funktioniert zufällig in PostgreSQL (weil dort nicht in Anführungszeichen eingeschlossene Bezeichner in Kleinbuchstaben umgewandelt werden), aber z.B. nicht in Oracle (dort würden Großbuchstaben funktionieren).

- Die inkonsistente Schreibweise der gleichen Spalte ist auch stilistisch schlecht.

Übungsblatt 2, Aufgabe 1 (11)

- Bitte schreiben Sie keine Namen in die Anfragen!
- Wir verwenden die Daten für ein Forschungsprojekt (natürlich ohne die Beziehung zu den realen Namen aus der Übungsplattform).
- Wenn Sie Ihren Namen als Kommentar in die Abgabe schreiben, kann es passieren, dass Ihr Klartext-Name in dem Datensatz landet, der am Ende auf einer Forschungs-Webseite veröffentlicht wird.

Forschungsergebnisse sollen ja reproduzierbar sein.

- Wir werden also offenbar nach den Namen aller Studierenden als Substrings in allen SQL-Anfragen suchen müssen (mit SQL LIKE).

Übungsblatt 2, Aufgabe 1 (12)

- Halten Sie sich bitte an die vorgegebene Codierung der Daten.
- Folgende Varianten werden voraussichtlich verstanden (die erste ist, was gewünscht war):
 - VORRECHNEN:5
 - VORRECHNEN: 5
 - Vorrechnen:5
 - Vorrechnen: 5
- Was soll „LZ:2“ bedeuten?
- Bitte „-- ChatGPT:2“ statt
Codierung: Quelle: ChatGPT4o (modifiziert)
- Schreiben Sie es als Kommentar! Ernstes Risiko: 0 Punkte.

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4**
- 4 Schlüssel
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3
- 7 Präsenzaufg.

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 (Schema **parabel**):

PARABEL		
<u>ID</u>	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, müsste man `P/2.0` schreiben.

Präsenzaufgabe: PQ-Formel in SQL (3)

Anmerkungen:

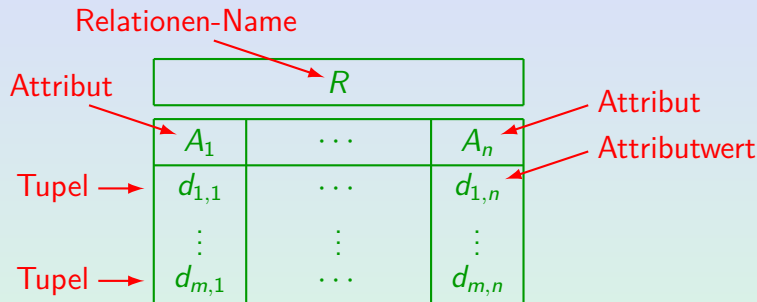
- Es gibt keine Funktion **SQUARE** zum Quadrieren in PostgreSQL (in Microsoft SQL Server schon).
- SQL hat natürlich das monadische Minus: $-(P/2)$. Einige haben $-1 * (P/2)$ geschrieben.
- Wenn man die **WHERE**-Klausel weglässt, bekommt man:
ERROR: cannot take square root of a negative number

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

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel**
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3
- 7 Präsenzaufg.

Grundbegriffe relationaler Datenbanken



Synonyme: Relation und Tabelle.

Tupel, Zeile und Record (**NICHT** „Reihe“!).

Attribut, Spalte, Feld.

Attributwert, Spaltenwert, Tabelleneintrag.

Beispiel zu Schlüsseln

- Was könnten hier Schlüssel sein?

R				
A	B	C	D	E
1	10	100	300	4
1	20	100	300	5
1	30	200	400	5

- Aber Vorsicht: Ein möglicher Zustand kann nur zeigen, was nicht Schlüssel sein darf.

Man kann sich inspirieren lassen, muss aber noch prüfen, ob der Schlüssel von der Anwendung her Sinn macht (darüber ist hier nichts bekannt) und tatsächlich für alle Zustände gelten soll.

