Einführung in Datenbanken

Übung 13: Relationale Algebra

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

Organisatorisches

- Haben Sie Fragen?
- Diese Folien stehen morgen auf der Webseite

  [https://users.informatik.uni-halle.de/~brass/db24/uebung.html]
- Bei den Übungsfolien von letzter Woche gibt es jetzt auch Lösungen.
- Die Klausur-Termine sind:
  - Montag, 03.03.2025, 10^{15}–12^{15}, Prüfungscenter der MLU
  - Montag, 14.07.2025, 10^{15}–12^{15}, Prüfungscenter der MLU
- Haben Sie die Beispiel-Klausur von letzter Woche zu Ende gerechnet?

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

Erste Anfrage in Relationaler Algebra (1)

Schreiben Sie folgende Anfrage in relationaler Algebra:

- Geben Sie die SID und die Punktzahl aller Bewertungen von Abgaben für Hausaufgabe 1 mit mindestens 8 Punkten aus.

  Schreiben Sie die Anfragen in ASCII, so dass RelaX sie versteht.
  [http://dbis-uibk.github.io/relax/calc/gist/8dc2652578ee12ae756a234c4cf21b3f]
  Auf die Webadresse wird auch unter „Übung“ auf der Webseite der Vorlesung verlinkt. Sie brauchen die URI nicht abzutippen.

Dies bezieht sich auf das bekannte Schema:

- STUDENTEN(SID, VORNAME, NACHNAME, EMAIL^o)
- AUFGABEN(ATYP, ANR, THEMA, MAXPT)
- BEWERTUNGEN(SID$\rightarrow$STUDENTEN, (ATYP, ANR)$\rightarrow$AUFGABEN, PUNKTE)

Erste Anfrage in Relationaler Algebra (2)

- Geben Sie die SID und die Punktzahl aller Bewertungen von Abgaben für Hausaufgabe 1 mit mindestens 8 Punkten aus.
- Lösung:

$$\pi_{\texttt{SID}, \texttt{PUNKTE}}\Big((\sigma_{\texttt{ATYP}=\texttt{'H'} \wedge \texttt{ANR} = 1 \wedge \texttt{PUNKTE} \geq 8}(\texttt{BEWERTUNGEN})\Big)$$

- ASCII-Schreibweise in RelaX:

```
pi SID, PUNKTE (
    sigma ATYP='H' and ANR=1 and PUNKTE >= 8 (
        BEWERTUNGEN))
```

- Ausgabe:

BEWERTUNGEN.SID	BEWERTUNGEN.PUNKTE
101	10
102	9

Erste Anfrage in Relationaler Algebra (3)

- Lösung (nochmals):

$$\pi_{\texttt{SID}, \texttt{PUNKTE}}\Big((\sigma_{\texttt{ATYP}=\texttt{'H'} \wedge \texttt{ANR} = 1 \wedge \texttt{PUNKTE} \geq 8}(\texttt{BEWERTUNGEN})\Big)$$

- Zum Vergleich:

```
SELECT SID, PUNKTE
FROM   BEWERTUNGEN
WHERE  ATYP = 'H' AND ANR = 1 AND PUNKTE >= 8
```

- Also:
  - Selektion σ entspricht der `WHERE`-Klausel.
  - Projektion π entspricht der `SELECT`-Klausel.

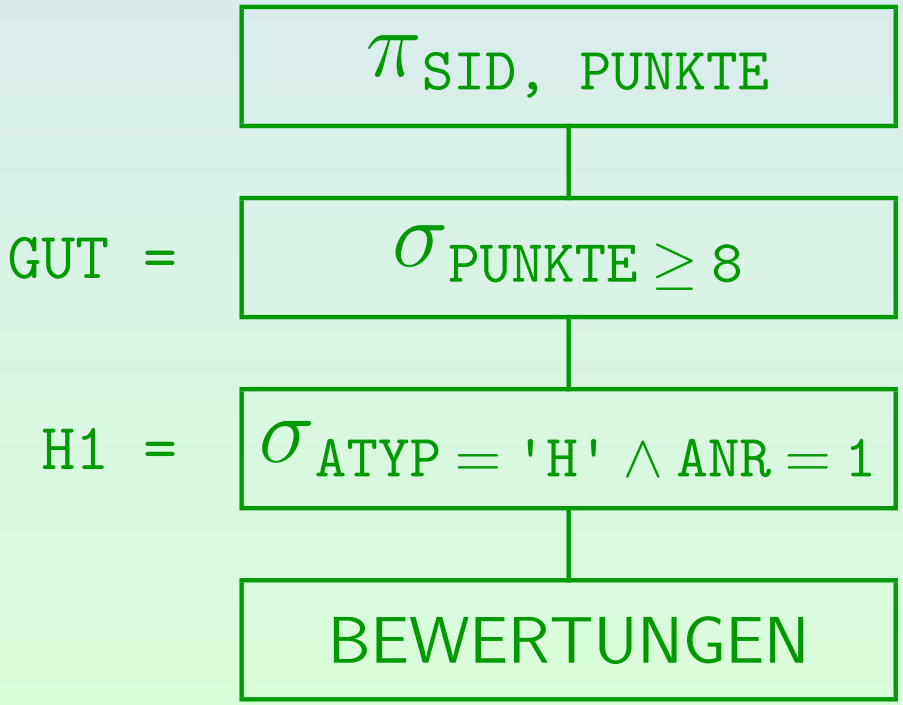
    Statt Projektion auf alle Spalten: Projektion ganz weglassen.

