Einführung in Datenbanken

Übung 4: Datentypen in SQL

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

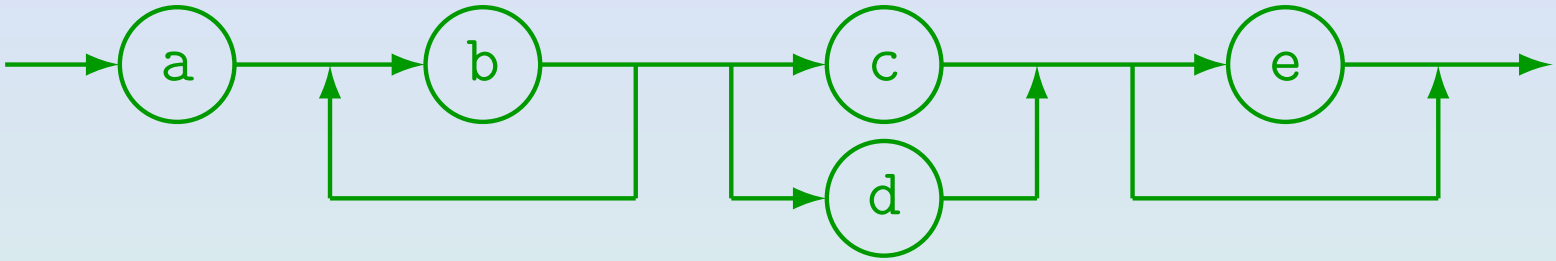
1. Präsenzaufgabe 3
2. Hausaufgabe 3
3. Datentypen
4. LIKE
5. Funktionen
6. Hausaufgabe 2
7. Präsenzaufgabe

Präsenzaufgabe: Syntaxgraph (1)

- Zeichnen Sie ein Syntaxdiagramm in der Notation der Vorlesung für Zeichenketten der folgenden Form:
  - Am Anfang steht immer der Buchstabe „a“.
  - Dann kommen beliebig viele „b“ (aber mindestens eins).
  - Anschließend entweder ein „c“ oder ein „d“.
  - Am Ende optional ein „e“ (d.h. das „e“ darf auch fehlen).
- Das Syntaxdiagramm soll die syntaktische Kategorie „Test“ beschreiben. Ein mögliches Wort der definierten Sprache wäre „abbbce“.

Präsenzaufgabe: Syntaxgraph (2)

- Test:



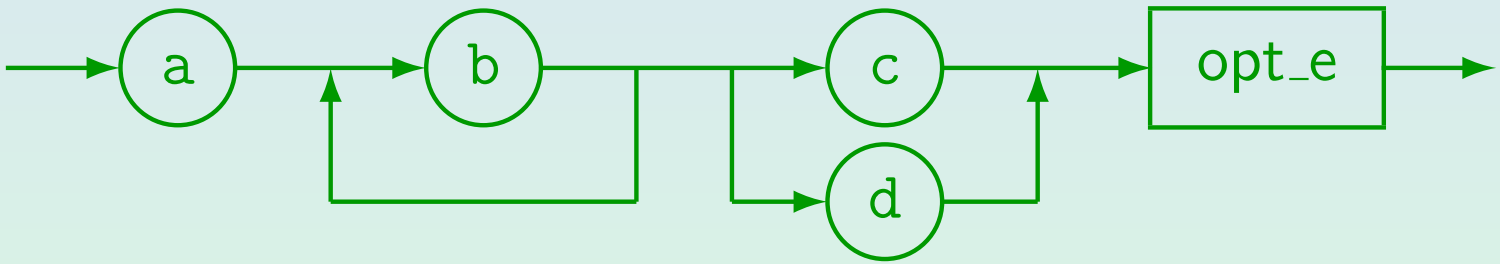
- Häufige Fehler:
  - Dieser Syntaxgraph enthält nur Terminalsymbole.
    Diese werden hier mit Ovalen/Kreisen notiert.

    Kästen/Rechtecke werden dagegen für Nichtterminalsymbole verwendet

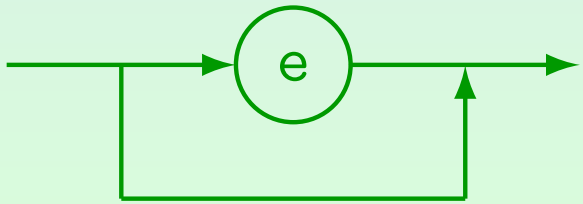
    (In der Literatur teils anders: Man muss es festlegen.).
  - Der Graph sollte genau einen Eingang und genau einen Ausgang haben. Ovale und Kästen ebenfalls.
  - Name des Graphen ist wichtig (was wird definiert?).