- Beachten Sie, dass interessante Schlüssel minimal bezüglich „ $\subseteq$ “ sein müssen.

Beispiel zu Fremdschlüsseln

- Sie haben die Punkte-Datenbank angelegt, u.a.:

STUDENTEN			
<u>SID</u>	VORNAME	NACHNAME	...
101	Lisa	Weiss	...
102	Michael	Grau	...
103	Daniel	Sommer	...
104	Iris	Winter	...

- Die Tabelle mit den Bewertungen enthält Fremdschlüssel:

BEWERTUNGEN(SID → STUDENTEN, ...)

- Was würden Sie tun, um zu prüfen, ob Ihr DBMS diesen Fremdschlüssel tatsächlich überwacht?

In MySQL wurden lange Fremdschlüssel im CREATE TABLE akzeptiert (aus Kompatibilitätsgründen), aber nicht überwacht.

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel
- 5 Hausaufgabe 4**
- 6 Hausaufgabe 3
- 7 Präsenzaufg.

4. Übungsblatt: Aufgabe 1 (1)

- Loggen Sie sich über das Adminer-Webinterface bei der PostgreSQL-Datenbank für diese Übungen ein:

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

Diese Aufgabe bezieht sich auf das Schema „komponist_public“.

Wir haben auf dem zweiten Übungsblatt schon die Tabelle KOMPONIST verwendet.

- Das Schema der Tabelle „KOMPONIST“ in der Kurznotation der Vorlesung ist:

KOMPONIST(KNr, Name, Vorname°, geboren°, gestorben°)

- In der Variante nur mit ASCII-Zeichen ist es:

KOMPONIST(#KNr, Name, Vorname?, geboren?,
gestorben?)

4. Übungsblatt: Aufgabe 1 (2)

- Man kann darüber streiten, ob es angemessen ist, Nullwerte in der Spalte „Vorname“ zu erlauben.

In den Daten kommt es nicht vor. Es gibt aber nicht in allen Kulturen Vornamen nach deutschem Muster. Die Datenbank enthält teils auch relativ alte Musikstücke von kaum bekannten Komponisten — wahrscheinlich war befürchtet worden, dass die Trennung von Namen in Vor- und Nachname nicht immer möglich ist.

- Bei der Spalte „gestorben“ braucht man Nullwerte sicher für noch lebende Komponisten.
- Es ist außerdem auch plausibel, dass man eventuell nicht von jedem Komponisten die Lebensdaten ermitteln kann.

4. Übungsblatt: Aufgabe 1 (3)

- Ihre Aufgabe ist, das Schema der übrigen vier Tabellen dieser Datenbank herauszufinden (**Stueck**, **CD**, **Aufnahme**, **Solist**), und in dieser Kurznotation aufzuschreiben (entsprechend **Kap. 6** der Vorlesung).
- Wählen Sie dafür am besten die ASCII-Variante der Notation und geben Sie das Schema als **.txt**-Datei ab.
- Wenn Sie im Adminer auf den Tabellennamen links klicken, wird das Schema der jeweiligen Tabelle angezeigt.
- Die Datentypen der Spalten sind für diese Aufgabe nicht relevant.
- Schlüssel und Fremdschlüssel müssen Sie aber angeben.

4. Übungsblatt: Aufgabe 1 (4)

- Schlüssel finden Sie unter „Indizes“, was in zweierlei Hinsicht problematisch ist:
 - „Index“ ist ein Konzept des internen Schemas.
Es wird u.a. benutzt, um einen Schlüssel zu implementieren. Auf Ebene des relationalen Modells interessieren wir uns nur für Schlüssel.
 - In dieser Vorlesung und vermutlich der Mehrheit der Lehrbücher wird „Indexe“ als Plural der Datenstruktur benutzt. Dagegen herrscht Einigkeit darüber, dass in $f(x_i, y_j)$ die Zahlen i und j „Indizes“ sind.
- Die Groß-/Kleinschreibung von Tabellen- und Spaltennamen ist egal.

