Einführung in Datenbanken

Übung 7: Logik, Duplikate

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

Wintersemester 2023/24

http://www.informatik.uni-halle.de/~brass/db23/

Inhalt

Anwesenheit in der Vorlesung (1)

- Es gibt keine Anwesenheitspflicht in der Vorlesung.

  In der Übung gibt es eine relativ schwache Anwesenheitspflicht, weil Sie bei extrem seltener Teilnahme die geforderte „Aktive Mitarbeit“ sicher nicht geleistet haben und Sie bereit sein müssen, 20% der Aufgaben vorzurechnen, was ohne Anwesenheit nicht geht.

- Es gibt aber auch keine Garantie, dass die Videoaufnahme der Vorlesung funktioniert.

  Bei der Vorlesung am 27.11.2023 fehlt ein Stück am Anfang, und es wurde viel Zeit verbraucht, das Problem zu lösen. Wenn es wieder ein Problem gibt, wäre möglich, dass ich schneller aufgebe.

- Die Aufzeichnung ist für Notfälle gedacht, nicht als Normalfall.

Anwesenheit in der Vorlesung (2)

- Ich brauche es nicht für mein Ego, dass viele Studierende zur Vorlesung kommen.
- Es ist mir aber wichtig, dass der Lehrerfolg am Ende stimmt — es also viele (sehr) gute Noten gibt.

  Und ohne Tricks. Die Art der (meisten) Aufgaben ist lange bekannt,
  die werde ich nicht ändern, und auch nicht die (relativ strenge) Benotung.
  Sie müssen schon recht komplexe SQL-Anfragen schreiben können.
  Das lernt man nur durch ausreichend intensive Beschäftigung mit SQL.
- Ich mache mir Sorgen, dass die Verfügbarkeit der Aufzeichnung vielleicht dazu führt, dass das Nacharbeiten des Vorlesungsstoffes länger verschoben wird.

  Oder die Beschäftigung zu oberflächlich ist (z.B. mit doppelter Geschwindigkeit)
  oder es mehr Ablenkungen als im Hörsaal gibt.

SQL-Aufgaben in letzter Klausur

- Die Klausur vom 28.02.2023 enthielt (wie alle Klausuren seit vielen Jahren) 5 SQL-Anfragen, jeweils mit 5 Punkten.

Aufgabe	Abgaben	5 Punkte		4.5 Punkte	
a)	84	14	(17%)	7	(8%)
b)	82	12	(15%)	21	(26%)
c)	82	9	(11%)	12	(15%)
d)	80	6	(8%)	11	(14%)
e)	69	8	(12%)	5	(7%)

Bei b) hatte ein Student 5.5 Punkte, bei c) zwei (sind bei 5 Punkte mitgezählt).

Die Prozentwerte beziehen sich auf die Abgaben der jeweiligen Aufgabe.

a) `CASE`, Nullwerte, `ORDER BY`, Spaltenumbenennung.

b) `GROUP BY`, `HAVING`, Aggregationsfunktionen.

c) `NOT EXISTS`, `ORDER BY`, Spaltenumbenennung `"..."`.

d) `NOT EXISTS`, Selbstverbund, `IS NULL`, Spaltenumbenennung.

e) Outer Join, `LIKE`, `UPPER`, `IS NULL`, `ORDER BY`.

Inhalt

Präsenzaufgabe: Variablen-Belegungen (1)

- Wählen Sie das Schema „studentenaufgaben_public“ im Adminer:
  [https://dbs.informatik.uni-halle.de/edb?pgsql=db&username=student_gast&db=postgres&ns=]
- Aufgabe: Suchen Sie eine FROM-Klausel aus (gern auch mit mehreren Tupelvariablen), und eine WHERE-Bedingung, so dass

```
SELECT COUNT(*)
FROM   ____________________________
WHERE  ____________________________
```

genau den Wert 150 liefert.

Es wird so die Anzahl der Variablenbelegungen gezählt, die die WHERE-Bedingung erfüllen. Die Tabellen-Größen sind: AUFGABEN: 3, STUDENTEN: 4, BEWERTUNGEN: 8. Es gibt 5 Bewertungen mit ATYP = 'H' und 2 Bewertungen mit SID = 103.

Tipp: Mehrere Tupelvariablen über einer Tabelle sind möglich.

Präsenzaufgabe: Variablen-Belegungen (2)

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	...
102	Michael	Grau	NULL
103	Daniel	Sommer	...
104	Iris	Winter	...

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
101	Z	1	12
102	H	1	9
102	H	2	9
102	Z	1	10
103	H	1	5
103	Z	1	7

AUFGABEN

ATYP	ANR	THEMA	MAXPT
H	1	ER	10
H	2	SQL	10
Z	1	SQL	14

Präsenzaufgabe: Variablen-Belegungen (3)

Lösung:

- Ziel dieser Aufgabe ist es, zu verstehen, dass
  - SQL alle möglichen Variablenbelegungen betrachtet, und
  - die Anzahl möglicher Kombinationen von Belegungen für mehrere Tupelvariablen das Produkt der Anzahl möglicher Belegungen für die einzelnen Variablen ist.

