Einführung in Datenbanken

Übung 8: Nichtmonotone Anfragen (NOT EXISTS u.a.)

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

Wintersemester 2020/21

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

Inhalt

Hausaufgabe 6a (1)

- Gesucht sind Aufnahmen von Stücken von Vivaldi und Telemann (d.h. der Name des Komponisten ist Vivaldi oder Telemann).
  - Geben Sie den Komponisten-Namen, den Titel des Stücks, die Nummer des Stücks, den Namen der CD, und die Gesamtspielzeit der CD (bzw. des CD-Packs) aus.
  - Nennen Sie die Spalte mit der Spielzeit „`min`“.
  - Sortieren Sie die Ausgabe absteigend nach der Gesamtspielzeit (d.h. größte Dauer zuerst). Bei gleicher Gesamtspielzeit nach dem Namen der CD, und falls auch der CD-Name gleich ist, nach der Nummer des Stücks (aufsteigend, von kleinen zu großen Stücknummern).

Hausaufgabe 6a (2)

- Die erwartete Antwort ist:

name	titel	snr	name	min
Telemann	Overture ...	1028	Telemann: ...	76
⋮	⋮	⋮	⋮	⋮
Vivaldi	Der Winter	1013	Vivaldi: ...	42

(27 Datensätze)

- Zwei gleich benannte Spalten in der Ausgabe sind normalerweise schlecht (hier „name“ des Komponisten und „name“ der CD).

- Es wurde in der Aufgabenstellung offenbar vergessen, eine Umbenennung dieser Spalten zu verlangen.

  Sie müssen sich an die Aufgabenstellung halten, dürfen aber selbstverständlich fragen, ob das wirklich beabsichtigt ist.

Hausaufgabe 6a (3)

- Die Komponisten-Namen (Vivaldi, Telemann) stehen nur in der Tabelle:

  KOMPONIST(KNR, NAME, VORNAME, GEBOREN, GESTORBEN°)

- Der verlangte Titel des Stücks nur in:

  STUECK(SNR, KNR→KOMPONIST, TITEL, TONART°, OPUS°)

- Der verlangte Name der CD nur in:

  CD(CDNR, NAME, HERSTELLER, ANZ_CDS,
  GESAMTSPIELZEIT)

- Schließlich benötigt man die Tabelle AUFNAHME, um mit Joins von CD zu STUECK zu kommen (oder umgekehrt):

  AUFNAHME(CDNR→CD, SNR→STUECK,
  ORCHESTER°, LEITUNG°)

Hausaufgabe 6a (4)

- Lösung:

```
SELECT   komponist.name, stueck.titel, stueck.snr,
         cd.name, cd.gesamtspielzeit AS "min"
FROM     aufnahme, cd, stueck, komponist
WHERE    aufnahme.cdnr = cd.cdnr
AND      aufnahme.snr = stueck.snr
AND      stueck.knr = komponist.knr
AND      (komponist.name = 'Telemann'
          OR komponist.name = 'Vivaldi')
ORDER BY cd.gesamtspielzeit DESC,
         cd.name, stueck.snr ASC
```

Dies ist eine Variante ohne explizite Tupelvariablen. Die implizit angelegten Tupelvariablen heißen wie die Tabellen. Sie wurden bei den Attributzugriffen auch dann angegeben, wenn das nicht unbedingt nötig wäre.

Hausaufgabe 6a (5)

- Komische `ORDER BY`-Klausel:

```
ORDER BY cd.gesamtspielzeit DESC,
         cd.name, stueck.snr ASC
```

- Es ist nicht nötig, `ASC` zu schreiben (Default, „Noiseword“).
- Wenn man es aber schreibt, sollte man es hier auch bei „`cd.name`“ schreiben.

  Es scheint unlogisch, den Default einmal zu nutzen und einmal nicht. Möglicherweise wird fälschlich angenommen, dass sich die Angabe `DESC`/`ASC` auf alle vorigen Elemente der Sortierspezifikation bezieht, solange nichts anderes gesagt wurde. Das ist aber falsch. Die Struktur der `ORDER BY`-Klausel ist zuerst eine durch Kommata getrennte Liste. In jedem einzelnen Element dieser Liste kann man `ASC` oder `DESC` angeben (oder eben nichts, was `ASC` bedeutet). In der Aufgabenstellung ist es als Kontrast zur Spielzeit genannt. Beim CD-Namen steht nichts, weil man umgekehrt alphabetisch wohl nur selten sortieren will.