Der Adminer zeigt die Namen aus dem Systemkatalog von PostgreSQL an, und PostgreSQL konvertiert alle Namen in Kleinbuchstaben (außer bei "...").

4. Übungsblatt: Aufgabe 2 (1)

- Es soll eine Datenbank mit allen Modulen eines Studiengangs erstellt werden, die auch die Voraussetzungen enthält.
- Ihre Aufgabe ist es, **CREATE TABLE**-Anweisungen für die beiden Tabellen dieser Datenbank zu erstellen.

4. Übungsblatt: Aufgabe 2 (2)

- Die Tabelle „Module“ sieht so aus:

Module						
ID	Titel	LP	Art	Fach	Sem	D.
INF.00677.09	OOP	5	P		1	1
MAT.02372.02	Mathematik B	15	P		1	2
INF.06483.05	Einf. Datenbanken	5	P		3	1
INF.06484.03	DB-Programmierung	5	W		4	1
WIW.00388.05	Grundlagen BWL	5	A	BWL	3	1

- Die Werte in der Spalte **ID** sind Zeichenketten, die immer genau 12 Zeichen lang sind. Diese Spalte ist Primärschlüssel der Tabelle.
- Die Werte in der Spalte **Titel** sind Zeichenketten bis zur Länge 80. Die Werte in dieser Spalte sind ebenfalls eindeutig.

4. Übungsblatt: Aufgabe 2 (3)

- Spalten der Tabelle **Module**, Forts.:

- In die Spalte **LP** werden nicht-negative ganze Zahlen eingetragen.

Stellen Sie bitte mit einem CHECK-Constraint sicher, dass die Einfügung negativer Zahlen zu einem Fehler führt. Der Wert 0 soll möglich sein.

- In der Spalte „**Art**“ steht einer der folgenden Werte (jeweils ein Zeichen):

- „**P**“ für ein Pflichtmodul,
- „**W**“ für ein Wahlpflichtmodul der Informatik.

Auch Bio- und Wirtschaftsinformatik.

- „**A**“ für ein Modul eines Anwendungsfaches (das in der Spalte „**Fach**“ steht).

Schreiben Sie bitte einen CHECK-Constraint, der sicherstellt, dass man keine anderen Werte in die Spalte einfügen kann. Sie können dafür auch die logische Verknüpfung OR („oder“) verwenden.

4. Übungsblatt: Aufgabe 2 (4)

- Spalten der Tabelle **Module**, Forts.:
 - Die Spalte „**Fach**“ soll Zeichenketten bis zur Länge 60 erlauben, und kann aber auch einen Nullwert enthalten.
 - Sie bekommen einen Bonuspunkt, wenn Sie mit einem CHECK-Constraint sicherstellen, dass die Spalte genau dann einen Nullwert enthält, wenn in der Spalte „Art“ nicht der Wert „A“ steht.

D.h. die Spalte soll nur für Anwendungsfach-Module verwendet werden. Sie können mit den Bedingungen „**Fach IS NULL**“ und „**Fach IS NOT NULL**“ die Spalte auf den Nullwert testen.

4. Übungsblatt: Aufgabe 2 (5)

- Spalten der Tabelle **Module**, Forts.:
 - In der Spalte „**Sem**“ steht das empfohlene Semester, eine positive ganze Zahl.

Der DB-Entwurf ist etwas vereinfacht, da es auch mehrere Semester zur Auswahl geben kann — das ist in dieser Tabellenstruktur nicht möglich. Die Spalte kann auch einen Nullwert enthalten, wenn das Modul z.B. nur ganz unregelmäßig angeboten wird.

- Die Spalte „**Dauer**“ gibt die Dauer des Moduls in Semestern an. Erlaubte Werte sind nur **1** und **2**.

Schreiben Sie wieder einen entsprechenden CHECK-Constraint.