    Sofern man keine Bedingungen hat, die nur bestimmte Kombinationen zulassen. Wenn man Join-Bedingungen hat (die sich auf mehrere Variablen beziehen) werden die möglichen Variablenbelegungen gefiltert, und nur das Ergebnis dieser Filterung wird unter `SELECT` gezählt.
- Eine Primfaktor-Zerlegung von 150 ist:

$$150 = 5 * 5 * 3 * 2$$

Präsenzaufgabe: Variablen-Belegungen (4)

Lösung, Forts.:

- Wegen $150 = 5 * 5 * 3 * 2$ erreicht man das Ziel also, wenn man vier Tupelvariablen `A`, `B`, `C`, `D` deklariert, wobei es
  - für `A` 5 mögliche Belegungen gibt,
  - für `B` auch,
  - für `C` 3 und
  - für `D` 2.

```
SELECT COUNT(*)
FROM   BEWERTUNGEN A, BEWERTUNGEN B,
       AUFGABEN C, BEWERTUNGEN D
WHERE  A.ATYP = 'H' AND B.ATYP = 'H'
AND    D.SID = 103
```

Präsenzaufgabe: Variablen-Belegungen (5)

Lösung, Forts.:

- Alternativen für `A` und `B` mit 5 möglichen Belegungen:
  - `BEWERTUNGEN A` mit `A.ATYP = 'H'` (Aufgabenstellung)
  - `BEWERTUNGEN A` mit `A.SID >= 102`
- Alternativen für `C` mit 3 möglichen Belegungen:
  - `AUFGABEN C` (Aufgabenstellung)
  - `BEWERTUNGEN C` mit `C.ATYP = 'Z'`
- Alternativen für `D` mit 2 möglichen Belegungen:
  - `BEWERTUNGEN D` mit `D.SID = 103` (Aufgabenstellung)
  - `AUFGABEN D` mit `D.ANR = 1`
  - `AUFGABEN D` mit `D.ATYP = 'H'`
  - `STUDENTEN D` mit `D.SID <= 102`

Präsenzaufgabe: Variablen-Belegungen (6)

Lösung, Forts.:

- Die gleiche Kombination exotischer Alternativen lässt vermuten, dass die Lösungen nicht unabhängig von einander zustande gekommen sind.

  Das ist nicht erwünscht und könnte geahndet werden (Täuschungsversuch).

- Statt zwei Tupelvariablen mit 3 und 2 möglichen Belegungen kann man auch eine mit 6 möglichen Belegungen verwenden:
  - `BEWERTUNGEN CD` mit `CD.ANR = 1`
- Ganz anders (natürlich nicht gewünscht, nicht portabel):

```
SELECT COUNT(*)
FROM   GENERATE_SERIES(1, 150)
```

Tabellenfunktion: Erzeugt 150 Zeilen mit Zahlwerten 1, 2, 3, . . . , 150.

Inhalt

Duplikate in SQL (1)

- Regeln in der Klausur:
  - Ihre Anfragen dürfen keine Duplikate erzeugen.
  - Im Beispielzustand lässt sich das ja leicht testen, aber die Forderung gilt für alle Zustände, die die Integritätsbedingungen (insbesondere Schlüssel) erfüllen.

    Wenn schon einfaches Ausprobieren Duplikate liefert, könnte man natürlich „`DISTINCT`“ verwenden. Es wäre aber dennoch besser, zuerst die Ursache der Duplikate zu verstehen. Die Ursache könnte z.B. auch eine vergessene Verbund-Bedingung sein.
  - Für ein unnötiges `DISTINCT` wird auch ein Punkt abgezogen.

    Das System muss extra die relativ teure Duplikat-Elimination anschmeißen. Diese braucht z.B. temporären Speicherplatz (die Duplikat-Elimination wird häufig durch Sortierung implementiert).

Duplikate in SQL (2)

- In der Praxis sind gelegentlich einzelne Duplikate interessant:
  - Wenn es z.B. zwei Studierende mit gleichem Vor- und Nachnamen gibt, die die Bedingung erfüllen,
  - möchte man von der Anzahl der Ausgabezeilen wahrscheinlich auf die Anzahl der Studierenden schließen können, und das Duplikat nicht entfernen.

    Man hätte dann wohl besser auch die Matrikelnummer mit ausgegeben.
- In der Klausur versuchen wir das durch zusätzliche Schlüssel eindeutig zu machen (z.B. hier: Vor- und Nachname).
- Wenn Sie in einer Firma bei jeder Anfrage `DISTINCT` schreiben, wird Ihr Chef wahrscheinlich nicht einverstanden sein (der Optimierer eliminiert das eher nicht).

Duplikate in SQL (3)

- Wir erwarten also, dass Sie in der Lage sind, Anfragen daraufhin zu analysieren, ob sie in irgendeinem Zustand Duplikate liefern.

  Zumindest in hinreichend einfachen Fällen. Im allgemeinen ist das Problem unentscheidbar — es gibt also keinen Algorithmus der das für beliebige Anfragen könnte.

- SQL erzeugt eine Ausgabezeile für jede Belegung der Tupelvariablen unter `FROM`, die die `WHERE`-Bedingung erfüllt.
- Die Frage ist also, ob es zwei verschiedene Belegungen geben kann,
  - die beide die `WHERE`-Bedingung erfüllen, und
  - für die alle unter `SELECT` genannten Terme die gleichen Werte haben.

