Einführung in Datenbanken

Übung 9: NOT EXISTS

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. HA 6: Logik
2. Präsenzaufg. 7
3. Unteranfragen
4. Hausaufgabe 8
5. Hausaufgabe 7
6. Präsenzaufg. 8

Datenbank: Komponisten, Musik-CDs

- Die Aufgaben dieses Übungsblattes beziehen sich auf die Datenbank mit Informationen über klassische Musik-CDs:

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

- 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 6.3: Logische Analyse (1)

- Im folgenden sollen Sie jeweils zwei Anfragen vergleichen, u.a. hinsichtlich Äquivalenz.
- 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
```

Aufgabe 6.3: Logische Analyse (2)

- 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.

Aufgabe 6.3: Logische Analyse (3)

- 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).

- Es ist daher nur begrenzt hilfreich, die Anfrage auszuprobieren.

  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.

Aufgabe 6.3: Logische Analyse (4)

Lösung:

- A. Die beiden Anfragen liefern immer die gleiche Antwort.
- Begründung:
  - Wenn es mindestens ein Stück `S` mit `S.KNR = 70` gibt, gibt es genau eine Zeile `K` in `KOMPONIST` mit `K.KNR = 70`.

    Aufgrund des Fremdschlüssels `STUECK(...,KNR`$\rightarrow$`KOMPONIST, ...)` muss es mindestens eine solche Zeile geben, aufgrund des Schlüssels `KOMPONIST(KNR, ...)` höchstens eine.
  - Wenn es kein solches Stück gibt, sind beide Anfrage-Ergebnisse leer.

Aufgabe 6.3: Logische Analyse (5)

Lösung, Forts.:

- Ausführlichere Begründung:
  - Jede Variablenbelegung $\mathcal{A}$ für S, die bei der ersten Anfrage eine Ausgabe erzeugt, kann also zu einer Variablenbelegung $\mathcal{A}'$ für K und S erweitert werden, die bei der zweiten Anfrage die gleiche Ausgabe erzeugt.

    Und es gibt genau eine solche Erweiterung. Es kommen keine Duplikate hinzu. Die Abbildung zwischen den Mengen von Variablenbelegungen mit Ausgaben ist bijektiv.
  - Hat man umgekehrt eine Variablenbelegung $\mathcal{A}'$, die bei der zweiten Anfrage eine Ausgabe erzeugt, vergisst man einfach die Belegung der Variablen K und erhält eine Variablenbelegung $\mathcal{A}$, die bei der ersten Anfrage die gleiche Ausgabe erzeugt.

Aufgabe 6.4: Logische Analyse (1)

- 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
```

Aufgabe 6.4: Logische Analyse (2)

Lösung:

- E. Anfrage 1 liefert immer eine Obermenge von Anfrage 2.

  Für alle DB-Zustände $\mathcal{I}$ gilt also: $\mathcal{I}$[Anfrage 1] $\supseteq \mathcal{I}$[Anfrage 2].

- Da `OR` schwächer bindet als `AND`, gibt die Anfrage 1 einen Komponisten mit `KNR = 70` auch dann aus, wenn er nicht vor 1500 geboren ist.

- Anfrage 2 ist dagegen äquivalent zu:

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

- Die Bedingung impliziert die Bedingung von Anfrage 1:

```
WHERE KNR = 70 OR (KNR = 80 AND GEBOREN < 1500)
```

Aufgabe 6.5: Logische Analyse (1)

- Vergleichen Sie entsprechend 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%'
```

Aufgabe 6.5: Logische Analyse (2)

Lösung:

- A. Die beiden Anfragen liefern immer die gleiche Antwort.
- Wenn die Tonart „C-dur“ ist, kann sie nicht gleichzeitig „a-moll“ sein.
- Die erste Hälfte der Disjunktion in Anfrage 1 kann also nie erfüllt sein, wenn die `WHERE`-Bedingung als Ganzes wahr ist.
- Daher kann man sie auch weglassen.
- Genauere Erklärung auf der nächsten Folie.

Aufgabe 6.5: Logische Analyse (3)

Lösung, Forts.:

- Genauer kann man die `WHERE`-Bedingung der Anfrage 1 mit dem Distributiv-Gesetz äquivalent umformen:

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

- Der erste Teil der Disjunktion ist äquivalent zu `false`, und `false OR F` ist äquivalent zu `F`.

Inhalt

1. HA 6: Logik
2. Präsenzaufg. 7
3. Unteranfragen
4. Hausaufgabe 8
5. Hausaufgabe 7
6. Präsenzaufg. 8

Beispiel-Datenbank

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 7: Duplikate (1)

- 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'
```