Hausaufgabe 6b (1)

- Gesucht sind alle Komponisten in der Datenbank, die Zeitgenossen von Wolfgang Amadeus Mozart sind, deren Lebenszeit also mit der von Mozart überlappt.

  Inklusive der Grenzfällte, z.B. in dem Jahr geboren, in dem Mozart gestorben ist, oder umgekehrt.

  - „Wolfgang Amadeus Mozart“ selbst soll in der Auflistung nicht erscheinen.

    Name und Vorname des Komponisten sind Alternativschlüssel.

  - Geben Sie Name, Vorname, Geburts- und Todesjahr der Komponisten aus, sowie das „Alter“ des Komponisten im Geburtsjahr von Mozart.

    Das kann negativ sein (falls anderer Komponist nach Mozart geboren).

  - Sortieren Sie die Ausgabe nach dem Geburtsjahr, bei gleichem Geburtsjahr nach dem Todesjahr.

Hausaufgabe 6b (2)

- Erwartete Antwort:

name	vorname	geboren	gestorben	alter
Telemann	Georg Philipp	1681	1767	75
Scarlatti	Domenico	1685	1757	71
Händel	Georg Friedrich	1685	1759	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

(8 Datensätze)

Hausaufgabe 6b (3)

- 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.
- Allgemein gilt die Regel, dass Sie nur Daten verwenden dürfen, die explizit in der Aufgabenstellung stehen.

  Normalerweise ist die Begründung, dass der DB-Zustand sich ändern kann, und Ihre Anfrage auch dann noch funktionieren muss. Diese Begründung greift hier nicht wirklich, weil eine Änderung der Lebensdaten von Mozart nicht plausibel gemacht werden kann.

Hausaufgabe 6b (4)

- Wir wollen Sie natürlich zwingen, hier einen Selbstverbund zu nutzen.
- Sie brauchen zwei Tupelvariablen über der Relation `KOMPONIST`:
  - Eine für Wolfgang Amadeus Mozart, um Zugriff auf seine Lebensdaten zu bekommen.
  - Eine für den anderen Komponisten (der ausgegeben werden soll).
- Die `FROM`-Klausel ist daher:

```
FROM KOMPONIST mozart, KOMPONIST anderer
```

Natürlich kann man die Tupelvariablen beliebig nennen, z.B. „`gesuchter`“ oder „`ergebnis`“ statt „`anderer`“.

Hausaufgabe 6b (5)

- Der Vergleich von Intervallen ist nicht ganz einfach.

  Häufig hilft es, sich die Intervalle aufzuzeichnen.

- In diesem Fall scheint mir die Negation zunächst einfacher: Man muss zwei Lagen des anderen Intervalls ausschliessen:
  - Das Lebenzeit-Intervall des anderen Komponisten liegt ganz vor dem von Mozart (er ist gestorben, bevor Mozart geboren wurde).
  - Das Lebenszeit-Intervall des anderen Komponisten liegt ganz nach dem von Mozart (er wurde geboren, nachdem Mozart gestorben ist).
- Dies sind die einzigen Möglichkeiten, dass sich die Intervalle **nicht** überlappen. Also drückt die Negation der Disjunktion dieser Bedingungen die Überlappung aus.

Hausaufgabe 6b (6)

- Lösung:

```
SELECT anderer.name, anderer.vorname,
       anderer.geboren, anderer.gestorben,
       (mozart.geboren-anderer.geboren) AS "alter"
FROM   komponist mozart, komponist anderer
WHERE  mozart.name = 'Mozart'
AND    mozart.vorname = 'Wolfgang Amadeus'
AND    NOT(anderer.geboren > mozart.gestorben
           OR anderer.gestorben < mozart.geboren)
AND    anderer.knr <> mozart.knr
ORDER  BY anderer.geboren, anderer.gestorben
```

- Natürlich kann man jetzt das De Morgan'sche Gesetz anwenden, um das `NOT` am `OR` vorbei zu schieben.