Erste Anfrage in Relationaler Algebra (4)

- In RelaX auch möglich mit Zwischenrelationen:

```
H1  = sigma ATYP = 'H' and ANR = 1 (BEWERTUNGEN)
GUT = sigma PUNKTE >= 8 (H1)
pi SID, PUNKTE (GUT)
```

Im Skript sind Zwischenrelationen/Sichten auch vorgesehen, dort wird „:=“ für die Zuweisung verwendet, und am Ende jeder Zuweisung steht „;“.



Aufgabe

- Welche der folgenden Ausdrücke der relationalen Algebra sind syntaktisch korrekt?
  Was bedeuten sie?

  - ☐ $\texttt{STUDENTEN}$.
  - ☐ $\sigma_{\texttt{MAXPT} \neq 10}(\texttt{AUFGABEN})$.
  - ☐ $\pi_{\texttt{VORNAME}}(\pi_{\texttt{NACHNAME}}(\texttt{STUDENTEN}))$.
  - ☐ $\sigma_{\texttt{PUNKTE} \leq 5}(\sigma_{\texttt{PUNKTE} \geq 1}(\texttt{BEWERTUNGEN}))$.
  - ☐ $\sigma_{\texttt{PUNKTE}}(\pi_{\texttt{PUNKTE} = 10}(\texttt{BEWERTUNGEN}))$.

Kartesisches Produkt und Spalten-Namen (1)

- Natürlich muss man auch mehrere Relationen verknüpfen können.
- Die Basis-Operation dazu ist das kartesische Produkt $\times$.
- Das kartesische Produkt entspricht der `FROM`-Klausel:
  Es betrachtet sämtliche Kombinationen von Tupeln aus den beiden Relationen und „klebt“ die beiden Tupel aus jedem Paar zu einem Tupel zusammen.

  Die Tupelkalkül-Sichtweise auf die `FROM`-Klausel ist etwas anders: Dort werden Variablenbelegungen betrachtet, die die Variablen jeweils auf Tupel abbilden. Dort findet das „Zusammenkleben“ nicht statt, aber es werden auch alle Kombinationen von Tupeln betrachtet.
- Wenn R die Attribute A und B hat, und S die Attribute C und D, so hat $R \times S$ die Attribute A, B, C, D.

Kartesisches Produkt und Spalten-Namen (2)

- Beispiel:

A	B
1	2
3	4

$\times$

C	D
6	7
8	9

$=$

A	B	C	D
1	2	6	7
1	2	8	9
3	4	6	7
3	4	8	9

- Da Attributnamen innerhalb eines Tupels eindeutig sein müssen, kann man das kartesische Produkt nur anwenden, wenn R und S keine gemeinsamen Attribute haben.

  Da man Spalten umbenennen kann, ist das keine echte Einschränkung.

- Das kartesische Produkt `STUDENTEN` $\times$ `BEWERTUNGEN` wäre also ein Syntaxfehler, weil es zwei Spalten mit Namen `SID` geben müsste.

Kartesisches Produkt und Spalten-Namen (3)

- RelaX akzeptiert dagegen `STUDENTEN x BEWERTUNGEN`.

  [http://dbis-uibk.github.io/relax/calc/gist/8dc2652578ee12ae756a234c4cf21b3f]

- Dort besteht der Name $R.A$ jeder Spalte aus zwei Teilen:
  - Dem Relationen-Präfix R und
  - dem eigentlichen Spalten-Name A.
- Falls der eigentliche Spalten-Name eindeutig ist, reicht der, um die Spalte zu benennen.

  Das ist so ähnlich wie bei SQL, wo der Präfix eine Tupelvariable ist.

- Bei `STUDENTEN x BEWERTUNGEN` muss man die `SID`-Spalten also mit `STUDENTEN.SID` und `BEWERTUNGEN.SID` ansprechen.
- Dagegen hört die `PUNKTE`-Spalte sowohl auf den Namen „`PUNKTE`“ wie auf den Namen „`BEWERTUNGEN.PUNKTE`“.

Kartesisches Produkt und Spalten-Namen (4)

- Im Prinzip versteht RelaX also z.B.

```
sigma STUDENTEN.SID = BEWERTUNGEN.SID
      (STUDENTEN x BEWERTUNGEN)
```

- Damit gibt es jetzt aber das Problem, dass das nicht mit dem Skript kompatibel wäre.

  Sie müssen damit rechnen, dass es einen kleinen Punktabzug gibt, wenn Sie in Klausur oder Hausaufgaben schreiben, die zwar in RelaX funktionieren, aber nach dem Skript illegal wären.

- Glücklicherweise gibt es eine Lösung, die in RelaX funktioniert UND mit dem Skript kompatibel ist:

```
sigma S.SID = B.SID
      (rho S (STUDENTEN) x rho B (BEWERTUNGEN))
```

  Mit dem rho-Operator kann man einen Präfix explizit einführen. Wenn Sie das so machen, sind sie auf der sicheren Seite.