4. Übungsblatt: Aufgabe 2 (6)

- Die Tabelle „**Voraussetzungen**“ sieht so aus:

Voraussetzungen		
zuerst	danach	Studienleistung
INF.00677.09	INF.06483.05	X
INF.00677.09	INF.06484.03	
INF.06483.05	INF.06484.03	

- Diese Beispiel-Zeilen bedeuten:
 - Die Studienleistung von „Objektorientierte Programmierung“ ist Voraussetzung für „Einführung in Datenbanken“.
 - Das Modul „Objektorientierte Programmierung“ (also mit Prüfung) ist Voraussetzung für „Datenbank-Programmierung“.
 - Das Modul „Einführung in Datenbanken“ (wieder mit Prüfung) ist Voraussetzung für „Datenbank-Programmierung“.

4. Übungsblatt: Aufgabe 2 (7)

- Schreiben Sie eine **CREATE TABLE**-Anweisung für diese Tabelle:
 - Die Spalten „**zuerst**“ und „**danach**“ sind jeweils ein Fremdschlüssel, der die Tabelle „**Module**“ referenziert.
 - Die Kombination aus „**zuerst**“ und „**danach**“ ist Primärschlüssel dieser Tabelle.
 - In der Spalte „**Studienleistung**“ sind nur die folgenden beiden Werte erlaubt:
 - „**X**“, wenn nur die Studienleistung vorausgesetzt wird, und
 - „ “ (ein Leerzeichen) für den Normalfall, dass das ganze Modul abgeschlossen sein muss.
- Alle drei Spalten erlauben keine Nullwerte.

4. Übungsblatt: Aufgabe 2 (8)

- Geben Sie eine Textdatei ab mit Endung `.sql`, die die beiden `CREATE TABLE`-Anweisungen enthält.
- Sie haben per EMail Benutzernamen und Passwort für eine Datenbank im Adminer bekommen, in der Sie auch eigene Tabellen anlegen können.

Der Benutzername ist gleichzeitig auch Name der Datenbank (Sie müssen ihn also in die Eingabefelder „Username“ und „Database“ beim Adminer eingeben). Sie müssen zunächst ein eigenes Schema anlegen, da Sie für das Standard-Schema `public` nicht das Recht haben, darin Tabellen anzulegen. Beim Adminer bekommen Sie einen Link „Create Schema“ angezeigt, wenn links kein Schema ausgewählt ist. Sie können aber auch den SQL-Befehl `CREATE SCHEMA xyz` eingeben. Wählen Sie dann dieses Schema in der Auswahlbox links.

4. Übungsblatt: Aufgabe 2 (9)

- Anschließend können Sie Ihre **CREATE TABLE**-Statements testen.
- Wenn Sie wollen, können Sie Ihre Tabellen anschließend mit den Beispieldaten füllen, indem Sie die folgende Datei mit „Import“ im Adminer ausführen (oder den Inhalt mit Copy&Paste ins SQL-Fenster einfügen):

[https://users.informatik.uni-halle.de/~brass/db24/homework/h4_insert.sql]

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3**
- 7 Präsenzaufg.

3. Übungsblatt: Aufgabe 1 (1)

- Lesen Sie das **Kapitel 5**: „SQL: Datentypen“ der Folien zu Ende. Notieren Sie sich ggf. Fragen und stellen Sie diese in der nächsten Übung (oder der Vorlesung).
- Schauen Sie sich die folgenden fünf Wertausdrücke an und überlegen Sie sich, was das Ergebnis sein sollte.

a) `SELECT ROUND(1e9*CAST(0.10 AS REAL))`

b) `SELECT SIN(0.5 * PI())`

c) `SELECT LOG(100)`

d) `SELECT UPPER(SUBSTRING('abcdef' FROM 2 FOR 3))`
`|| '.'`

e) `SELECT POSITION('de' IN TRIM(' abcdef'))`

3. Übungsblatt: Aufgabe 1 (2)

- Anschließend testen Sie das Ergebnis von PostgreSQL im **Adminer**.
- Geben Sie eine **.txt**-Datei ab mit den fünf Werten und jeweils einer kurzen Erklärung.