Es sei angenommen, dass `VORNAME`, `NACHNAME` zusammen ein Alternativschlüssel der Tabelle `STUDENTEN` sind.

- ☐ Ja ☐ Nein
  - Falls Ja: Geben Sie einen DB-Zustand an, in dem die Anfrage ein Duplikat liefert.
  - Falls Nein: Beweisen Sie die Aussage, z.B. mit dem Algorithmus aus der Vorlesung.

Präsenzaufgabe 7: Duplikate (2)

Lösung:

- Ja! (Student/in mit 10 Punkten für HA 1 und HA 2)
- Also z.B. Duplikat in folgendem DB-Zustand:

STUDENTEN			
SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	weiss@acm.org

BEWERTUNGEN			
SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	10

Präsenzaufgabe 7: Duplikate (3)

Lösung, Forts.:

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

S →

STUDENTEN			
SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	weiss@acm.org

B →

BEWERTUNGEN			
SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	10

Präsenzaufgabe 7: Duplikate (4)

Lösung, Forts.:

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

	STUDENTEN			
	SID	VORNAME	NACHNAME	EMAIL
S →	101	Lisa	Weiss	weiss@acm.org

	BEWERTUNGEN			
	SID	ATYP	ANR	PUNKTE
	101	H	1	10
B →	101	H	2	10

Präsenzaufgabe 7: Duplikate (5)

- Der Algorithmus kann nicht beweisen, dass es keine Duplikate gibt und liefert aber einen Hinweis auf das Problem:

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

- $\mathcal{K} := \{\texttt{S.VORNAME}, \texttt{S.NACHNAME}\}$
- $\mathcal{K} := \mathcal{K} \cup \{\texttt{B.PUNKTE}, \texttt{B.ATYP}\}$
- $\mathcal{K} := \mathcal{K} \cup \{\texttt{S.SID}, \texttt{S.EMAIL}\}$

  Weil $\{\texttt{S.VORNAME}, \texttt{S.NACHNAME}\} \subseteq \mathcal{K}$ ein Schlüssel von S ist.

- $\mathcal{K} := \mathcal{K} \cup \{\texttt{B.SID}\}$

Präsenzaufgabe 7: Duplikate (6)

- Die finale Menge $\mathcal{K}$ enthält also:
  - Alle Attribute von `S`:
    `S.VORNAME`, `S.NACHNAME`, `S.SID`, `S.EMAIL`.
  - Von `B` are nur folgende Attribute:
    `B.PUNKTE`, `B.ATYP`, `B.SID`.
- Damit enthält $\mathcal{K}$ keinen Schlüssel von `B`.
- Wenn es überhaupt möglich ist, ein Duplikat zu erzeugen, braucht man zwei verschiedene Werte für das fehlende Attribut `B.ANR`.

  Wenn der Algorithmus (wie hier) nicht beweisen kann, dass es keine Duplikate gibt, könnte es natürlich trotzdem sein, dass tatsächlich keine Duplikate möglich sind (wegen komplexen `WHERE`-Bedingungen oder `CHECK`-Constraints).
  Aber es gilt natürlich, dass die Elemente der Menge $\mathcal{K}$ für ein Ergebnistupel nur einen Wert haben können. Das schränkt die Suche ein.

Inhalt

1 HA 6: Logik

2 Präsenzaufg. 7

3 Unteranfragen

4 Hausaufgabe 8

5 Hausaufgabe 7

6 Präsenzaufg. 8

Syntax von Unteranfragen (1)

- `NOT EXISTS (SELECT * FROM ... WHERE ...)`

  Natürlich geht das auch ohne `NOT`, das ist aber selten.