Kartesisches Produkt und Spalten-Namen (5)

- Nach dem Skript hat jede Spalte nur einen Namen. Wenn Sie mit dem ρ_X-Operator einen Präfix X für alle Attribute eingeführt haben, müssen Sie den auch immer schreiben.
- Die Projektion erlaubt bei Bedarf auch einzelne Spalten-Umbenennungen, auch in RelaX:

```
pi NAME <- NACHNAME, EMAIL (STUDENTEN)
```

- Auch Berechnungen sind in der Projektion möglich (wie im Skript):

```
pi NAME <- concat(VORNAME, concat(' ', NACHNAME))
   (STUDENTEN)
```

- RelaX nimmt bei $\cup$ die Spalten-Namen der linken Tabelle. Damit gilt in RelaX nicht: $R \cup S = S \cup R$.

  Nach den Definitionen im Skript gilt es dagegen.

Joins/Verbunde

- Wenn der natürliche Verbund funktioniert, erspart er Ihnen die Umbenennung:

```
pi VORNAME, NACHNAME, PUNKTE
    (sigma ATYP = 'H' and ANR = 1
        (STUDENTEN join BEWERTUNGEN))
```

Er funktioniert, wenn die Join-Spalten gleiche Namen haben und die einzigen gleich benannten Spalten sind.

- Ein Join mit expliziter Bedingung ist aber auch möglich:

```
pi S.VORNAME, S.NACHNAME, B.PUNKTE
    (sigma B.ATYP = 'H' and B.ANR = 1
        ((rho S STUDENTEN) join S.SID = B.SID
            (rho B BEWERTUNGEN)))
```

Um mit dem Skript kompatibel zu sein, muss man jetzt z.B. auch `S.VORNAME` schreiben. Bei RelaX würde `VORNAME` reichen.

Häufiges Anfragemuster

- Folgende Anfragestruktur ist sehr häufig:

$$\pi_{A_1,\ldots,A_k}\Big(\sigma_F(R_1 \bowtie \cdots \bowtie R_n)\Big).$$

  - Erst verknüpft man alle Tabellen, die für die Anfrage benötigt werden, mit einem Verbund.

    Wenn die Attribute der Relationen gut benannt sind, reicht häufig ein natürlicher Verbund. Sonst muss man ggf. umbenennen und explizite Join-Bedingungen verwenden.

  - Dann selektiert man die relevanten Tupel.

    Die Selektionsbedingung kann sich auf die Attribute von allen Tabellen beziehen, die im ersten Schritt verknüpft wurden.

  - Als drittes projiziert man auf die Attribute, die ausgeben werden sollen.

Übung

Schreiben Sie folgende Anfragen in relationaler Algebra:

- Geben Sie alle Hausaufgabenergebnisse von Lisa Weiss aus (Übungsnummer und Punkte).
- Wer hat auf eine Hausaufgabe volle Punktzahl (Vorname, Nachname und Aufgabennummer)?

Dies bezieht sich auf das bekannte Schema:

- STUDENTEN(SID, VORNAME, NACHNAME, EMAIL^o)
- AUFGABEN(ATYP, ANR, THEMA, MAXPT)
- BEWERTUNGEN(SID$\rightarrow$STUDENTEN, (ATYP, ANR)$\rightarrow$AUFGABEN, PUNKTE)

Mengendifferenz

- Die Mengendifferenz kann man in RelaX „`-`“ oder „`\`“ oder „`except`“ schreiben.

  Weitere Mengenoperationen: `union` ($\cup$) und `intersect` ($\cap$).

- Z.B. Studenten ohne Hausaufgaben:

```
STUDENTEN join
    (pi SID (STUDENTEN) -
        pi SID (sigma ATYP = 'H' (BEWERTUNGEN)))
```

  Der natürliche Verbund dient hier nur als Filter: Es werden die Studenten ausgewählt, deren `SID` im Ergebnis der Mengendifferenz vorkommt.

- RelaX hat auch den im Skript erwähnten „Anti-Join“ als Abkürzung (nur das Symbol ist anders):

```
STUDENTEN anti join
    pi SID (sigma ATYP = 'H' (BEWERTUNGEN))
```

Inhalt

EMP-DEPT-Datenbank

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

  `dept(deptno, dname, loc)`

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

  `emp(empno, ename, job, mgr`$^\circ \to$`emp, hiredate, sal, comm`$^\circ$`, deptno`$^\circ \to$`dept)`

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

  `salgrade(grade, losal`$^\circ$`, hisal`$^\circ$`)`

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

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

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (1)

- Geben Sie für jede Abteilung aus:
  - Nummer und Name der Abteilung,
  - die Anzahl der Angestellten in der Abteilung,
  - die Anzahl Angestellter mit Provision (d.h. kein Nullwert in `comm`),
  - das Gesamt-Budget für die Personalkosten der Abteilung, d.h. die Summe aller Gehälter (`sal`) und Provisionen (`comm`).
- **Sortieren** Sie das Ergebnis absteigend nach dem Budget, bei gleichem Budhet nach der Abteilungsnummer.