1–2 Zeilen pro Teilaufgabe sollten reichen.

3. Übungsblatt: Aufgabe 1 (3)

Lösung:

a) `SELECT ROUND(1e9*CAST(0.10 AS REAL)):`
 100000001

REAL ist eine 32-Bit Gleitkommazahl. Für die Mantisse stehen 24 Bit Genauigkeit zur Verfügung. Da das erste Bit immer 1 ist (der Exponent wird entsprechend gewählt), müssen nur 23 Bit gespeichert werden, und es bleiben 8 Bit für den Exponenten und 1 Bit für das Vorzeichen. Die Darstellung ist binär mit Exponent zur Basis 2. Der Wert 0.10 ist ein periodischer Bruch im Binärsystem, kann also nicht exakt dargestellt werden. Der tatsächlich gespeicherte Wert ist 0.10000000149...

[IEEE-754 Floating Point Converter]. Der Fehler ist also ungefähr $1.49 \cdot 10^{-9}$.

Bei 0.3 wäre der Fehler ungefähr $11.9 \cdot 10^{-9}$ (man würde ihn also schon bei Multiplikation mit 10^8 bemerken). Bei 0.7 und Multiplikation mit 10^8 ist das Ergebnis 69999999, der Fehler springt also noch stärker ins Auge. Die Genauigkeit von 32-Bit Gleitkommazahlen wird üblicherweise mit 7 Dezimalstellen angegeben.

3. Übungsblatt: Aufgabe 1 (4)

Lösung:

b) `SELECT SIN(0.5 * PI()):`

1

PostgreSQL verwendet also das Bogenmaß (**rad**, „Radiant“) als Einheit für die Winkel. $0.5 * \pi$ entspricht 90° .

c) `SELECT LOG(100):`

2

Es ist also der Logarithmus zur Basis 10.

d) `SELECT UPPER(SUBSTRING('abcdef' FROM 2 FOR 3))`
`|| '':`

BCD.

Positionen in Strings werden in SQL von 1 an gerechnet, nicht von 0 an wie bei Arrays in Java. `||` ist die String-Konkatenation.

3. Übungsblatt: Aufgabe 1 (5)

Lösung:

e) `SELECT POSITION('de' IN TRIM(' abcdef')):`

4

TRIM entfernt Leerzeichen am Anfang und Ende der Zeichenkette (nicht im Innern). Ohne das Leerzeichen am Anfang beginnt die Zeichenkette 'de' beim vierten Zeichen von 'abcdef'.

EMP-DEPT-Datenbank (1)

- Klassische Beispiel-Datenbank von Oracle.
- Schema „`empdept_public`“ im Adminer:
 - `dept(deptno, dname, loc)`
 - `emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)`
- `dept`: „Department“ (Abteilungen einer Firma)
- `emp`: „Employee“ (Angestellte der Firma)
- Der Link zum Adminer ist:

`https://dbs.informatik.uni-halle.de/edb?`

`pgsql=db&username=student_gast&
db=postgres&ns=empdept_public`

EMP-DEPT-Datenbank (2)

- Die Tabelle dept hat die Spalten
 - deptno**: Abteilungsnummer (Department Number),
 - dname**: Name der Abteilung (Department Name),
 - loc**: Ort/Sitz der Abteilung (Location):

DEPT		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMP-DEPT-Datenbank (3)

- Die Tabelle **emp** hat folgende Spalten:
 - empno**: Angestellten-Nummer (identifiziert den Angestellten)
 - ename**: Name des Angestellten.
 - job**: Berufsbezeichnung des Angestellten.
 - mgr**: Angestellten-Nummer des direkten Vorgesetzten.
mgr von „manager“.
 - hiredate**: Datum der Einstellung.
 - sal**: Gehalt des Angestellten („salary“).
 - comm**: Provision (nur für Verkäufer) („commission“).
 - deptno**: Abteilung des Angestellten.