  Anschließend dreht man die Vergleichsoperatoren um, und elimiert `NOT` ganz.

Hausaufgabe 6b (7)

- Alternative Lösung:

```
SELECT komponist.name, komponist.vorname,
       komponist.geboren, komponist.gestorben,
       (mozart.geboren-komponist.geboren) "alter"
FROM   komponist, komponist mozart
WHERE  mozart.name = 'Mozart'
AND    mozart.vorname = 'Wolfgang Amadeus'
AND    komponist.geboren <= mozart.gestorben
AND    komponist.gestorben >= mozart.geboren)
AND    komponist.knr <> mozart.knr
ORDER  BY komponist.geboren, komponist.gestorben
```

- Hier heißt die Tupelvariable für den anderen Komponisten einfach „`komponist`“.

Das Schlüsselwort `AS` für die Umbenennung des Spalte ist überflüssig.

Hausaufgabe 6c (1)

- Gesucht sind alle CDs, die Stücke von mehr als einem Komponisten enthalten (also mindestens zwei verschiedenen Komponisten).
  - Geben Sie jeweils den Namen der CD aus. Sortieren Sie die Ausgabe nach diesen Namen.
  - 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.

Hausaufgabe 6c (2)

- Erwartete Antwort:

name
Corelli,Albinoni,Scarlatti,Manfredini,...
Leopold Mozart: Sinfonia D-Dur ...
Mozart/Beethoven: Klassische Ouvertüren
Oboenkonzerte
Schlager um 1500
Tschaikowsky/Mendelssohn: Violinkonzerte

(6 Datensätze)

Hausaufgabe 6c (3)

- Man braucht natürlich eine Tupelvariable über CD, denn nur dort steht der CD-Name, der ausgegeben werden soll:

  CD(CDNR, NAME, HERSTELLER, ANZ_CDS, ...)

- Diese CD muss zwei Aufnahmen enthalten (von Stücken verschiedener Komponisten). Man benötigt also zwei Tupelvariablen über:

  AUFNAHME(CDNR→CD, SNR→STUECK, ORCHESTER, ...)

- Schließlich braucht man auch zwei Einträge in der Tabelle STUECK, um die Komponistennummern vergleichen zu können:

  STUECK(SNR, KNR→KOMPONIST, TITEL, TONART°, OPUS°)

  Die Tabelle KOMPONIST benötigt man dagegen nicht: Es reicht, die Komponisten-Nummern zu vergleichen, die schon in der Tabelle STUECK stehen.

Hausaufgabe 6c (4)

- Lösung:

```
SELECT DISTINCT c.name
FROM   cd c, aufnahme a1, aufnahme a2,
       stueck s1, stueck s2
WHERE  c.cdnr = a1.cdnr AND c.cdnr = a2.cdnr
AND    a1.snr = s1.snr AND a2.snr = s2.snr
AND    s1.knr <> s2.knr
ORDER  BY c.name
```

- Die CD-Nummer in c, a1 und a2 muss gleich sein.
  Man braucht dazu zwei Gleichungen, die letzte Gleichheit folgt mit dem Transitivitätsgesetz. Z.B. auch möglich:

```
WHERE c.cdnr = a1.cdnr AND a1.cdnr = a2.cdnr
```

Hausaufgabe 6c (5)

- So, wie die Anfrage formuliert ist, gibt es ohne `DISTINCT` auf jeden Fall Duplikate:
  - Wenn eine CD ein Stück von Bach enthält und eins von Telemann,
  - enthält sie natürlich auch eins von Telemann und eins von Bach.
- Wenn man eine Variablenbelegung hat, die die Bedingung erfüllt, kann man die Werte von `a1` und `a2` vertauschen, und die von `s1` und `s2`, und erhält eine andere Variablenbelegung, die die Bedingung auch erfüllt, und die gleiche CD liefert.

  Ohne `DISTINCT` generiert jede Variablenbelegung (die die `WHERE`-Bedingung erfüllt) eine eigene Ausgabezeile.