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (2)

- Das erwartete Ergebnis ist:

deptno	dname	Angestelle	prov.	Budget
30	SALES	6	4	11600
20	RESEARCH	4	0	10075
10	ACCOUNTING	3	0	8750

- Bitte wählen Sie die Ausgabespalten wie im Beispiel.

  Die Spalte „`prov.`“ heißt eigentlich „`provisionsberechtigt`“.

- Leere Abteilungen sollen nicht in der Ausgabe erscheinen.

  Das vereinfacht die Aufgabe eher.

Aufgabe 10.1: Abteilungs-Daten inkl. Budget (3)

Lösung:

- Die Anfrage wird am besten mit `GROUP BY` gelöst:

```
SELECT d.deptno, d.dname,
       COUNT(*) AS "Angestelle",
       COUNT(comm) AS "provisionsberechtigt",
       SUM(SAL+COALESCE(comm,0)) AS "Budget"
FROM   emp e, dept d
WHERE  e.deptno = d.deptno
GROUP  BY d.deptno, d.dname
ORDER  BY "Budget" desc, deptno
```

Wenn man einfach `sal+comm` berechnet, ist das Ergebnis `NULL` für Angestellte ohne Provision (mit einem Nullwert in `comm`). Man muss also den Nullwert durch die Zahl `0` ersetzen.

Aufgabe 10.2: Job-Statistik (1)

- Geben Sie für jeden Job, den mindestens drei Angestellte haben, Folgendes aus:
  - die Anzahl Angestellter mit diesem Job,
  - die Anzahl Abteilungen, in denen der Job vorkommt, sowie
  - das minimale, maximale und durchschnittliche Gehalt aus.
- Runden Sie bitte das durchschnittliche Gehalt auf eine ganze Zahl und **sortieren** Sie die Ausgabe alphabetisch nach dem Job.

Aufgabe 10.2: Job-Statistik (2)

- Das erwartete Ergebnis ist:

job	Ang.	Abt.	von	bis	Durchschnitt
CLERK	4	3	800	1300	1038
MANAGER	3	3	2450	2975	2758
SALESMAN	4	1	1250	1600	1400

- Bitte wählen Sie die Ausgabespalten wie im Beispiel.

  Die Spalte „`Ang.`“ heißt eigentlich „`Angestellte`“ und die Spalte „`Abt.`“ eigentlich „`Abteilungen`“.

Aufgabe 10.2: Job-Statistik (3)

Lösung:

- Diese Anfrage kann gut mit `GROUP BY` und `HAVING` gelöst werden:

```
SELECT job, COUNT(*) AS "Angestellte",
       COUNT(DISTINCT deptno) AS "Abteilungen",
       MIN(sal) AS "von", MAX(sal) AS "bis",
       ROUND(AVG(sal)) AS "Durchschnitt"
FROM   emp
GROUP  BY job
HAVING COUNT(*) >= 3
ORDER  BY job
```

Aufgabe 10.3: Gehaltsklassen pro Abteilung (1)

- Geben Sie für jede Abteilung Folgendes aus:
  - Nummer und Name der Abteilung,
  - das durchschnittliche Gehalt (gerundet auf eine ganze Zahl),
  - die Anzahl Angestellter mit Gehalt bis 2000 $,
  - die Anzahl Angestellter mit Gehalt über 2000 $.
- Das erwartete Ergebnis ist:

deptno	dname	AVG	bis 2000	ueber 2000
10	ACCOUNTING	2917	1	2
20	RESEARCH	2519	1	3
30	SALES	1567	5	1

Aufgabe 10.3: Gehaltsklassen pro Abteilung (2)

- Wählen Sie die Spaltenüberschriften bitte wie im Beispiel gezeigt.

  Die Spalte „`AVG`“ heißt eigentlich „`Durchschnitt`“.

- Eine spezielle Sortierung ist nicht verlangt.

  Wenn Sie sich den Vergleich mit der Beispiel-Ausgabe erleichtern wollen, sortieren Sie nach `deptno`.

Aufgabe 10.3: Gehaltsklassen pro Abteilung (3)

Lösung:

- Hier kann man einen Trick mit CASE verwenden, und die jeweils nicht zu zählenden Zeilen auf einen Nullwert abbilden:

```
SELECT d.deptno, d.dname,
       ROUND(AVG(sal)) AS "Durchschnitt",
       COUNT(CASE WHEN SAL <= 2000 THEN 1 END)
           AS "bis 2000",
       COUNT(CASE WHEN SAL > 2000 THEN 1 END)
           AS "ueber 2000"
FROM   dept d, emp e
WHERE  d.deptno = e.deptno
GROUP  BY d.deptno, d.dname
ORDER  BY d.deptno
```

Aufgabe 10.3: Gehaltsklassen pro Abteilung (4)

Lösung, Forts.:

- Wenn man im `CASE` kein `ELSE` angibt, ist der Default `ELSE NULL`.
- Bei der Lösung spielt es keine Rolle, welchen Nicht-Nullwert man wählt. Statt `1` hätte man auch `27` schreiben können oder `'zaehlen'`.
- Wenn man sich an den Trick mit `CASE` nicht erinnert, kann man auch Unteranfragen unter `SELECT` einsetzen, wie auf der nächsten Folie gezeigt.