EMP-DEPT-Datenbank (4)

EMP					
EMPNO	ENAME	JOB	MGR	SAL	DEPTNO
7369	SMITH	CLERK	7902	800	20
7499	ALLEN	SALESMAN	7698	1600	30
7521	WARD	SALESMAN	7698	1250	30
7566	JONES	MANAGER	7839	2975	20
7654	MARTIN	SALESMAN	7698	1250	30
7698	BLAKE	MANAGER	7839	2850	30
7782	CLARK	MANAGER	7839	2450	10
7788	SCOTT	ANALYST	7566	3000	20
7839	KING	PRESIDENT		5000	10
7844	TURNER	SALESMAN	7698	1500	30
7876	ADAMS	CLERK	7788	1100	20
7900	JAMES	CLERK	7698	950	30
7902	FORD	ANALYST	7566	3000	20
7934	MILLER	CLERK	7782	1300	10

3. Übungsblatt: Aufgabe 2 (1)

- Angenommen, Sie wissen, dass das durchschnittliche Gehalt in der Firma 2073.21 € ist.

Später lernen wir, wie man das mit SQL berechnen kann.

- Geben Sie für jeden Angestellten
 - seinen Angestellten-Nummer (**empno**),
 - seinen Namen (**ename**),
 - sein Gehalt (**sal**) und
 - die prozentuale Abweichung seines Gehalts bezogen auf den Durchschnittswert **2073.21** aus. Negative Werte stehen für ein unterdurchschnittliches Gehalt, positive Werte für ein überdurchschnittliches Gehalt. Runden Sie den Prozentsatz bitte auf eine ganze Zahl.
- Erwartetes Ergebnis: Siehe nächste Folie.

3. Übungsblatt: Aufgabe 2 (2)

empno	ename	sal	Abweichung in %
7369	SMITH	800	-61
7499	ALLEN	1600	-23
7521	WARD	1250	-40
7566	JONES	2975	43
7654	MARTIN	1250	-40
7698	BLAKE	2850	37
7782	CLARK	2450	18
7788	SCOTT	3000	45
7839	KING	5000	141
7844	TURNER	1500	-28
7876	ADAMS	1100	-47
7900	JAMES	950	-54
7902	FORD	3000	45
7934	MILLER	1300	-37

3. Übungsblatt: Aufgabe 2 (3)

Hinweise:

- Da wir das noch nicht besprochen haben, sind keine bestimmten Spaltennamen verlangt.
- Ebenso keine spezielle Sortierung der Zeilen.
- Bei der Klausur sind dagegen meistens genau die Spaltennamen verlangt, die auch im erwarteten Ergebnis stehen (einschließlich Groß-/Kleinschreibung).
- Oft ist auch eine bestimmte Sortierung der Zeilen verlangt.

Es gibt dann einen Punktabzug, wenn die ORDER BY-Klausel fehlt, selbst wenn das DBMS zufällig die Zeilen in der gewünschten Reihenfolge ausgeben sollte.

3. Übungsblatt: Aufgabe 2 (4)

- Mögliche Lösung:

```
SELECT empno, ename, sal,  
       ROUND((sal-2073.21)*100.0/2073.21)  
       AS "Abweichung in %"  
FROM   emp
```

Die Umbenennung der Ergebnisspalte ist optional.

- Alternative:

```
SELECT empno, ename, sal,  
       ROUND(sal*100.0/2073.21 - 100)  
       AS "Abweichung in %"  
FROM   emp
```

Die 100% könnten natürlich auch außerhalb der Rundung abgezogen werden.

3. Übungsblatt: Aufgabe 3 (1)

- Das Management hat festgestellt, dass für den Monat der Einstellung (**hiredate**) immer ein volles Gehalt gezahlt worden ist.
- Es möchte eine Aufstellung mit dem korrigierten Einstellungsdatum (der erste Tag des jeweiligen Monats) und den fehlenden Arbeitstagen in diesem Monat.

Wenn jemand z.B. am 5. des Monats eingestellt wurde, fehlen 4 Arbeitstage.