  Statt `SELECT *` kann man eine beliebige `SELECT`-Klausel schreiben. Da es völlig egal ist, was man dort schreibt, wären komplizierte `SELECT`-Klauseln schlechter Stil. In der Logik gibt es beim Quantor auch nur die Variablendeklaration (`FROM`) und die quantifizierte Formel (`WHERE`).

  Man beachte, dass links vom `NOT EXISTS` nichts steht. Manche Studierenden verwechseln es mit `NOT IN`, wo man links einen Wertausdruck schreiben muss.

- ⟨Term⟩ `NOT IN (SELECT` ⟨Term⟩ `FROM ... WHERE ...)`

  Statt „Term“ kann man natürlich auch „Wertausdruck“ sagen.

  Auch hier kann man auch `IN` ohne `NOT` verwenden.

- ⟨Term⟩ `>= ALL (SELECT` ⟨Term⟩ `FROM ... WHERE ...)`

  Statt `ALL` sind auch `ANY` und `SOME` möglich (diese beiden Schlüsselworte sind Synonyme: mindestens eins). Statt `>=` auch andere Vergleichsoperatoren.

Syntax von Unteranfragen (2)

- ⟨Term⟩ >= (SELECT ⟨Term⟩ FROM ... WHERE ...)

  Man kann Unteranfragen mit einer Ergebnisspalte, und höchstens einer Ergebniszeile (für gegebene Variablenbelegung außen), auch wie einen Term verwenden. Auch als Teil eines größeren Terms. Auch unter SELECT.

- Sollte jemals mehr als eine Zeile geliefert werden, gibt es einen Laufzeitfehler. (Falls Ergebnis leer: Nullwert.)
  - Da SQL üblicherweise interpretiert wird (also direkt nach der Syntaxanalyse auch ausgeführt wird), scheint der Unterschied zu Syntaxfehlern zunächst nicht groß.
  - Aufgabe: Warum sind Laufzeitfehler dennoch besonders problematisch (im Vergleich mit Syntaxfehlern)?

- Unteranfragen können auch unter FROM eingesetzt werden (anstelle von Tabellen). Siehe Kapitel 11 über Sichten, WITH.

Zugriff auf Tupelvariablen/Attribut-Referenzen

- Unteranfragen können Variablen der äußeren Anfrage nutzen.
- Beispiel/Aufgabe: Was ist korrekt, was falsch?

```
SELECT SID[1], B.PUNKTE[2]
FROM   STUDENTEN S, AUFGABEN A
WHERE  A.THEMA = 'SQL' AND A.ATYP = 'H'
AND    EXISTS(SELECT NACHNAME[3]
              FROM   BEWERTUNGEN B
              WHERE  SID[4] = S.SID[5]
              AND    A.ANR = B.ANR
              AND    B.ATYP ='H'
              AND    B.PUNKTE = MAXPT)
```

Abstimmung für jede gekennzeichnete Attributreferenz: Ja: Korrekt. Nein: falsch.

STUDENTEN(SID,VORNAME,NACHNAME,EMAIL°) AUFGABEN(ATYP,ANR,THEMA,MAXPT)
BEWERTUNGEN(SID→STUDENTEN,(ATYP,ANR)→AUFGABEN,PUNKTE)

Korrelierte und unkorrelierte Unteranfragen

- Unteranfragen, die Variablen der äußeren Anfrage verwenden, nennt man „korrelierte Unteranfragen“.

  Korrelierte Unteranfragen kann man sich als parametrisiert mit Tupeln der äußeren Anfrage vorstellen. Konzeptionell werden diese Unteranfragen einmal für jede Belegung der Tupelvariablen der äußeren Anfrage ausgeführt.

- Unteranfragen, die nicht auf Variablen der äußeren Anfrage zugreifen, nennt man „unkorrelierte Unteranfragen“.

  Es genügt, eine unkorrelierte Unteranfrage nur einmal auszuführen (da das Ergebnis nicht von Tupelvariablen der äußeren Anfrage abhängt).