  Das ist das deutlich länger (es passt nur mit Auslassungen auf die Folie). Es gibt auch Code-Duplizierung für jeden Wert, den man unter `SELECT` berechnen möchte. Mit der `EXISTS`-Bedingung muss man prüfen, dass die Abteilung nicht leer ist. Sonst bekommt man auch Abteilung `40 OPERATIONS`.

Aufgabe 10.3: Gehaltsklassen pro Abteilung (5)

Lösung, Forts.:

- Lösung mit Unteranfragen unter SELECT:

```
SELECT d.deptno, d.dname,
       (SELECT ROUND(AVG(sal))
        FROM   emp e
        WHERE  e.deptno = d.deptno)
           AS "Durchschnitt",
       (SELECT COUNT(*)
        FROM   emp e
        WHERE  e.deptno = d.deptno
        AND    SAL <= 2000) AS "bis 2000",
       ... AS "ueber 2000"
FROM   dept d
WHERE  EXISTS(SELECT * FROM emp e WHERE ...)
```

Aufgabe 10.4: Maximale Anzahl Untergebene (1)

- Geben Sie den oder die Angestellten mit der maximalen Anzahl (direkter) Untergebener aus.

  Die Angestellten-Nummer des direkten Vorgesetzten steht in der Spalte `mgr`.

- Geben Sie Nummer, Name und Job des Vorgesetzten, sowie die Anzahl direkter Untergebener aus.

- Das erwartete Ergebnis ist:

empno	ename	job	Untergebene
7698	BLAKE	MANAGER	5

Aufgabe 10.4: Maximale Anzahl Untergebene (2)

Lösung:

- Lösung mit `HAVING` und `>= ALL`:

```
SELECT v.empno, v.ename, v.job,
       COUNT(*) AS "Untergebene"
FROM   emp v, emp u
WHERE  v.empno = u.mgr
GROUP  BY v.empno, v.ename, v.job
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                        FROM   emp
                        GROUP  BY mgr)
```

Aufgabe 10.4: Maximale Anzahl Untergebene (3)

Lösung, Forts.:

- Alternative mit Hilfstabelle, die zu jedem Angestellten die Anzahl Untergebenen enthält:

```
WITH   anz_unt(empno, anz) AS
       (SELECT mgr, COUNT(*)
        FROM   emp
        GROUP  BY mgr)
SELECT v.empno, v.ename, v.job,
       a.anz AS "Untergebene"
FROM   emp v, anz_unt a
WHERE  v.empno = a.empno
AND    anz = (SELECT MAX(anz)
              FROM   anz_unt)
```

Aufgabe 10.5: Nummerierung der Angestellten (1)

- Nummerieren Sie die Angestellten durch (1, 2, ...), und zwar nach dem Gehalt (größtes zuerst), und bei gleichem Gehalt nach dem Namen.

  Sie können diese Aufgabe lösen, indem Sie für jeden Angstellten zählen, wie viele Angestellte (A) mehr verdienen als der aktuelle Angestellte, oder (B) genauso viel verdienen wie der aktuelle Angestellte, aber einen Namen haben, der alphabetisch vor dem des aktuellen Angestellten kommt (oder gleich ist). Bei dieser Aufgabe wird angenommen, dass die Angestellten-Namen eindeutig sind.

- Geben Sie die berechnete Nummer, den Namen und das Gehalt für jeden Angestellten aus.
- **Sortieren** Sie die Ausgabe nach der Nummer.
- Die erwartete Antwort steht auf der nächsten Folie.

Aufgabe 10.5: Nummerierung der Angestellten (2)

Nr	ename	sal
1	KING	5000
2	FORD	3000
3	SCOTT	3000
4	JONES	2975
5	BLAKE	2850
6	CLARK	2450
7	ALLEN	1600
8	TURNER	1500
9	MILLER	1300
10	MARTIN	1250
11	WARD	1250
12	ADAMS	1100
13	JAMES	950
14	SMITH	800

Aufgabe 10.5: Nummerierung der Angestellten (3)

- Lösung, die die Anzahl von Angestellten zählt, die in der Sortierreihenfolge vor dem aktuellen Angestellten kommen:

```
SELECT COUNT(*) AS "Nr", e.ename, e.sal
FROM   emp e, emp vorher
WHERE  e.sal < vorher.sal
OR     (e.sal = vorher.sal
        AND e.ename >= vorher.ename)
GROUP  BY e.empno, e.ename
ORDER  BY "Nr"
```

Die Bedingung für `vorher` wird auch vom aktuellen Angestellten `e` erfüllt, sonst würde der erste Angestellte in der Ausgabe fehlen. So bekommt er korrekt die Nummer 1, weil er selbst als `vorher` eingesetzt werden kann.

Inhalt

Hausaufgabe 11: Klausur

- Die Aufgaben dieses Übungsblattes stammen zum großen Teil aus der Zwischenklausur zu „Datenbanken I“ im WS 2003/2004.

  Die ersten drei Aufgaben sind daher eher unterhalb des Niveaus, das bei unserer Klausur vorausgesetzt wird (bei uns gibt es nur eine Klausur am Ende).