Duplikate in SQL (4)

- Weil Duplikate so eng mit dem zentralen Begriff der Variablenbelegung zusammenhängen, macht es Sinn, zu fordern, dass Sie in der Lage sind, Duplikate vorherzusagen.
- Dadurch zeigen Sie, dass Sie verstanden haben, wie SQL funktioniert.

Wichtige Klarstellung:

- Duplikate beziehen sich immer auf ganze Ergebniszeilen.

  In den folgenden Beispielen gibt es häufig nur eine Ergebnisspalte, deswegen wird der Unterschied möglicherweise nicht ganz deutlich. Gleiche Werte in einer Spalte wären aber noch kein Duplikat, wenn es noch weitere Ergebnisspalten gibt. Nur wenn zwei Ergebniszeilen in allen Spalten übereinstimmen, ist es ein Duplikat.

Erstes Beispiel zu Duplikaten (1)

Aufgabe:

- Gegeben sei die Tabelle

  STUDENTEN(SID, VORNAME, NACHNAME, EMAIL°)

- Die Kombination VORNAME, NACHNAME sei auch Schlüssel.

  Neben dem Primärschlüssel SID.

- Kann folgende Anfrage Duplikate liefern?

```
SELECT S.NACHNAME
FROM   STUDENTEN S
WHERE  S.VORNAME LIKE 'Alex%'
```

  Ja/Nein-Abstimmung.

  Die Frage ist nicht, ob hier Dupliate sinnvoll wären, sondern nur, ob sie in irgendeinem DB-Zustand herauskommen können.

Erstes Beispiel zu Duplikaten (2)

- Ja, z.B. in folgendem Zustand:

STUDENTEN			
SID	VORNAME	NACHNAME	EMAIL
101	Alexandra	Müller	a_mueller@acm.org
102	Alexander	Müller	alex_m99@gmail.com

- In diesem Zustand liefert die Anfrage folgende Ausgabe:

NACHNAME
Müller
Müller

Die Duplikate entstehen dadurch, dass die anderen Spalten ausgeblendet werden. (In der Relationenalgebra wäre das die Operation „Projektion“.)

Erstes Beispiel zu Duplikaten (3)

- Variablenbelegung $\mathcal{A}_1$ erzeugt Ausgabe „Müller“:

	STUDENTEN			
	SID	VORNAME	NACHNAME	EMAIL
S →	101	Alexandra	Müller	...
	102	Alexander	Müller	...

- Variablenbelegung $\mathcal{A}_2$ erzeugt auch Ausgabe „Müller“:

	STUDENTEN			
	SID	VORNAME	NACHNAME	EMAIL
	101	Alexandra	Müller	...
S →	102	Alexander	Müller	...

Erstes Beispiel zu Duplikaten (4)

- Damit hat man für diesen Zustand zwei Variablenbelegungen,
  - die beide die `WHERE`-Bedingung erfüllen,
  - beide die gleiche Ausgabe (unter `SELECT`) erzeugen.
- Man kann das auch mit dem naiven Auswertungsalgorithmus (Schleife über allen Tabellenzeilen) verstehen:

```
for S in STUDENTEN do
    if like(S.VORNAME, 'Alex%') then
        print S.NACHNAME
```

Mit der Schleife werden gerade die möglichen Variablenbelegungen durchgegangen.

- Im Beispiel wären die Duplikate wohl erwünscht.

Algorithmus für mögliche Duplikate (1)

- In der Vorlesung ist ein Algorithmus angegeben, der eine hinreichende Bedingung für ein überflüssiges `DISTINCT` enthält:
  - Wenn der Algorithmus ausgibt, dass die Anfrage (ohne `DISTINCT`) keine Duplikate liefern kann, ist das sicher.
  - Im anderen Fall weiss man nicht, ob die Anfrage wirklich Duplikate liefern kann, oder der Algorithmus zu schwach ist.

    Da die Frage unentscheidbar ist, kann es keinen Algorithmus geben, der immer eine definitive (und korrekte) Antwort gibt. Man kann das Duplikatsproblem auf das Konsistenzproblem zurückführen und umgekehrt.
- Der Algorithmus sollte für die Klausuraufgaben ausreichen.

  Sie können die Frage nach Duplikaten natürlich auch weniger formal klären. Stellen Sie sich die (geschachtelten) Schleifen über den Tabellen vor (eine für jede Tupelvariable).

Algorithmus für mögliche Duplikate (2)

- Der Algorithmus verwendet eine Menge $\mathcal{K}$
  - von Attributreferenzen, also Einträgen der Form $\langle\text{Tupelvariable}\rangle\,.\,\langle\text{Attribut}\rangle$,
  - von denen bekannt ist, dass sie für eine gegebene Ausgabezeile nur einen Wert haben können.
- Das gilt insbesondere für Attribut-Referenzen, die direkt unter `SELECT` stehen, im Beispiel also `S.Nachname`.
- Falls nun die Menge $\mathcal{K}$ einen Schlüssel von jeder Tupelvariable enthält, kann es zu einer gegebenen Ausgabezeile nicht zwei verschiedene Variablenbelegungen geben.