- Drucken Sie Angestelltennummer und Angestelltenamen, das tatsächliche Einstellungsdatum, das korrigierte Einstellungsdatum und die fehlenden Tage.
- Die Daten sollen im Format **DD.MM.YYYY** gedruckt werden (wie es in Deutschland üblich ist).

3. Übungsblatt: Aufgabe 3 (2)

empno	ename	hiredate	korrigiert	Minus-Tage
7369	SMITH	17.12.1980	01.12.1980	16
7499	ALLEN	20.02.1981	01.02.1981	19
7521	WARD	22.02.1981	01.02.1981	21
7566	JONES	02.04.1981	01.04.1981	1
7654	MARTIN	28.09.1981	01.09.1981	27
7698	BLAKE	01.05.1981	01.05.1981	0
7782	CLARK	09.06.1981	01.06.1981	8
7788	SCOTT	09.12.1982	01.12.1982	8
7839	KING	17.11.1981	01.11.1981	16
7844	TURNER	08.09.1981	01.09.1981	7
7876	ADAMS	12.01.1983	01.01.1983	11
7900	JAMES	03.12.1981	01.12.1981	2
7902	FORD	03.12.1981	01.12.1981	2
7934	MILLER	23.01.1982	01.01.1982	22

3. Übungsblatt: Aufgabe 3 (4)

- Mögliche Lösung:

```
SELECT empno, ename,  
       TO_CHAR(hiredate, 'DD.MM.YYYY')  
         AS "hiredate",  
       TO_CHAR(hiredate -  
               CAST(EXTRACT(DAY FROM hiredate)-1  
                   AS INTEGER),  
               'DD.MM.YYYY') AS "korrigiert",  
       EXTRACT(DAY FROM hiredate)-1  
         AS "fehlende Tage"  
FROM   emp
```

3. Übungsblatt: Aufgabe 3 (5)

- Alternativen zur Berechnung des ersten Tages im Monat:

- `TO_CHAR(hiredate, '01.MM.YYYY')`

Zeichen im Formatstring, die keine spezielle Bedeutung haben, werden einfach unverändert gedruckt. Der Ergebnis-Datentyp ist dann natürlich eine Zeichenkette, kein Datum. Man könnte es mit der Funktion `TO_DATE(..., 'DD.MM.YYYY')` wieder umwandeln. Solange man den Wert nur ausdrucken will, ist das nicht nötig.

- `DATE_TRUNC('month', hiredate)`

Diese Funktion hat als Eingabetyp eigentlich `TIMESTAMP` und verwandete Typen (also Datum und Uhrzeit zusammen). Entsprechend liefert sie auch einen Zeitstempel. Man kann diesen dann mit `TO_CHAR` so ausdrucken, dass nur der Datums-Anteil angezeigt wird. [\[PostgreSQL Handbuch\]](#)

Inhalt

- 1 Organisatorisches
- 2 Nochmal HA 2.1
- 3 Präsenzaufg. 4
- 4 Schlüssel
- 5 Hausaufgabe 4
- 6 Hausaufgabe 3
- 7 Präsenzaufg.**

Präsenzaufgabe: Schema für Koch-/Back-Rezepte

- Entwerfen Sie ein Schema für Rezepte (zum Kochen/Backen):
 - Zu jedem Rezept muss eine Nummer, eine Bezeichnung, ein Schwierigkeitsgrad, und eine Dauer (der Zubereitung) gespeichert werden. Bezeichnungen sind nicht eindeutig.
Die Nummer soll das Rezept eindeutig identifizieren.
 - Ein Rezept besteht aus mehreren Schritten. Es ist die Reihenfolge der Schritte festzuhalten, außerdem ein Text (Anweisung für den Schritt).
 - Für jeden Schritt ist zu speichern, welche Zutaten in welcher Menge benötigt werden. In einen Schritt können auch mehrere Zutaten benötigt werden (oder keine).

Die gleiche Zutat in verschiedenen Schritten eines Rezepts ist möglich.

- Geben Sie das Schema in Kurznotation als **.txt**-Datei ab.