- Die Original-Klausur steht auf der Webseite der Vorlesung:

  [https://users.informatik.uni-halle.de/~brass/db24/exams/exam7.pdf]

  Es gibt in dieser Klausur noch eine Reihe interessanter Ankreuzaufgaben.

- Adminer-Link zum Testen der Anfragen:

  [https://dbs.informatik.uni-halle.de/edb-probeklausur-ws1920?pgsql=db&username=student_gast&db=postgres&ns=exam7]

Hausaufgabe 11: Datenbank (1)

- Der Dozent möchte sein Skript in ein Online-Lernsystem umwandeln. Dies soll u.a. erlauben, dass Studierende selbst die Teile des Skripts auswählen, die sie lernen wollen.
- ABSCHNITTE(AB, TITEL, URL, SEITEN)

  Die Tabelle „ABSCHNITTE“ enthält einen Eintrag für jeden Abschnitt des Skripts (Lerneinheit). Jeder Abschnitt hat eine eindeutige Nummer „AB“ (Primärschlüssel). Die Überschrift „TITEL“ des Abschnitts ist Sekundärschlüssel.
- BEGRIFFE(BEG, NAME, OBERBEG$^{\circ}\rightarrow$BEGRIFFE)

  NAME ist Sekundärschlüssel. Die Begriffe sind in eine Hierarchie eingeordnet.
- VORKOMMEN(AB$\rightarrow$ABSCHNITTE, BEG$\rightarrow$BEGRIFFE, ART)

  ART kann die Werte VORAUS, DEF und BSP annehmen.

Hausaufgabe 11: Datenbank (2)

ABSCHNITTE			
AB	TITEL	URL	SEITEN
101	Einleitung	c1_intro/s1.pdf	5
201	Relationales Modell: Basis	c2_relmo/s1.pdf	3
202	Integritätsbedingungen	c2_relmo/s2.pdf	7
203	Relationale Algebra	c2_relmo/s3.pdf	12
301	Einfache Anfragen in SQL	c3_sql_i/s1.pdf	6

Hausaufgabe 11: Datenbank (3)

BEGRIFFE		
BEG	NAME	OBERBEG^o
1000	Datenmodell	(null)
1001	Relationales Modell	1000
1002	Schlüssel	1001
1003	Fremdschlüssel	1001
1004	Datei	(null)
1005	Anfragesprachen	(null)
1006	Relationale Algebra	1005
1007	Projektion	1006

Hausaufgabe 11: Datenbank (4)

VORKOMMEN		
AB	BEG	ART
101	1004	VORAUS
101	1000	DEF
101	1005	DEF
201	1000	VORAUS
202	1002	DEF
202	1002	BSP
202	1003	DEF
202	1007	VORAUS
203	1007	BSP
203	1007	DEF

Hausaufgabe 11.1: NOT EXISTS/NOT IN (1)

- Geben Sie alle Begriffe aus (jeweils Nummer und Name), die irgendwo im Skript vorausgesetzt werden (Art des Vorkommens „VORAUS“), aber nirgendwo definiert werden (Art „DEF“).
- Das erwartete Ergebnis ist:

BEG	NAME
1004	Datei

Die Groß-/Kleinschreibung der Ausgabespalten ist nicht vorgeschrieben. Die Klausur stammt noch aus einer Zeit, in der Oracle in der Vorlesung benutzt wurde. Dort ist Großschreibung normal.

Hausaufgabe 11.1: NOT EXISTS/NOT IN (2)

Lösung:

- Anfrage mit symmetrischer Behandlung von Voraussetzung und Definition:

```
SELECT b.BEG, b.NAME
FROM   BEGRIFFE b
WHERE  b.BEG IN (SELECT v.BEG
                 FROM   VORKOMMEN v
                 WHERE  v.ART = 'VORAUS')
AND    b.beg NOT IN (SELECT v.BEG
                     FROM VORKOMMEN v
                     WHERE v.ART = 'DEF')
```

Die Spalte `BEG` ist Teil des Primärschlüssels von `VORAUS`. Daher können die Unteranfragen keinen Nullwert liefern (das wäre ja für `NOT IN` ein Problem).

Hausaufgabe 11.1: NOT EXISTS/NOT IN (3)

Lösung, Forts.:

- Anfrage mit Join in Hauptanfrage für positive Teilbedingung:

```
SELECT DISTINCT b.BEG, b.NAME
FROM   BEGRIFFE b, VORKOMMEN v
WHERE  b.BEG = v.BEG
AND    v.ART = 'VORAUS'
AND    NOT EXISTS (SELECT *
                   FROM   VORKOMMEN d
                   WHERE  d.BEG = b.BEG
                   AND    d.ART = 'DEF')
```

`DISTINCT` ist nötig, weil ein Begriff mehrfach vorausgesetzt werden könnte.
In den Beispieldaten kommt das nicht vor, aber die Anfrage soll bei beliebigen Zuständen (die dem Schema entsprechen) keine Duplikate liefern.