  Die Ausgabezeile bestimmt dann ja die Schlüsselwerte eindeutig. Zu einer gegebenen Ausgabezeile kann es dann nur jeweils eine Tabellenzeile geben, die man der Tupelvariablen zuweisen kann, um diese Ausgabezeile zu erzeugen.

Algorithmus für mögliche Duplikate (3)

- Im Beispiel ist: $\mathcal{K} = \{\texttt{S.Nachname}\}$.
- `Nachname` ist kein Schlüssel der Tabelle, über die `S` läuft (`STUDENTEN`).
- Daher sagt der Algorithmus nichts aus.

  Bzw.: „Die Anfrage könnte eventuell Duplikate liefern“.
- Um ein konkretes Beispiel zu finden, versuche man zwei Variablenbelegungen für die Tupelvariablen zu konstruieren,
  - die die gleichen Werte für alle Elemente von $\mathcal{K}$ haben, und
  - die `WHERE`-Bedingung erfüllen.

    Im allgemeinen greift hier die Unentscheidbarkeit des Konsistenzproblems. Für einfache `WHERE`-Bedingungen wäre das natürlich automatisch möglich. D.h. man könnte den Algorithmus so ausbauen, dass er zumindest manchmal ein klares „Duplikate sind möglich“ ausgibt.

Zweites Beispiel zu Duplikaten

Aufgabe:

- Gegeben sei die Tabelle

  STUDENTEN(SID, VORNAME, NACHNAME, EMAIL°)

- Die Kombination VORNAME, NACHNAME sei auch Schlüssel.

  Neben dem Primärschlüssel SID.

- Kann folgende Anfrage Duplikate liefern?

```
SELECT S.NACHNAME
FROM   STUDENTEN S
WHERE  S.VORNAME = 'Alex'
```

  Ja/Nein-Abstimmung.

Zweites Beispiel zum Duplikat-Algorithmus

- Die Menge $\mathcal{K}$ wird wie eben mit Attributreferenzen initialisiert, die direkt unter SELECT als Ausgabespalte stehen:

$$\mathcal{K} = \{\texttt{S.NACHNAME}\}.$$

  Um eventuelle Alternativen mit OR auszuschließen, wandelt man die Bedingung zunächst in konjunktive Normalform um (eine mit AND-verknüpfte Liste), und schaut dann, ob ein Element dieser Liste die Form $X.A = c$ hat.

- Dann werden alle Attribut-Referenzen hinzugefügt, die aufgrund der WHERE-Klausel einen festen Wert haben:

$$\mathcal{K} = \{\texttt{S.NACHNAME}, \texttt{S.VORNAME}\}.$$

- Da VORNAME und NACHNAME zusammen einen Schlüssel von STUDENTEN bilden (über dieser Tabelle läuft S), garantiert der Algorithmus, dass diese Anfrage keine Duplikate liefern kann.

Einfache Aufgaben zu Duplikaten (1)

a)

- Kann diese Anfrage Duplikate liefern?

```
SELECT SID
FROM   STUDENTEN
WHERE  NACHNAME LIKE 'M%'
```

- Wie eben sind die Schlüssel von `STUDENTEN`:
  - `SID` (Primärschlüssel)
  - `VORNAME`, `NACHNAME` (Sekundärschlüssel)
- Ja/Nein-Abstimmung.
- Was ist $\mathcal{K}$?

Einfache Aufgaben zu Duplikaten (2)

a)

- Nein (kann keine Duplikate liefern).
- $\mathcal{K} = \{\texttt{STUDENTEN.SID}\}$.

b)

- Kann diese Anfrage Duplikate liefern?

```
SELECT VORNAME
FROM   STUDENTEN
WHERE  SID = 153
```

- Schlüssel wie gehabt.

  {SID} und {VORNAME, NACHNAME}.

- Was ist $\mathcal{K}$?

Einfache Aufgaben zu Duplikaten (3)

b)

- Nein (kann keine Duplikate liefern).
- $\mathcal{K} = \{\texttt{STUDENTEN.VORNAME}, \texttt{STUDENTEN.SID}\}$.

  SID ist Schlüssel von STUDENTEN.

  Es ist hier unwesentlich, dass VORNAME auch in $\mathcal{K}$ enthalten ist.

c)

- Kann diese Anfrage Duplikate liefern?

```
SELECT VORNAME
FROM   STUDENTEN
WHERE  VORNAME = 'Dorothea'
```

- Was ist $\mathcal{K}$?

Einfache Aufgaben zu Duplikaten (4)

c)

- Ja (kann Duplikate liefern).

  Es kann mehrere Studentinnen geben, die Dorothea heißen.

- $\mathcal{K} = \{\texttt{STUDENTEN.VORNAME}\}$.

  VORNAME allein ist nicht Schlüssel von Studenten.

d)

- Kann diese Anfrage Duplikate liefern?

```
SELECT SID
FROM   STUDENTEN
WHERE  VORNAME = 'Lisa' OR NACHNAME = 'Weiss'
```

Einfache Aufgaben zu Duplikaten (5)

d)

- Nein (keine Duplikate).