Hausaufgabe 6c (6)

- Man kann (und sollte) das etwas verbessern, indem man statt `s1.knr <> s2.knr` fordert: `s1.knr < s2.knr`.
- Im Beispielzustand kommt die Komponistennummer von Telemann vor der von Bach, dann müssten sich `a1` und `s1` also auf Telemann beziehen und `a2` und `s2` auf Bach.
- Leider ist das Problem damit nicht gelöst, da die CD ja z.B. ein Stück von Telemann und zwei von Bach enthalten kann, dann gibt es zwei verschiedene Belegungen für `a2` und `s2`, die die gleiche CD liefern.
- Man braucht also `DISTINCT`.

Inhalt

Präsenzaufgabe: Duplikate (1)

- Kann die folgende Anfrage Duplikate liefern?

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN LISA, STUDENTEN S,
       BEWERTUNGEN H1L, BEWERTUNGEN H1S
WHERE  LISA.VORNAME = 'Lisa'
AND    LISA.NACHNAME = 'Weiss'
AND    H1L.SID = LISA.SID
AND    H1S.SID = S.SID
AND    H1L.ATYP = 'H' AND H1S.ATYP = 'H'
AND    H1L.ANR  =  1  AND H1S.ANR  =  1
AND    H1S.PUNKTE > H1L.PUNKTE
```

Übliche Schlüssel: {SID} und {VORNAME, NACHNAME} von STUDENTEN, {SID, ATYP, ANR} von BEWERTUNGEN.

- Ja/Nein reicht (1 Punkt), Bonuspunkt für gute Begründung.

Präsenzaufgabe: Duplikate (2)

- Nein, die Anfrage liefert keine Duplikate.
- Man kann den Algorithmus anwenden, der Attribute berechnet, die für ein gegebenes Ergebnistupel einen eindeutigem Wert haben. Initialisierung:

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN LISA, STUDENTEN S,
       BEWERTUNGEN H1L, BEWERTUNGEN H1S
WHERE  LISA.VORNAME = 'Lisa'
AND    LISA.NACHNAME = 'Weiss'
AND    H1L.SID = LISA.SID
AND    H1S.SID = S.SID
AND    H1L.ATYP = 'H' AND H1S.ATYP = 'H'
AND    H1L.ANR  =  1  AND H1S.ANR  =  1
AND    H1S.PUNKTE > H1L.PUNKTE
```

Präsenzaufgabe: Duplikate (3)

- Nun hat man jeweils einen Schlüssel der Tupelvariablen LISA und S (VORNAME, NACHNAME bilden auch einen Schlüssel von STUDENTEN), damit auch alle anderen ihrer Attribute:

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN LISA, STUDENTEN S,
       BEWERTUNGEN H1L, BEWERTUNGEN H1S
WHERE  LISA.VORNAME = 'Lisa'
AND    LISA.NACHNAME = 'Weiss'
AND    H1L.SID = LISA.SID
AND    H1S.SID = S.SID
AND    H1L.ATYP = 'H' AND H1S.ATYP = 'H'
AND    H1L.ANR  =  1  AND H1S.ANR  =  1
AND    H1S.PUNKTE > H1L.PUNKTE
```

Präsenzaufgabe: Duplikate (4)

- Wenn man eine Seite einer Gleichung schon hat, bekommt man auch die andere Seite:

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN LISA, STUDENTEN S,
       BEWERTUNGEN H1L, BEWERTUNGEN H1S
WHERE  LISA.VORNAME = 'Lisa'
AND    LISA.NACHNAME = 'Weiss'
AND    H1L.SID = LISA.SID
AND    H1S.SID = S.SID
AND    H1L.ATYP = 'H' AND H1S.ATYP = 'H'
AND    H1L.ANR  =  1  AND H1S.ANR  =  1
AND    H1S.PUNKTE > H1L.PUNKTE
```

- Damit hat man Schlüssel aller Tupelvariablen und ist fertig.

Präsenzaufgabe: Duplikate (5)

- Kann die folgende Anfrage Duplikate liefern?