Hausaufgabe 11.2: LIKE (1)

- In welchen Titeln von Abschnitten kommen sowohl die Teilzeichenkette „SQL“ als auch die Teilzeichenkette „Anfragen“ vor?
- Es ist nicht bekannt, ob zuerst „SQL“ und dann „Anfragen“ kommt, oder umgekehrt.
- Geben Sie jeweils die Abschnittsnummer, den Titel und die URL aus.
- Das erwartete Ergebnis ist:

AB	TITEL	URL
301	Einfache Anfragen in SQL	c3_sql_i/s1.pdf

Hausaufgabe 11.2: LIKE (2)

Lösung:

- Da die Reihenfolge nicht bekannt ist, testet man beide Teilzeichenketten einzeln:

```
SELECT AB, TITEL, URL
FROM   ABSCHNITTE
WHERE  TITEL LIKE '%SQL%'
AND    TITEL LIKE '%Anfragen%'
```

- Dies wäre auch möglich, ist aber länger:

```
SELECT AB, TITEL, URL
FROM   ABSCHNITTE
WHERE  TITEL LIKE '%SQL%Anfragen%'
OR     TITEL LIKE '%Anfragen%SQL%'
```

Hausaufgabe 11.3: IS NULL (1)

- Geben Sie Nummer und Titel aller Abschnitte aus, die mindestens einen Begriff definieren, der in der Begriffshierarchie ganz oben steht, für den also „OBERBEG“ einen Nullwert enthält.
- Im Beispiel würde Abschnitt 101 die Begriffe 1000 (Datenmodell) und 1005 (Anfragesprachen) definieren.
- Denken Sie wie bei allen Anfragen darüber nach, ob eine explizite Duplikateliminierung notwendig ist.
- Das erwartete Ergebnis ist:

AB	TITEL
101	Einleitung

Hausaufgabe 11.3: IS NULL (2)

Lösung:

- Lösung mit Join aller drei Tabellen und `IS NULL`:

```
SELECT DISTINCT a.AB, a.TITEL
FROM   BEGRIFFE b, VORKOMMEN v, ABSCHNITTE a
WHERE  b.BEG = v.BEG
AND    v.AB = a.AB
AND    b.OBERBEG IS NULL
AND    v.ART = 'DEF'
```

`DISTINCT` is nötig, weil ein Abschnitt auch mehrere Begriffe definieren könnte, die in der Begriffshierarchie ganz oben stehen.

Hausaufgabe 11.3: IS NULL (3)

Lösung, Forts.:

- Man würde kein `DISTINCT` benötigen, wenn man den Teil, der für die Duplikate verantwortlich ist, in eine Unteranfrage verlegt:

```
SELECT a.AB, a.TITEL
FROM   ABSCHNITTE a
WHERE  EXISTS(SELECT *
              FROM   BEGRIFFE b, VORKOMMEN v
              WHERE  b.BEG = v.BEG
              AND    v.AB = a.AB
              AND    b.OBERBEG IS NULL
              AND    v.ART = 'DEF')
```

Da ein Schlüssel von der Tupelvariable in der Hauptanfrage ausgegeben wird, erzeugt diese Anfrage beweisbar keine Duplikate.

Hausaufgabe 11.4: Selbstverbund (1)

- Es ist schlecht, wenn Begriffe definiert werden, nachdem Sie bereits vorausgesetzt wurden.
- Schreiben Sie eine Anfrage, die Abschnitte A und B findet, so dass B eine größere Nummer (`AB`) als A hat, aber in B ein Begriff definiert wird, der in A vorausgesetzt wird.
- Geben Sie den Namen des Begriffs und die Nummern der beiden Abschnitte A und B aus.
- Die Ausgabe-Spalte mit der Abschnittsnummer von A soll „`VORAUSGESETZT_IN`“ heißen, und die mit der Abschnittsnummer von B soll „`DEFINIERT_IN`“ heißen.

Hausaufgabe 11.4: Selbstverbund (2)

- Das erwartete Ergebnis ist:

NAME	VORAUSGESETZT_IN	DEFINIERT_IN
Projektion	202	203

- Bei dieser Aufgabe sind die Spaltennamen vorgeschrieben, inklusive der Großschreibung.

  In der echten Klausur ist üblicherweise der Default, dass die Spaltennamen genau so heißen müssen, wie im Beispiel gezeigt (inklusive der Groß- bzw. Kleinschreibung).
  Da neuere Klausuren mit PostgreSQL entwickelt sind, werden die meisten Spaltennamen dort aber klein geschrieben sein, so dass oft keine Umbenennung nötig ist.

Hausaufgabe 11.4: Selbstverbund (3)

Lösung:

- Man kann die Bezeichner A und B aus der Aufgabenstellung direkt als Tupelvariablen verwenden:

```
SELECT X.NAME, A.AB AS "VORAUSGESETZT_IN",
       B.AB AS "DEFINIERT_IN"
FROM   VORKOMMEN A, VORKOMMEN B, BEGRIFFE X
WHERE  B.AB > A.AB
AND    B.ART = 'DEF'
AND    A.ART = 'VORAUS'
AND    X.BEG = A.BEG
AND    X.BEG = B.BEG
```

Da `B` schon verbraucht war, wurde für `BEGRIFFE` leider ein etwas ungünstiger Tupelvariablen-Name gewählt.