- Unkorrelierte `EXISTS`-Unteranfragen sind fast immer falsch!

  Es entspricht einer vergessenen Verbund-Bedingung. Die Unteranfrage ist wahr oder falsch unabhängig von den Zeilen, die gerade außen betrachtet werden. Unkorrelierte `IN`-Unteranfragen sind dagegen ok (der Verbund geschieht hier durch das `IN`).

Inhalt

1. HA 6: Logik
2. Präsenzaufg. 7
3. Unteranfragen
4. Hausaufgabe 8
5. Hausaufgabe 7
6. Präsenzaufg. 8

Datenbank der US-Präsidenten

- 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)

Aufgabe 8.1: Staaten ohne Präsidenten (1)

- Aus welchen Staaten gab es seit 1850 keinen Presidenten mehr?
- Genauer: Es gab keinen Presidenten, der in dem Staat geboren ist (`state_born`) und dessen Administration ab 1850 (einschließlich) eingesetzt wurde (`year_inaugurated`).
- Geben Sie jeweils den Namen der Staaten aus (alphabetisch **sortiert**).
- Das erwartete Ergebnis hat 30 Zeilen, siehe nächste Folie.

Aufgabe 8.1: Staaten ohne Präsidenten (2)

state_name
Alabama
Alaska
Arizona
Colorado
Delawhare
Florida
Idaho
Indiana
Kansas
Louisiana

state_name
Maine
Maryland
Michigan
Minnesota
Mississippi
Montana
Nevada
New Mexico
North Dakota
Oklahoma

state_name
Oregon
Rhode Island
South Carolina
South Dakota
Tennesee
Utah
Washington
West Virginia
Wisconsin
Wyoming

Aufg. 8.2: Staaten mit genau einem Präsidenten (1)

- Aus welchen Bundesstaaten gab es genau einen Presidenten?
- D.h. es gab einen Presidenten, der in dem Staat geboren ist, aber keinen weiteren Präsidenten, der im gleichen Staat geboren ist.

  Im Gegensatz zu Aufgabe 1 spielt das Jahr der Amtseinführung hier keine Rolle.

- Geben Sie den Namen des Staates und den Namen des Präsidenten aus und **sortieren** Sie die Ausgabe nach dem Staat.
- Die Ausgabespalten sollen `State` und `President` heißen (mit dieser Groß-/Kleinschreibung).

Aufg. 8.2: Staaten mit genau einem Präsidenten (2)

- Das erwartete Ergebnis ist:

State	President
Arkansas	Clinton W J
California	Johnson L B
Connecticut	Bush G W
Hawaii	Obama B H
Illinois	Reagan R
Iowa	Hoover H C
Kentucky	Lincoln A
Missouri	Truman H S
Nebraska	Nixon R M
New Hampshire	Pierce F
New Jersey	Cleveland G
South Carolina	Jackson A

Aufgabe 8.3: Spät verheiratete Präsidenten

- Welche Präsidenten haben im Alter von 40 und mehr erstmals geheiratet, d.h.
  - sie waren verheiratet, wobei `pr_age` mindestens 40 war,
  - aber es gibt keine andere Eheschließung des gleichen Präsidenten, wo er jünger als 40 Jahre alt war.
- Geben Sie den Namen des Präsidenten, sein Alter bei der Hochzeit und das Jahr der Eheschließung aus.
- Das erwartete Ergebnis ist:

pres_name	pr_age	mar_year
Madison J	43	1794
Cleveland G	49	1886

Aufgabe 8.4: Hobbies von zwei Präsidenten (1)

- Welche Hobbies haben nach der Datenbank genau zwei Präsidenten?
- Geben Sie das Hobby und die Namen der beiden Präsidenten aus, wobei der Name des ersten Präsidenten alphabetisch vor dem Namen des zweiten Präsidenten kommen soll.
- Sie müssen prüfen, dass es außer den beiden Präsidenten keinen weiteren Präsidenten gibt, der das gleiche Hobby hat.
- Nennen Sie die Spalten für die beiden Namen „`President 1`“ und „`President 2`“.