Präsenzaufgabe: Syntaxgraph (3)

- Es wäre auch möglich, Teile in einen extra Syntaxgraphen auszulagern, also zusätzliche „Syntaktische Kategorien“ (Nichtterminalsymbole) einzuführen.
- Test:



- opt_e:



Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

Vorstellung des 3. Übungsblattes: Aufgabe 1 (1)

- Lesen Sie das Kapitel 5: „SQL: Datentypen“ der Folien zu Ende. Notieren Sie sich ggf. Fragen und stellen Sie diese in der nächsten Übung (oder der Vorlesung).
- Schauen Sie sich die folgenden fünf Wertausdrücke an und überlegen Sie sich, was das Ergebnis sein sollte.

a)
```
SELECT ROUND(1e9*CAST(0.10 AS REAL))
```

b)
```
SELECT SIN(0.5 * PI())
```

c)
```
SELECT LOG(100)
```

d)
```
SELECT UPPER(SUBSTRING('abcdef' FROM 2 FOR 3))
       || '.'
```

e)
```
SELECT POSITION('de' IN TRIM(' abcdef'))
```

Vorstellung des 3. Übungsblattes: Aufgabe 1 (2)

- Anschließend testen Sie das Ergebnis von PostgreSQL im Adminer.
- Geben Sie eine `.txt`-Datei ab mit den fünf Werten und jeweils einer kurzen Erklärung.

  1–2 Zeilen pro Teilaufgabe sollten reichen.

EMP-DEPT-Datenbank (1)

- Klassische Beispiel-Datenbank von Oracle.
- Schema „empdept_public“ im Adminer:
  - dept(deptno, dname, loc)
  - emp(empno, ename, job, mgr°→emp, hiredate, sal, comm°, deptno°→dept)
- dept: „Department“ (Abteilungen einer Firma)
- emp: „Employee“ (Angestellte der Firma)
- Der Link zum Adminer ist:

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

EMP-DEPT-Datenbank (2)

- Die Tabelle dept hat die Spalten
  - deptno: Abteilungsnummer (Department Number),
  - dname: Name der Abteilung (Department Name),
  - loc: Ort/Sitz der Abteilung (Location):

DEPT		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMP-DEPT-Datenbank (3)

- Die Tabelle `emp` hat folgende Spalten:
  - `empno`: Angestellten-Nummer (identifiziert den Angestellten)
  - `ename`: Name des Angestellten.
  - `job`: Berufsbezeichnung des Angestellten.
  - `mgr`: Angestellten-Nummer des direkten Vorgesetzten.

    `mgr` von „manager“.
  - `hiredate`: Datum der Einstellung.
  - `sal`: Gehalt des Angestellten („salary“).
  - `comm`: Provision (nur für Verkäufer) („commission“).
  - `deptno`: Abteilung des Angestellten.

EMP-DEPT-Datenbank (4)

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

Vorstellung des 3. Übungsblattes: Aufgabe 2 (1)

- Angenommen, Sie wissen, dass das durchschnittliche Gehalt in der Firma 2073.21 € ist.

  Später lernen wir, wie man das mit SQL berechnen kann.

- Geben Sie für jeden Angestellten
  - seinen Angestellten-Nummer (`empno`),
  - seinen Namen (`ename`),
  - sein Gehalt (`sal`) und
  - die prozentuale Abweichung seines Gehalts bezogen auf den Durchschnittswert `2073.21` aus. Negative Werte stehen für ein unterdurchschnittliches Gehalt, positive Werte für ein überdurchschnittliches Gehalt. Runden Sie den Prozentsatz bitte auf eine ganze Zahl.
- Erwartetes Ergebnis: Siehe nächste Folie.

Vorstellung des 3. Übungsblattes: Aufgabe 2 (2)

empno	ename	sal	Abweichung in %
7369	SMITH	800	-61
7499	ALLEN	1600	-23
7521	WARD	1250	-40
7566	JONES	2975	43
7654	MARTIN	1250	-40
7698	BLAKE	2850	37
7782	CLARK	2450	18
7788	SCOTT	3000	45
7839	KING	5000	141
7844	TURNER	1500	-28
7876	ADAMS	1100	-47
7900	JAMES	950	-54
7902	FORD	3000	45
7934	MILLER	1300	-37

Vorstellung des 3. Übungsblattes: Aufgabe 2 (3)

Hinweise:

- Da wir das noch nicht besprochen haben, sind keine bestimmten Spaltennamen verlangt.
- Ebenso keine spezielle Sortierung der Zeilen.
- Bei der Klausur sind dagegen meistens genau die Spaltennamen verlangt, die auch im erwarteten Ergebnis stehen (einschließlich Groß-/Kleinschreibung).
- Oft ist auch eine bestimmte Sortierung der Zeilen verlangt.