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN LISA, STUDENTEN S,
       BEWERTUNGEN H1L, BEWERTUNGEN H1S
WHERE  LISA.VORNAME = 'Lisa'
AND    LISA.NACHNAME = 'Weiss'
AND    H1L.SID = LISA.SID
AND    H1S.SID = S.SID
AND    H1L.ATYP = 'H' AND H1S.ATYP = 'H'
AND    H1L.ANR = H1S.ANR -- statt beides 1
AND    H1S.PUNKTE > H1L.PUNKTE
```

Übliche Schlüssel: {SID} und {VORNAME, NACHNAME} von STUDENTEN, {SID, ATYP, ANR} von BEWERTUNGEN.

- Ja/Nein plus gute Begründung (2 Punkte)

Präsenzaufgabe: Duplikate (6)

- Diese Anfrage kann Duplikate liefern.
- Man braucht dazu einen Studenten oder eine Studentin, der/die Lisa Weiss für zwei Hausaufgaben schlägt:

STUDENTEN

SID	VORNAME	NACHNAME	EMAIL
101	Lisa	Weiss	...
104	Iris	Winter	...

AUFGABEN

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

BEWERTUNGEN

SID	ATYP	ANR	PUNKTE
101	H	1	10
101	H	2	8
104	H	1	11
104	H	2	9

Inhalt

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

Monotone und nichtmonotone Anfragen (1)

- Eine Anfrage Q heißt monoton, wenn Einfügungen in die Datenbank niemals dazu führen können, dass
  - vorher korrekte Ergebnis-Zeilen
  - hinterher nicht mehr im Ergebnis erscheinen.
- Andernfalls (Einfügungen in die DB können zu Löschungen aus dem Ergebnis führen) heisst die Anfrage nichtmonoton.
- Satz: SQL-Anfragen Q der Form

$$\begin{array}{ll} \texttt{SELECT} & t_1, \ldots, t_k \\ \texttt{FROM} & R_1\ X_1,\ \ldots,\ R_n\ X_n \\ \texttt{WHERE} & F \end{array}$$

  wobei die Bedingung F keine Unteranfragen enthält (also eine quantorenfreie Formel ist), sind monoton.

Monotone und nichtmonotone Anfragen (2)

- Welche der folgenden Anfragen verhält sich monoton? Falls nein, welche Einfügung im Beispiel-Zustand würde zu einer Löschung aus dem Ergebnis führen?
  - Wer hat die wenigsten Punkte in Hausaufgabe 1?

    Ja/Nein-Abstimmung. Ja: monoton, Nein: nichtmonoton.

    Es sollen hier (und bei den anderen Fragen) nur echte Abgaben berücksichtigt werden, nicht die „implizit 0 Punkte“ wegen Nichtabgabe.
  - Wer hat nicht die meisten Punkte in Hausaufgabe 2?
  - Wer hat nicht an der Zwischenklausur teilgenommen?
  - Wer hat genau eine Hausaufgabe abgegeben?

    D.h. eine Hausaufgaben abgegeben, aber auch nur eine.
  - Wer hat keine EMail-Adresse angegeben (Nullwert)?

    Die Änderung eines Nullwertes in einen anderen Wert ist ein Update und keine Einfügung. Einfügung ist immer eine ganze Zeile.

Monotone und nichtmonotone Anfragen (3)

- Liefert diese Anfrage den Studenten / die Studentin mit den meisten Punkten für Hausaufgabe 1?

```
SELECT DISTINCT S.VORNAME, S.NACHNAME, X.PUNKTE
FROM   STUDENTEN S, BEWERTUNGEN X, BEWERTUNGEN Y
WHERE  S.SID = X.SID
AND    X.ATYP = 'H' AND X.ANR = 1
AND    Y.ATYP = 'H' AND Y.ANR = 1
AND    X.PUNKTE > Y.PUNKTE
```

- Wenn nicht, was berechnet sie?

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`).

Häufige Fehler (1)

- Wie oben erwähnt, ist die Verwendung einer unkorrelierten Unteranfrage mit `NOT EXISTS` meist falsch.
- Trifft dies auch in diesem Fall zu?
  (Es gibt eine Verbundbedingung in der Unteranfrage.)

```
SELECT VORNAME, NACHNAME
FROM   STUDENTEN S
WHERE  NOT EXISTS
       (SELECT *
        FROM   BEWERTUNGEN B, STUDENTEN S
        WHERE  B.SID = S.SID
        AND    B.ATYP = 'H' AND B.ANR = 1)
```

Häufige Fehler (2)

- Gibt es irgendein Problem mit dieser Anfrage?
  Es sollen alle Studenten ausgegeben werden, die noch nicht aktiv an der Vorlesung teilgenommen haben, d.h. weder eine Hausaufgabe gelöst, noch eine Prüfung absolviert haben.