Aufgabe 8.4: Hobbies von zwei Präsidenten (2)

- Das erwartete Ergebnis ist:

hobby	President 1	President 2
Billards	Adams J Q	Garfield J A
Poker	Harding W G	Truman H S
Reading	Bush G W	Clinton W J
Shooting	Hayes R B	Roosevelt T

Aufgabe 8.5: Immer gewonnen (1)

- Welche Präsidenten ab 1980 (`year_inaugurated`) haben jede Wahl, zu der sie angetreten sind, gewonnen?

  Wert „`W`“ in `winner_loser_indic` in allen `election`-Zeilen mit dem Präsidenten als `candidate`.

- Geben Sie außerdem das Jahr der Amtseinführung (`year_inaugurated`) des Präsidenten aus und **sortieren** Sie nach dem Jahr.

- Manche Präsidenten hatten mehrere (zwei) Präsidentschaften. Geben Sie aber jeden Präsidenten nur einmal aus, und zwar jeweils mit dem Jahr seiner ersten Präsidentschaft (das minimale `year_inaugurated`).

  Die Grenze 1980 ist so gemeint, dass die erste Präsidentschaft frühestens 1980 startete. Bei den Wahlen sollen Sie dagegen alle berücksichtigen, ggf. auch solche vor 1980.

Aufgabe 8.5: Immer gewonnen (2)

- Das erwartete Ergebnis ist:

pres_name	year_inaugurated
Reagan R	1981
Clinton W J	1993
Bush G W	2001
Obama B H	2009
Biden J R	2021

Inhalt

1 HA 6: Logik

2 Präsenzaufg. 7

3 Unteranfragen

4 Hausaufgabe 8

5 Hausaufgabe 7

6 Präsenzaufg. 8

EMP-DEPT-Datenbank

- `dept`: „Department“ (Abteilungen einer Firma)

  `dept(deptno, dname, loc)`

- `emp`: „Employee“ (Angestellte der Firma)

  `emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)`

- `salgrade`: „Salary Grade“ (Gehaltsstufen)

  `salgrade(grade, losal°, hisal°)`

- Link zum Adminer (Schema „`empdept_public`“):

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

Aufgabe 7.1: Gehaltsvergleich (1)

- Wer verdient mehr als sein direkter Vorgesetzter?

  Die Angestelltennummer (`empno`) des Vorgesetzten steht in der Spalte `mgr` („Manager“). Das Gehalt steht in der Spalte `sal` („Salary“).

- Geben Sie Name und Gehalt des Untergebenen und Name und Gehalt des Vorgesetzten aus, und nennen Sie die Ergebnisspalten wie in der Beispiel-Ausgabe gezeigt.

  Die Spalten sollen heißen: (1) Angestellter, (2) Gehalt, (3) Vorgesetzter, (4) Gehalt des Vorgesetzten. In der Beispiel-Ausgabe auf der nächsten Folie ist die letzte Spalte etwas gekürzt.

- **Sortieren** Sie das Ergebnis nach dem Namen des Angestellten.

Aufgabe 7.1: Gehaltsvergleich (2)

- Erwartetes Ergebnis:

Angestellter	Gehalt	Vorgesetzter	Gehalt d.V.
FORD	3000	JONES	2975
SCOTT	3000	JONES	2975

Aufgabe 7.1: Gehaltsvergleich (3)

Lösung:

- Selbstverbund mit zwei Tupelvariablen über emp:

```
SELECT u.ename AS "Angestellter",
       u.sal AS "Gehalt",
       v.ename AS "Vorgesetzter",
       v.sal AS "Gehalt des Vorgesetzten"
FROM   emp u, emp v
WHERE  u.mgr = v.empno
AND    u.sal > v.sal
ORDER  by "Angestellter"
```

Das Schlüsselwort `AS` kann auch weggelassen werden.

Unter `ORDER BY` könnte man auch `u.ename` schreiben.

Aufgabe 7.2: Gehaltsspanne in Abteilungen (1)

- In welchen Abteilungen gibt es Angestellte, die um mindestens drei Gehaltsstufen auseinander liegen?