  Obwohl es „Lisa Weiss“ gibt, die beide Teile der Disjunktion erfüllt, wird sie nicht doppelt geliefert: Jede Variablenbelegung wird nur ein Mal getestet.

- $\mathcal{K} = \{\texttt{STUDENTEN.SID}\}$.

  `SID.VORNAME` und `SID.NACHNAME` sind nicht dabei, weil die `WHERE`-Bedingung nicht eine einzelne Konstante erzwingt (durch das `OR`).

e)

- Kann diese Anfrage Duplikate liefern?

```
SELECT VORNAME
FROM   STUDENTEN
WHERE  SID <= 101 AND SID >= 101
OR     SID >  200 AND SID <= 200
```

Einfache Aufgaben zu Duplikaten (6)

e)

- Nein, die Bedingung ist äquivalent zu `SID = 101`.

  Die Anfrage liefert also nur höchstens eine Zeile.

  Der Teil nach dem `OR` ist inkonsistent.

- Der Algorithmus liefert $\mathcal{K} = \{\texttt{STUDENTEN.VORNAME}\}$ und deswegen „ich weiss nicht/Duplikate eventuell möglich“.

  Äquivalenz ist unentscheidbar. Der Algorithmus findet $X.A = c$ nur, wenn es syntaktisch offensichtlich ist.

f)

- Kann diese Anfrage Duplikate liefern?

```
SELECT X.SID
FROM   STUDENTEN X, STUDENTEN Y
```

Einfache Aufgaben zu Duplikaten (7)

f)

- Ja, Duplikate sind möglich: Die Tupelvariable `Y` erzeugt eine zusätzliche Schleife über den `STUDENTEN`.

```
for X in STUDENTEN do
    for Y in STUDENTEN do
        print X.SID
```

- Wenn `STUDENTEN` n Zeilen hat, werden n^2 Ergebniszeilen ausgegeben, und zwar die n verschiedenen `SID`s jeweils n mal.

  D.h. im Fall $n = 0$ und $n = 1$ gibt es keine Duplikate. Die Frage ist aber immer, ob es mindestens einen DB-Zustand gibt, in dem die Anfrage Duplikate erzeugt. Das gilt hier natürlich.

- $\mathcal{K} = \{\texttt{X.SID}\}$.
  Dies enthält zwar einen Schlüssel von `X`, aber nicht von `Y`.

Inhalt

Hausaufgabe 6.1: Logik-Rätsel (1)

- Ein König lässt eine junge Dame einsperren, weil sie für mehr Umweltschutz demonstriert hat. Ihr Verlobter möchte sie gern befreien. Der König ist ein Fan von Logik-Rätseln, und stellt den Verlobten vor die Wahl zwischen drei Türen:
  - In einem der drei Räume ist seine Herzens-Dame. Wenn er diese Tür öffnet, darf er mit ihr gehen.
  - In einem anderen Raum ist ein hungriger Tiger. Wenn er diese Tür öffnet, wird er höchstwahrscheinlich aufgefressen.
  - Der dritte Raum ist leer. Er überlebt dann zwar, muss aber ohne seine Liebste gehen.

Dieses Logikrätsel stammt aus dem Buch „Dame oder Tiger“ von Raymond Smullyan. Sie können Auszüge aus dem titel-gebenden Abschnitt des Buches auf dieser Webseite finden: [https://emath.de/Referate/Smullyan.pdf]
Ich habe die Geschichte um das Rätsel leicht verändert.

Hausaufgabe 6.1: Logik-Rätsel (2)

- An den drei Räumen gibt es Schilder mit folgenden Inschriften:
  - Raum 1: „Raum 3 ist leer“.
  - Raum 2: „In Raum 1 ist ein Tiger“.
  - Raum 3: „Dieser Raum ist leer“.
- Um die Sache etwas komplizierter zu machen, gelten folgende Regeln:
  - Wenn die Dame in dem Raum ist, ist die Inschrift des Schildes wahr.
  - Die Inschrift des Schildes an dem Raum, in dem der Tiger ist, ist falsch.
  - Die Inschrift des Schildes am leeren Raum kann wahr oder falsch sein.

Hausaufgabe 6.1: Logik-Rätsel (3)

- Wir codieren die Möglichkeiten als Zeilen einer Tabelle:

MOEGLICH		
Raum1	Raum2	Raum3
D	T	
D		T
T	D	
	D	T
T		D
	T	D

Hausaufgabe 6.1: Logik-Rätsel (4)

- Aus den Inschriften und den Zusatzbedingungen ergeben sich folgende logischen Formeln (jeweils zwei Wenn-Dann-Bedingungen pro Schild):
  - `Raum1 = 'D'` $\rightarrow$ `Raum3 = ' '`
  - `Raum1 = 'T'` $\rightarrow \neg$ `Raum3 = ' '`
  - `Raum2 = 'D'` $\rightarrow$ `Raum1 = 'T'`
  - `Raum2 = 'T'` $\rightarrow \neg$ `Raum1 = 'T'`
  - `Raum3 = 'D'` $\rightarrow$ `Raum3 = ' '`
  - `Raum3 = 'T'` $\rightarrow \neg$ `Raum3 = ' '`

In SQL gibt es leider kein „$\rightarrow$“ (wenn-dann), aber im Skript steht, wie man es durch `NOT` und `OR` ersetzen kann. In den obigen Formeln steht $\neg$ für die Negation `NOT` (wenn ein Tiger im Raum ist, ist die Inschrift des Schildes falsch).

