

Einführung in Datenbanken

— Übungsblatt 6 (Selbstverbund, Logik) —

Ihre Lösungen laden Sie bitte in die Übungsplattform in StudIP hoch ([StudIP-Eintrag der Vorlesung], Reiter „Übungsplattform“).

Einsendeschluss ist Montag, der 25.11.2024, 18⁰⁰.

Hausaufgaben können einzeln oder in Zweier-Gruppen bearbeitet werden. Sie können die Gruppe für jede Aufgabe neu wählen.

Denken Sie daran, dass Sie bei jeder Aufgabe angeben müssen, ob Sie bereit sind, vorzurechnen:

- „VORRECHNEN:0“: Ich werde nicht zur Übung kommen.
- „VORRECHNEN:1“: Ich möchte diese Aufgabe nicht vorrechnen.
- „VORRECHNEN:2“: Ich möchte diese Aufgabe nur ungern vorrechnen.
- „VORRECHNEN:3“: Ich kann vorrechnen, lasse aber gern anderen den Vortritt.
- „VORRECHNEN:4“: Ich kann problemlos vorrechnen.
- „VORRECHNEN:5“: Ich möchte gerne, dass meine Abgabe besprochen wird.

Falls Sie als Gruppe abgeben, muss jedes Gruppenmitglied einzeln die Bereitschaft zum Vorrechnen erklären (VORRECHNEN1:N ist der Wert für den Studierenden, der die Aufgabe in die Übungsplattform hochgeladen hat, und VORRECHNEN2:M der Wert für den anderen Studierenden).

Vergessen Sie nicht, eventuell verwendete Quellen wie ChatGPT (oder auch die Lösung eines anderen Studierenden) anzugeben (mit der in Kapitel 0, Folien 22 bis 27 beschriebenen Codierung). Ohne Quellenangabe werden „zu ähnliche Lösungen“ als Plagiat behandelt. Mit Quellenangabe werden sie normal korrigiert. In der Klausur müssen Sie aber ähnliche Aufgaben ohne Hilfe lösen! Sie sind erwachsene Menschen und können sich ja denken, dass Sie nur durch eigene Beschäftigung mit den Aufgaben lernen.

Wir würden uns freuen, wenn Sie die für die jeweilige Aufgabe verwendete Zeit in Minuten in der Form „ZEIT:N“ angeben würden (bei Gruppenarbeit „ZEIT1:N“ und „ZEIT2:M“). Diese Angabe ist freiwillig.

Hinweis: SQL-Anfragen mit Syntaxfehlern werden automatisch mit 0 Punkten bewertet! Testen Sie also Ihre Anfragen (z.B. im Adminer).

Datenbank

Die Aufgaben dieses Übungsblattes beziehen sich auf die schon bekannte Datenbank mit klassischen Musikstücken im Schema „komponist_public“:

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

Die Datenbank enthält folgende Tabellen:

- **KOMPONIST**(KNR, NAME, VORNAME°, GEBOREN°, GESTORBEN°)
- **STUECK**(SNR, KNR°→KOMPONIST, TITEL, TONART°, OPUS°)
- **CD**(CDNR, NAME, HERSTELLER°, ANZ_CDS°, GESAMTSPIELZEIT°)
- **AUFNAHME**(CDNR→CD, SNR→STUECK, ORCHESTER°, LEITUNG°)
- **SOLIST**((CDNR, SNR)→AUFNAHME, NAME, INSTRUMENT°)

Aufgabe 1 (5 Punkte)

Gesucht sind alle Komponisten in der Datenbank, die Zeitgenossen von Wolfgang Amadeus Mozart sind, deren Lebenszeit also mit der von Mozart überlappt. Es sollen dabei auch Komponisten ausgegeben werden, die in dem Jahr gestorben sind, in dem Mozart geboren wurde, bzw. die in dem Jahr geboren wurden, in dem Mozart gestorben ist (wenn die Überlappung also etwas grenzwertig ist).

„Wolfgang Amadeus Mozart“ selbst soll in der Auflistung nicht erscheinen. (Sie können voraussetzen, dass Name und Vorname des Komponisten zusammen einen Alternativschlüssel darstellen, dass es also nicht zwei verschiedene Einträge mit gleichem Namen und Vornamen gibt.)

Geben Sie Name, Vorname, Geburts- und Todesjahr der Komponisten aus, sowie das „Alter“ des Komponisten im Geburtsjahr von Mozart (das kann natürlich negativ sein, wenn der andere Komponist erst nach Mozart geboren wurde). Die Spalte **Alter** nennen Sie bitte in der Ausgabe genau so (auch mit dieser Groß-/Kleinschreibung).

Sie dürfen nicht die Lebensdaten von Mozart in die Anfrage einsetzen. Zwar werden die sich nicht mehr ändern, aber der Name „Wolfgang Amadeus Mozart“ soll nur ein Beispiel sein. Ihre Anfrage könnte z.B. später in einem Programm verwendet werden, mit Parametern für den tatsächlichen Namen.

Verwenden Sie bitte nur Konstrukte, die in der Vorlesung schon behandelt wurden. Insbesondere sind Joins unter **FROM** ausgeschlossen. In der Klausur gibt es natürlich keine solchen Beschränkungen (sehr ungewöhnliche Konstrukte könnten aber zu Punktabzügen wegen mangelnder Portabilität führen).