  Die Gehaltsstufen sind in der Tabelle `salgrade` definiert: Wenn ein Gehalt zwischen den Werten `losal` und `hisal` liegt (einschließlich der beiden Grenzen), wird der Angestellte der Gehaltsstufe `grade` zugeordnet. Gesucht sind Abteilungen, in denen es Angestellte mit Gehaltsstufen gibt, deren Differenz wenigstens 3 ist.

- Geben Sie Nummer und Name der jeweiligen Abteilung aus und **sortieren** Sie das Ergebnis nach der Abteilungsnummer.

deptno	dname
10	ACCOUNTING
20	RESEARCH
30	SALES

Aufgabe 7.2: Gehaltsspanne in Abteilungen (2)

Lösung:

- Dies braucht fünf Tupelvariablen:

```
SELECT DISTINCT d.deptno, d.dname
FROM   dept d, emp e1, emp e2,
       salgrade s1, salgrade s2
WHERE  d.deptno = e1.deptno
AND    d.deptno = e2.deptno
AND    e1.sal BETWEEN s1.losal AND s1.hisal
AND    e2.sal BETWEEN s2.losal AND s2.hisal
AND    s1.grade - s2.grade >= 3
ORDER  BY deptno
```

Statt `BETWEEN` kann man natürlich auch zwei Vergleiche (>=, <=) schreiben.

`DISTINCT` ist nötig, weil es mehrere Angestelltenpaare geben kann, die drei Gehaltsstufen auseinander liegen.

Aufgabe 7.3: Manager/Clerk pro Abteilung (1)

- Geben Sie für jede Abteilung, die einen „MANAGER“ und einen „CLERK“ (Sachbearbeiter, Sekretär) hat, Nummer und Name der Abteilung und den Namen des Managers und den Namen des Sachbearbeiters aus.

  Gemeint sind natürlich immer männliche und weibliche Personen mit diesen Berufsbezeichnungen.

- Es ist in Ordnung, dass, falls eine Abteilung mehrere Manager und mehrere Sachberarbeiter haben sollte, jede Kombination ausgegeben wird.

  Im Beispielzustand hat jede Abteilung aber nur jeweils einen Angestellten mit diesen Berufsbezeichnungen.

- Verwenden Sie die Spaltennamen wie im Beispiel-Ergebnis und **sortieren** Sie die Ausgabe nach der Abteilungsnummer.

Aufgabe 7.3: Manager/Clerk pro Abteilung (2)

- Erwartetes Ergebnis:

deptno	dname	Manager	Clerk
10	ACCOUNTING	CLARK	MILLER
20	RESEARCH	JONES	ADAMS
30	SALES	BLAKE	JAMES

Aufgabe 7.3: Manager/Clerk pro Abteilung (3)

Lösung:

- Wieder ein Selbstverbund mit zwei Tupelvariablen über emp:

```
SELECT d.deptno, d.dname,
       m.ename AS "Manager",
       c.ename AS "Clerk"
FROM   dept d, emp m, emp c
WHERE  d.deptno = m.deptno
AND    d.deptno = c.deptno
AND    m.job = 'MANAGER'
AND    c.job = 'CLERK'
ORDER  BY deptno
```

Aufgabe 7.4: Angestellte ohne Untergebene (1)

- Welche Angestellten haben selbst keinen Untergebenen?

  Gesucht sind also Angestelle `e`, so dass es keinen Angestellten `x` gibt, dessen direkter Vorgesetzter `e` ist. Die Angestellten-Nummer `empno` des direkten Vorgesetzten `e` würde in der Spalte `mgr` des Untergebenen `x` stehen.

- Geben Sie Angestellten-Nummer (`empno`), Name (`ename`), Berufsbezeichung (`job`) und Gehalt (`sal`) der gesuchten Angestellten aus.

- **Sortieren** Sie die Ausgabe absteigend nach dem Gehalt (also größtes Gehalt zuerst) und bei gleichem Gehalt nach der Angestellten-Nummer (normal, d.h. aufsteigend).