Hausaufgabe 11.5: GROUP BY (1)

- Die Abschnitte in der Tabelle `AB` sind durch Aufteilung von größeren Kapiteln entstanden.
- Die Kapitelnummer ist die Abschnittsnummer `AB` ohne die letzten zwei Dezimalstellen.

  Man kann Sie berechnen, indem man die Abschnitts-Nummer `AB` durch 100 teilt (mit Integer-Division, d.h. Abrundung — glücklicherweise ist „`AB`“ als `INTEGER` deklariert).
- Geben Sie für jedes Kapitel Folgendes aus:
  - die Kapitel-Nummer,
  - die Anzahl der Abschnitte des Kapitels und
  - die Gesamtanzahl der Seiten (d.h. Summe der einzelnen Seitenzahlen)

Hausaufgabe 11.5: GROUP BY (2)

- **Sortieren** Sie die Ausgabe nach der Kapitelnummer.
- Bitte wählen Sie die Spaltenüberschriften wie in der Beispiel-Ausgabe gezeigt (auch mit der vorgegebenen Groß-/Kleinschreibung).
- Wenn Sie wollen, können Sie noch freiwillig das Verzeichnis des Kapitels aus der `URL` ausgeben.

  In der Klausur würde das wohl einen Bonuspunkt geben, bei der Hausaufgabe zur administrativen Vereinfachung leider nicht. Sie können voraussetzen, dass jede `URL` einen „/“ enthält, und dass der Präfix der `URL` bis zum Schrägstrich für alle Abschnitte eines Kapitels gleich ist.
  Zur Erinnerung:
  `POSITION(`s_1 `IN` s_2`)`: Position der Zeichenkette s_1 als Teilzeichenkette von s_2.
  `SUBSTRING(`s `FROM` n `FOR` m`)`: Teilzeichenkette von s ab Position n mit einer Länge von m Zeichen.

Hausaufgabe 11.5: GROUP BY (3)

- Die erwartete Antwort ist:

Kapitel	Abschnitte	Seiten	Verzeichnis
1	1	5	c1_intro
2	3	22	c2_relmo
3	1	6	c3_sql_i

- Dabei ist die letzte Spalte „`Verzeichnis`“ optional.

Hausaufgabe 11.5: GROUP BY (4)

Lösung:

- Anfrage mit dem optionalen Teil:

```
SELECT AB/100 AS "Kapitel",
       COUNT(*) AS "Abschnitte",
       SUM(SEITEN) AS "Seiten",
       MIN(SUBSTRING(URL FROM 1
                     FOR POSITION('/' IN URL)-1))
           AS "Verzeichnis"
FROM   ABSCHNITTE
GROUP  BY AB/100
ORDER  BY "Kapitel"
```

Es spielt keine Rolle, ob man `MIN` oder `MAX` verwendet. Die Verzeichnisse sind nach der Aufgabenstellung alle gleich. Alternativ könnte man danach gruppieren.

Hausaufgabe 11.6: Selbstverbund, HAVING (1)

- Geben Sie zu jedem vorkommenden Oberbegriff die Anzahl seiner Unterbegriffe an, allerdings nur, wenn er mindestens zwei Unterbegriffe hat.
- Drucken Sie die Nummer und den Namen des Oberbegriffs, die Anzahl seiner Unterbegriffe und den alphabetisch ersten Unterbegriff (als ein Beispiel eines Unterbegriffs).
- Nennen Sie die Ausgabespalten wie im Beispiel gezeigt.
- Die erwartete Antwort ist:

Nr	Name	UB	Beispiel
1001	Relationales Modell	2	Fremdschlüssel

Die dritte Spalte heißt eigentlich „`Unterbegriffe`“.

Hausaufgabe 11.6: Selbstverbund, HAVING (2)

Lösung:

- Man braucht einen Selbstverbund.
  Die Forderung „mindestens zwei Unterbegriffe“ läßt sich am einfachsten mit `HAVING` umsetzen:

```
SELECT OBER.BEG AS "Nr",
       OBER.NAME AS "Name",
       COUNT(*) AS "Unterbegriffe",
       MIN(UNTER.NAME) AS "Beispiel"
FROM   BEGRIFFE OBER, BEGRIFFE UNTER
WHERE  UNTER.OBERBEG = OBER.BEG
GROUP  BY OBER.BEG, OBER.NAME
HAVING COUNT(*) >= 2
```

Inhalt

Präsenzaufgabe: Vereinigung vs. Verbund

- Zwei alternative Darstellungen der Punkte für HA und Zwischen- und Endklausur der Studenten sind:

Resultate_1

STUDENT	H	Z	E
Jim Ford	95	60	75
Ann Lloyd	80	90	95

Resultate_2

STUDENT	ATYP	PROZENT
Jim Ford	H	95
Jim Ford	Z	60
Jim Ford	E	75
Ann Lloyd	H	80
Ann Lloyd	Z	90
Ann Lloyd	E	95

- Für die Umrechnung von `Resultate_1` in `Resultate_2` braucht man: ☐ Vereinigung ☐ Verbund

  D.h. eine Anfrage, die nur `Resultate_1` verwendet, und `Resultate_2` liefert.