Das erwartete Ergebnis ist:

name	vorname	geboren	gestorben	Alter
Telemann	Georg Philipp	1681	1767	75
Händel	Georg Friedrich	1685	1759	71
Scarlatti	Domenico	1685	1757	71
Locatelli	Pietro	1695	1764	61
Leclair	Lean-Marie	1697	1764	59
Mozart	Leopold	1719	1787	37
Hayden	Joseph	1732	1809	24
Beethoven	Ludwig van	1770	1827	-14

Wenn in der Aufgabenstellung keine spezielle Sortierung verlangt ist (wie hier), dürfen Sie eine beliebige (sinnvolle) Sortierung vornehmen (oder eben nicht sortieren). Die obige Tabelle ist mit

```
ORDER BY GEBOREN, NAME
```

(am Ende der Anfrage) erzeugt.

Aufgabe 2 (5 Punkte)

Gesucht sind alle CDs, die Stücke von mehr als einem Komponisten enthalten (also mindestens zwei verschiedenen Komponisten). Geben Sie jeweils Nummer und Name der CD aus.

Bei dieser Anfrage werden Sie wahrscheinlich Duplikate erhalten. Verwenden Sie ggf. `SELECT DISTINCT`, um diese zu eliminieren.

Für die Lösung dieser Aufgabe verwenden Sie bitte kein `GROUP BY` und keine Aggregationsfunktionen wie `COUNT`, selbst wenn Sie das schon kennen sollten (in der Vorlesung war es noch nicht dran).

Das erwartete Ergebnis ist:

cdnr	name
106	Leopold Mozart: Sinfonia D-Dur ...
111	Mozart/Beethoven: Klassische Ouvertüren
125	Tschaikowsky/Mendelssohn: Violinkonzerte
139	Oboenkonzerte
140	Corelli,Albinoni,Scarlatti,Manfredini,...
142	Schlager um 1500

Aufgabe 3 (1 Punkt)

Im folgenden sollen Sie jeweils zwei Anfragen vergleichen, u.a. hinsichtlich Äquivalenz. Beachten Sie, dass Sie alle möglichen Datenbank-Zustände in Betracht ziehen müssen, nicht nur den aktuellen Zustand. Es ist daher nur begrenzt hilfreich, die Anfragen auszuprobieren.

Sie können natürlich voraussetzen, dass Schlüssel, Fremdschlüssel und die NOT NULL-Bedingungen aus dem Schema erfüllt sind (und auch die Datentypen so sind, wie im Adminer angegeben).

In der Klausur könnte es ähnliche Aufgaben auch zu einem Schema geben, das nicht im Adminer zur Verfügung steht. Es sind also „theoretische“ Aufgaben, die durch Nachdenken gelöst werden sollten.

Anfrage 1:

```
SELECT KNR, SNR
FROM   STUECK
WHERE  KNR = 70
```

Anfrage 2:

```
SELECT K.KNR, S.SNR
FROM   STUECK S, KOMPONIST K
WHERE  S.KNR = 70 AND K.KNR = 70
```

Welche der folgenden Aussagen ist korrekt? Wenn mehrere korrekt sein sollten, wählen Sie die erste korrekte Aussage.

- A. Die beiden Anfragen liefern immer die gleiche Antwort (äquivalent)
- B. Die beiden Anfragen liefern bis auf Duplikate die gleiche Antwort
- C. Das Ergebnis von Anfrage 1 ist immer leer (inkonsistent)
- D. Das Ergebnis von Anfrage 2 ist immer leer (inkonsistent)
- E. Anfrage 1 liefert immer eine Obermenge (\supseteq) von Anfrage 2
- F. Anfrage 1 liefert immer eine Teilmenge (\subseteq) von Anfrage 2
- G. Keine der Aussagen trifft zu

Bei „B.“ ist gemeint, dass die Tupelmengen (nach Duplikat-Eliminierung) identisch sind. Auch „E.“ und „F.“ beziehen sich auf die Tupel-Mengen nach Eliminierung eventueller Duplikate (d.h. gemeint ist die übliche Mengen-Inklusion und nicht die Multimengen-Inklusion — blenden Sie ggf. Duplikate in Ihren Überlegungen aus).

Geben Sie den Buchstaben der **ersten korrekten Aussage** ab sowie eine kurze Begründung (maximal 3 Zeilen, die entscheidenden Stichworte reichen). Bitte laden Sie eine `.txt`-Datei mit Ihrer Antwort hoch (kein PDF, kein Word).

Aufgabe 4 (1 Punkt)

Vergleichen Sie nach dem Schema von Aufgabe 3 die folgenden beiden Anfragen:

Anfrage 1:

```
SELECT *
FROM   KOMPONIST
WHERE  KNR = 70 OR KNR = 80 AND GEBOREN < 1500
```

Anfrage 2:

```
SELECT *
FROM   KOMPONIST
WHERE  KNR IN (70, 80) AND NOT GEBOREN >= 1500
```

Sie können voraussetzen, dass GEBOREN keine Nullwerte enthält.

Aufgabe 5 (1 Punkt)

Vergleichen Sie nach dem Schema von Aufgabe 3 die folgenden beiden Anfragen:

Anfrage 1:

```
SELECT *
FROM   STUECK
WHERE  TONART = 'C-dur'
AND    (TONART = 'a-moll' OR TITEL LIKE 'Symphon%')
```

Anfrage 2:

```
SELECT *
FROM   STUECK
WHERE  TONART = 'C-dur'
AND    TITEL LIKE 'Symphon%'
```

Sie können hier ignorieren, dass TONART Nullwerte enthalten kann und SQL eigentlich eine dreiwertige Logik verwendet. Die Fragen beziehen sich auf das Anfrageergebnis, und SELECT behandelt den dritten Wahrheitswert wie `false`. Nur mit NOT wäre ein Unterschied feststellbar, aber das kommt hier nicht vor.