```
SELECT S.VORNAME, S.NACHNAME
FROM   STUDENTEN S, BEWERTUNGEN B
WHERE  S.SID = B.SID
AND    NOT EXISTS (SELECT *
                   FROM   BEWERTUNGEN B
                   WHERE  S.SID = B.SID)
```

Beispiel/Aufgabe (1)

- Wählen Sie im Adminer das Schema „empdept_public“.
- Tabellen:
  - dept(deptno, dname, loc)
  - emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)
- Schreiben Sie eine SQL-Anfrage, um leere Abteilungen zu finden, also Abteilungen ohne Angestellte.
- Erwartete Antwort:

deptno	dname
40	OPERATIONS

Die Lösung mit IN hat ein Problem, s.u.

Beispiel/Aufgabe (2)

- Mögliche Lösung:

```
SELECT d.deptno, d.dname
FROM   department d
WHERE  NOT EXISTS(SELECT *
                  FROM   emp e
                  WHERE  e.deptno = d.deptno)
```

Dies wäre ein „Antijoin“: Es gibt gerade keinen Verbundpartner in `emp`.

- Warum funktioniert folgende Lösung nicht?

```
SELECT deptno, dname
FROM   department
WHERE  deptno NOT IN (SELECT deptno
                      FROM   emp)
```

Erst Abstimmung, wer es weiss, dann Auflösung mündlich/Chat.

Beispiel/Aufgabe (3)

- Das Problem ist, dass es einen Angestellten mit einem Nullwert in der Spalte `deptno` gibt (`SMITH`).
- `IN` ist äquivalent zu `= ANY`, und das wird wie eine große Disjunktion ausgewertet.
- Für die Abteilung 40 ist die `NOT IN` Bedingung:

```
NOT(40 = 10 OR 40 = 20 OR 40 = 30 OR 40 = NULL)
```

  - Beim Vergleich mit dem Nullwert ergibt sich der dritte Wahrheitswert „unbekannt“.
  - Alle anderen Glieder der Disjunktion sind falsch.
  - Damit ist die Disjunktion „unbekannt“.
  - Die Negation von „unbekannt“ ist „unbekannt“.
  - Ausgaben gibt es nur, wenn die `WHERE`-Bedingung wahr ist.

Beispiel/Aufgabe (4)

- Lösung:

```
SELECT deptno, dname
FROM   department
WHERE  deptno NOT IN (SELECT deptno
                      FROM   emp
                      WHERE  deptno IS NOT NULL)
```

- Hier wird der Nullwert explizit ausgeschlossen aus der Wertemenge, die die Unteranfrage liefert.
- Hinweis: Der Test auf einen Nullwert muss mit `IS NULL` bzw. `IS NOT NULL` ausgeführt werden.

  `= NULL` bzw. `<> NULL` funktionieren nicht: Bei einigen Systemen (und nach dem Standard) ist das ein Syntaxfehler (Das Schlüsselwort `NULL` ist kein Term.). Bei anderen liefert es den dritten Wahrheitswert „unbekannt“ (Jeder normale Vergleich mit einem Nullwert liefert „unbekannt“.).

Inhalt

Präsenzaufgabe: Nichtmonotone Anfragen

- Tabellen des Schemas „empdept_public“ im Adminer:
  - dept(<u>deptno</u>, dname, loc)
  - emp(<u>empno</u>, ename, job, mgr$^{\circ}\rightarrow$emp, hiredate, sal, comm$^{\circ}$, deptno$^{\circ}\rightarrow$dept)
- Formulieren Sie folgende Anfragen in SQL (je zwei Punkte):
  a) Wer (empno, ename) von allen Angestellten mit Job „CLERK“ verdient am meisten (sal)?

7934	MILLER	1300

  b) Gibt es einen Angestellten, der direkter Vorgesetzter (mgr) von allen Angestellten mit Job „SALESMAN“ ist (d.h. haben alle denselben Vorgesetzten)?

Drucken Sie ggf. empno, ename, job:

7698	BLAKE	MANAGER

- Bitte keine Aggregationsfunktionen (count, max) verwenden!