Hausaufgabe 6.1: Logik-Rätsel (5)

- Ihre Aufgabe ist nun, die obigen Bedingungen in eine `WHERE`-Bedingung in SQL zu überführen (alle sechs Bedingungen müssen gelten, also mit `AND` verknüpft werden).
- Sie dürfen, wenn Sie möchten, `NOT` entfernen, indem Sie den jeweiligen Vergleich invertieren.
- Ansonsten bleiben Sie bitte möglichst nahe an den gegebenen Bedingungen (in der Geschichte wäre es ja fatal, wenn irgendein Fehler geschehen würde).

Hausaufgabe 6.1: Logik-Rätsel (6)

- Leider gibt es die Tabelle MOEGLICH nicht in unserer Adminer-Installation, aber man kann auf folgende Weise eine lokale Tabelle nur für die eine Anfrage definieren:

```
WITH MOEGLICH(Raum1, Raum2, Raum3) AS
    (VALUES
        ('D', 'T', ' '),
        ('D', ' ', 'T'),
        ('T', 'D', ' '),
        (' ', 'D', 'T'),
        ('T', ' ', 'D'),
        (' ', 'T', 'D'))
SELECT *
FROM MOEGLICH
```

[https://users.informatik.uni-halle.de/~brass/db23/homework/h6_dt.sql]

Hausaufgabe 6.1: Logik-Rätsel (7)

- Auch so kann man die Tabelle definieren:

```
WITH Bewohner(B) AS
    (VALUES ('D'),
            ('T'),
            (' ')),
MOEGLICH(Raum1, Raum2, Raum3) AS
    (SELECT R1.B AS Raum1, R2.B AS Raum2,
            R3.B AS Raum3
    FROM Bewohner R1, Bewohner R2, Bewohner R3
    WHERE R1.B <> R2.B
    AND   R1.B <> R3.B
    AND   R2.B <> R3.B)
SELECT *
FROM MOEGLICH
```

Hausaufgabe 6.1: Logik-Rätsel (8)

- Für diese Aufgabe ist wesentlich, dass man die folgende Äquivalenz kennt:
  - $F \to G$ ist äquivalent zu $\neg F \vee G$, bzw. in SQL: `NOT F OR G`.
- Eine „Wenn-Dann-Bedingung“ ist erfüllt gdw.
  - Die Voraussetzung falsch ist, oder

    Dann ist egal, ob die Folgerung wahr oder falsch ist. Es ist ja keine „Genau-dann-wenn-Bedingung“.
  - die Folgerung wahr ist.

    Dann ist egal, ob die Voraussetzung wahr oder falsch ist.
- Dies ist eine nützliche Äquivalenz, die man sich merken sollte.

  Man braucht diese Äquivalenz z.B. für `CHECK`-Constraints, weil es kein $\to$ in SQL gibt, aber natürlich schon `NOT` und `OR`.

Hausaufgabe 6.1: Logik-Rätsel (9)

- Lösung:

```
SELECT *
FROM   MOEGLICH
WHERE  -- Schild an Raum 1: Raum 3 ist leer.
       (NOT Raum1 = 'D' OR Raum3 = ' ')
AND    (NOT Raum1 = 'T' OR NOT Raum3 = ' ')
       -- An Raum 2: In Raum 1 ist ein Tiger.
AND    (NOT Raum2 = 'D' OR Raum1 = 'T')
AND    (NOT Raum2 = 'T' OR NOT Raum1 = 'T')
       -- An Raum 3: Dieser Raum ist leer.
AND    (NOT Raum3 = 'D' OR Raum3 = ' ')
AND    (NOT Raum3 = 'T' OR NOT Raum3 = ' ')
```

Hausaufgabe 6.1: Logik-Rätsel (10)

- Man kann noch jeweils das `NOT` eliminieren, indem man = in <> umwandelt:

```
SELECT *
FROM   MOEGLICH
WHERE  -- Schild an Raum 1: Raum 3 ist leer.
       (Raum1 <> 'D' OR Raum3 =  ' ')
AND    (Raum1 <> 'T' OR Raum3 <> ' ')
       -- An Raum 2: In Raum 1 ist ein Tiger.
AND    (Raum2 <> 'D' OR Raum1 =  'T')
AND    (Raum2 <> 'T' OR Raum1 <> 'T')
       -- An Raum 3: Dieser Raum ist leer.
AND    (Raum3 <> 'D' OR Raum3 =  ' ')
AND    (Raum3 <> 'T' OR Raum3 <> ' ')
```

Hausaufgabe 6.1: Logik-Rätsel (11)

- Abgegebene Lösung mit gaaanz vielen Klammern:

```
SELECT *
FROM MOEGLICH
WHERE (( NOT ( Raum1 = 'D' ) ) OR ( Raum3 = ' ' ) )
AND   (( NOT ( Raum1 = 'T' ) ) OR ( NOT( Raum3 = ' ' ) ))
AND   (( NOT ( Raum2 = 'D' ) ) OR ( Raum1 = 'T' ) )
AND   (( NOT ( Raum2 = 'T' ) ) OR ( NOT( Raum1 = 'T' ) ))
AND   (( NOT ( Raum3 = 'D' ) ) OR ( Raum3 = ' ' ) )
AND   (( NOT ( Raum3 = 'T' ) ) OR ( NOT( Raum3 = ' ' ) ))
```

- Man muss die Prioritäten der logischen Operatoren lernen: `NOT` bindet am stärksten, dann `AND` und zum Schluss `OR`.