Aufgabe 7.4: Angestellte ohne Untergebene (2)

- Das erwartete Ergebnis ist:

empno	ename	job	sal
7499	ALLEN	SALESMAN	1600
7844	TURNER	SALESMAN	1500
7934	MILLER	CLERK	1300
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7369	SMITH	CLERK	800

Aufgabe 7.4: Angestellte ohne Untergebene (3)

Lösung:

- Dies ist eine nichtmonotone Anfrage, die `NOT EXISTS` erfordert:

```
SELECT empno, ename, job, sal
FROM   emp e
WHERE  not exists(SELECT * FROM emp x
                  WHERE  x.mgr = e.empno)
ORDER  BY sal desc, empno
```

Natürlich ginge `NOT IN` auch. Dann muss man aber aufpassen, weil `x.mgr` einen Nullwert enthält (für den Präsidenten der Firma). Der Nullwert muss in der Unterabfrage, die die Menge der Angestelltennummern von Vorgesetzten berechnet, explizit ausgeschlossen werden.

Aufgabe 7.5: Obere Gehaltshälfte (1)

- Welche Angestellten verdienen mindestens die Hälfte vom Top-Verdiener in der Firma?

  Sie müssen also einen Angestellten `t` mit maximalem Gehalt finden (d.h. es gibt keinen, der mehr verdient) und dann alle Angestellten `e` ausgeben, die mindestens 50% des Gehalts von `t` haben.

- Drucken Sie Nummer und Namen des Angestellten, sowie Berufsbezeichnung und Gehalt.

- **Sortieren** Sie die Ausgabe absteigend nach dem Gehalt (also größtes Gehalt zuerst) und bei gleichem Gehalt nach dem Angestellten-Namen.

Aufgabe 7.5: Obere Gehaltshälfte (2)

- Die Anfrage soll keine Duplikate liefern (in keinem DB-Zustand), aber Sie sollen auch kein unnötiges `DISTINCT` verwenden.

  Begründen Sie Ihre Entscheidung, ob Sie eine Duplikat-Eliminierung brauchen oder nicht (in einem kurzen Kommentar).

- Das erwartete Ergebnis ist:

empno	ename	job	sal
7839	KING	PRESIDENT	5000
7902	FORD	ANALYST	3000
7788	SCOTT	ANALYST	3000
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850

Aufgabe 7.5: Obere Gehaltshälfte (3)

Lösung:

- In dieser Anfrage wird `NOT EXISTS` verwendet, um das maximale Gehalt zu bestimmen. Später geht das natürlich auch mit der Aggregationsfunkion `MAX`:

```
SELECT DISTINCT e.empno, e.ename, e.job, e.sal
FROM   emp e, emp t
WHERE  NOT EXISTS(SELECT * FROM emp x
                  WHERE  x.sal > t.sal)
AND    e.sal >= 0.5 * t.sal
ORDER  BY sal DESC, ename
```

„t“ ist der Top-Verdiener.

`DISTINCT` ist nötig, weil es mehrere Angestellte mit dem gleichen maximalen Gehalt (Top-Verdiener) geben könnte.

Inhalt

1. HA 6: Logik
2. Präsenzaufg. 7
3. Unteranfragen
4. Hausaufgabe 8
5. Hausaufgabe 7
6. Präsenzaufg. 8

Präsenzaufgabe: Nichtmonotone Anfrage (1)

- Tabellen des Schemas „empdept_public“ im Adminer:
  - dept(deptno, dname, loc)
  - emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)
- Formulieren Sie die folgende Anfrage in SQL:
  - Gibt es einen Angestellten, der direkter Vorgesetzter (mgr) von allen Angestellten mit Job „SALESMAN“ ist (d.h. haben alle Verkäufer denselben Vorgesetzten)?
  - Drucken Sie ggf. empno, ename, job:

7698	BLAKE	MANAGER

Falls es keinen Angestellten mit dieser Bedingung gibt, soll das Ergebnis der Anfrage leer sein.

Falls es keinen „SALESMAN“ gibt, sollen alle Angestellten herauskommen.

Präsenzaufgabe: Nichtmonotone Anfrage (2)

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