  So, wie es ist, hat man Schierigkeiten, die jeweils zugehörigen Klammern zu finden.

Hausaufgabe 6.1: Logik-Rätsel (12)

- Die Lösung ist diese Zeile:

Raum1	Raum2	Raum3
D	T	

- Diese Lösung enthält einen fatalen Fehler (`''` statt `' '`):

```
SELECT *
FROM   MOEGLICH
WHERE  (not raum1='D' OR raum3='')
AND    (not raum1='T' OR not raum3='')
AND    (not raum2='D' OR raum1='T')
AND    (not raum2='T' OR not raum1='T')
AND    (not raum3='D' OR raum3='')
AND    (not raum3='T' OR not raum3='')
```

Raum1	Raum2	Raum3
T	D	

Hausaufgabe 6.2: Logische Analyse (1)

- Schema president_public im Adminer:
  - state(state_name, admin_entered, year_entered)
  - president(pres_name, birth_year, years_serv, death_age, party, state_born→state)
  - pres_hobby(pres_name→president, hobby)
  - administration(admin_nr, pres_name→president, year_inaugurated)
  - admin_pr_vp((admin_nr, pres_name)→administration, vice_pres_name)
  - pres_marriage(pres_name→president, spouse_name, pr_age, sp_age, nr_children, mar_year)
  - election(election_year, candidate, votes, winner_loser_indic)

Hausaufgabe 6.2: Logische Analyse (2)

- Anfrage 1:

```
SELECT pres_name
FROM   president
WHERE  (state_born = 'Arizona'
         OR state_born = 'Nevada')
AND    party = 'Democratic'
```

- Anfrage 2:

```
SELECT pres_name
FROM   president
WHERE  party = 'Democratic'
       AND state_born = 'Arizona'
OR     party = 'Democratic'
       AND state_born = 'Nevada'
```

Hausaufgabe 6.2: Logische Analyse (3)

- 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. Die Teile „E.“ und „F.“ beziehen sich auch auf die Tupel-Mengen ohne Berücksichtigung eventueller Duplikate.

Hausaufgabe 6.2: Logische Analyse (4)

- Geben Sie den Buchstaben der ersten korrekten Aussage ab sowie eine kurze Begründung.

  Maximal 3 Zeilen, die entscheidenen Stichworte reichen. Bitte laden Sie eine `.txt`-Datei mit Ihrer Antwort hoch (kein PDF, kein Word).

- Beachten Sie, dass Sie alle möglichen Datenbank-Zustände in Betracht ziehen müssen, nicht nur den aktuellen Zustand.

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

- Noch gibt es keine Präsidenten, die in Arizona oder Nevada geboren sind. Im Beispiel-Zustand liefern beide Anfragen die leere Menge.

  Das war etwas unclever von mir, weil Sie deswegen vielleicht A. ankreuzen.

  Das wäre auch richtig, aber aus anderen Gründen.

Hausaufgabe 6.2: Logische Analyse (5)

- Als Schulstoff sollte bekannt sein, dass für $+$ und $*$ das Kommutativgesetz gilt:
  - $A + B = B + A$
  - $A * B = B * A$
- In der Vorlesung wurde gelehrt, dass das auch für $\wedge$/`AND` und $\vee$/`OR` gilt:
  - `A AND B` $\equiv$ `B AND A`
  - `A OR B` $\equiv$ `B OR A`

Hausaufgabe 6.2: Logische Analyse (6)

- Bedingung von Anfrage 1:

```
WHERE (state_born = 'Arizona'
       OR state_born = 'Nevada')
AND   party = 'Democratic'
```

- Nach Anwendung des Kommutativgesetzes auf AND:

```
WHERE party = 'Democratic'
AND   (state_born = 'Arizona'
       OR state_born = 'Nevada')
```

Hausaufgabe 6.2: Logische Analyse (7)

- Aus der Mathematik ist das Distributiv-Gesetz bekannt:

$$A * (B + C) = A * B + A * C$$

Man könnte die rechte Seite auch $(A * B) + (A * C)$ schreiben, aber Punktrechnung geht ohnehin vor Strichrechnung.

- Das gilt auch mit den logischen Operationen:

```
A AND (B OR C) ≡ A AND B OR A AND C
```

Auch hier könnte man die rechte Seite auch `(A AND B) OR (A AND C)` schreiben, aber `AND` bindet sowieso stärker als `OR`.

- Im Beispiel ist:
  - A: `party = 'Democratic'`
  - B: `state_born = 'Arizona'`
  - C: `state_born = 'Nevada'`

Hausaufgabe 6.2: Logische Analyse (8)

- Anfrage 1 nach Anwendung des Kommutativgesetzes auf `AND` (s.o.):

```
WHERE party = 'Democratic'
AND   (state_born = 'Arizona'
       OR state_born = 'Nevada')
```

  Man bekommt Anfrage 2 durch Anwendung des Distributivgesetzes:

```
WHERE party = 'Democratic'
      AND state_born = 'Arizona'
OR    party = 'Democratic'
      AND state_born = 'Nevada'
```

- Richtige Antwort: A, minimale Begründung: Distributivgesetz.

Hausaufgabe 6.3: Logische Analyse (1)

- Wie bei Aufg. 2 vergleichen Sie bitte die folgenden Anfragen.
- Anfrage 1:

```
SELECT pres_name
FROM   president
WHERE  NOT (years_serv > 1 OR
            party <> 'Democratic')
```

- Anfrage 2:

```
SELECT pres_name
FROM   president
WHERE  NOT years_serv > 1 AND
       party = 'Democratic'
```

Hausaufgabe 6.3: Logische Analyse (2)

- Antwort A ist korrekt: Die beiden Anfragen sind äquivalent.
- Dies ist eine Anwendung des De Morgan'schen Gesetzes:
  - $\neg(F \vee G) \equiv (\neg F) \wedge (\neg G)$.
  - In SQL: `NOT (F OR G) ≡ NOT F AND NOT G`
  - Es gilt auch die duale Version des Gesetzes, mit `AND` und `OR` vertauscht (wird aber in dieser Aufgabe nicht gebraucht).
- Außerdem wurde folgende Vereinfachung durchgeführt:
  - `NOT party <> 'Democratic'`
  - `party = 'Democratic`

Hausaufgabe 6.4: Logische Analyse (1)

- Wie bei Aufg. 2 vergleichen Sie bitte die folgenden Anfragen.
- Anfrage 1:

```
SELECT pres_name
FROM   president
WHERE  state_born = 'Arizona'
```

- Anfrage 2:

```
SELECT p.pres_name
FROM   president p, pres_marriage m
WHERE  p.pres_name = m.pres_name
AND    'Arizona' = p.state_born
```

Beachten Sie, dass der Name der Tupelvariable nicht Bestandteil des Namens der Ergebnisspalte wird (d.h. durch die Verwendung einer Tupelvariable in einem `SELECT`-Ausdruck wird die Äquivalenz nicht gestört).

Hausaufgabe 6.4: Logische Analyse (2)

- Anfrage 1 liefert Präsidenten aus Arizona.
- Anfrage 2 liefert nur verheiratete Präsidenten aus Arizona.

  Wenn ein Präsident mehrfach verheiratet war, wird er mehrfach ausgegeben.
- Also liefert Anfrage 1 immer eine Obermenge ($\supseteq$) von Anfrage 2 (wieder Buchstabe E).
- Es war ausdrücklich gesagt, dass bei E und F Duplikate ignoriert werden sollen.

  Wenn man die Multimengen betrachtet, würde die Inklusion in keiner Richtung gelten.

Hausaufgabe 6.5: Verbund-Anfrage (1)

- Welche Präsidenten haben als Hobby „`Running`“ oder „`Walking`“?
- Geben Sie den Präsidenten-Namen, das Hobby und Geburts- und Todesjahr des Präsidenten aus.

  Das Todesjahr müssen Sie approximieren als Summe von Geburtsjahr und dem erreichten Alter (`death_age`). Die Spalte soll in der Ausgabe „Todesjahr (ca.)“ heissen.
- Offensichtlich muss Ihre Anfrage die Tabellen `pres_hobby` und `president` miteinander verknüpfen:
  - Sie brauchen `pres_hobby`, weil nur dort die Hobbies der Präsidenten gespeichert sind.
  - Sie brauchen `president`, weil nur dort das Geburtsjahr und das erreichte Alter gespeichert sind.

Hausaufgabe 6.5: Verbund-Anfrage (2)

- Das Ergebnis sollte so aussehen:

pres_name	hobby	birth_year	Todesjahr (ca.)
Adams J Q	Walking	1767	1847
Lincoln A	Walking	1809	1865
McKinley W	Walking	1843	1901
Wilson W	Walking	1856	1923
Truman H S	Walking	1884	1972
Clinton W J	Running	1946	NULL

Hausaufgabe 6.5: Verbund-Anfrage (3)

- Lösung:

```
SELECT h.pres_name, h.hobby, p.birth_year,
       p.birth_year+p.death_age
           AS "Todesjahr (ca.)"
FROM   pres_hobby h, president p
WHERE  (h.hobby = 'Running' OR
        h.hobby = 'Walking')
AND    p.pres_name = h.pres_name
```

Inhalt

Präsenzaufgabe 4: Duplikate

- Kann diese Anfrage Duplikate liefern?

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.SID = B.SID
AND    B.PUNKTE = 10
AND    B.ATYP = 'H'
```

- Laden Sie eine `.txt`-Datei hoch mit der Antwort „`Ja`“ oder „`Nein`“ und außerdem:
  - Falls Nein: Beweisen Sie die Aussage, z.B. mit dem obigen Algorithmus.
  - Falls Ja: Geben Sie einen Datenbankzustand an, in dem die Anfrage ein Duplikat liefert.