  Es gibt dann einen Punktabzug, wenn die `ORDER BY`-Klausel fehlt, selbst wenn das DBMS zufällig die Zeilen in der gewünschten Reihenfolge ausgeben sollte.

Vorstellung des 3. Übungsblattes: Aufgabe 3 (1)

- Das Management hat festgestellt, dass für den Monat der Einstellung (`hiredate`) immer ein volles Gehalt gezahlt worden ist.
- Es möchte eine Aufstellung mit dem korrigierten Einstellungsdatum (der erste Tag des jeweiligen Monats) und den fehlenden Arbeitstagen in diesem Monat.

  Wenn jemand z.B. am 5. des Monats eingestellt wurde, fehlen 4 Arbeitstage.
- Drucken Sie Angestelltennummer und Angestelltennamen, das tatsächliche Einstellungsdatum, das korrigierte Einstellungsdatum und die fehlenden Tage.
- Die Daten sollen im Format `DD.MM.YYYY` gedruckt werden (wie es in Deutschland üblich ist).

Vorstellung des 3. Übungsblattes: Aufgabe 3 (2)

empno	ename	hiredate	korrigiert	Minus-Tage
7369	SMITH	17.12.1980	01.12.1980	16
7499	ALLEN	20.02.1981	01.02.1981	19
7521	WARD	22.02.1981	01.02.1981	21
7566	JONES	02.04.1981	01.04.1981	1
7654	MARTIN	28.09.1981	01.09.1981	27
7698	BLAKE	01.05.1981	01.05.1981	0
7782	CLARK	09.06.1981	01.06.1981	8
7788	SCOTT	09.12.1982	01.12.1982	8
7839	KING	17.11.1981	01.11.1981	16
7844	TURNER	08.09.1981	01.09.1981	7
7876	ADAMS	12.01.1983	01.01.1983	11
7900	JAMES	03.12.1981	01.12.1981	2
7902	FORD	03.12.1981	01.12.1981	2
7934	MILLER	23.01.1982	01.01.1982	22

Vorstellung des 3. Übungsblattes: Aufgabe 3 (3)

Hinweise:

- Bei Bedarf können Sie `CAST(x AS INTEGER)` verwenden, um einen `NUMERIC`-Wert x in den Datentyp `INTEGER` zu konvertieren.
  - Sie können nur ganze Zahlen auf Datumswerte addieren oder davon subtrahieren.
  - `EXTRACT(...)` liefert aber den Typ `NUMERIC`.
  - Den Typ eines Ausdrucks e können Sie in PostgreSQL mit `pg_typeof(e)` bestimmen.
- Die Datentyp-Funktionen von PostgreSQL finden Sie in Kapitel 9 des Handbuchs:
  [https://www.postgresql.org/docs/current/functions.html]

Die Aufgabe lässt sich mit den im Skript vorgestellten Funktionen lösen.

Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

Datentypen

- Kann man `'ab'` in eine Spalte vom Typ `CHAR(3)` einfügen?
  A. Nein, das gibt eine Fehlermeldung.
  B. Es geht, `CHAR(3)` bedeutet „maximal 3 Zeichen“.
  C. Die Zeichenkette wird rechts mit einem Leerzeichen aufgefüllt.
- Ein Kollege verwendet den Datentyp `CHAR(30)` für Nachnamen. Was sagen Sie dazu?

  `Leutheusser-Schnarrenberger` (Bundesjustizministerin a.D.) hat 27 Zeichen.
- Ist `123.4` ein legaler Wert von `NUMERIC(3,1)`?

Tests mit PostgeSQL

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

  Die Zugangsdaten unserer Installation stehen in StudIP, Reiter „Adminer“.

- Was liefern folgende Anfragen?
  - `select 1/0;`
  - `select 3/2;`
  - `select pg_typeof(2);`
  - `select 3/cast(2 as numeric);`

    `CAST(`E `AS` T`)` liefert den Wert des Ausdrucks E umgewandelt nach T.
  - `select cast(1000 as numeric(2));`

Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

LIKE zum Mustervergleich (1)

- Was sind die Ergebnisse der folgenden Anfrage?

```
SELECT ID, S
FROM   TEST
WHERE  S LIKE 'A%B_'
```

- Die Tabelle ist:

TEST	
ID	S
1	A%B_
2	ABC
3	ABBBC
4	AXYZB
5	ACB␣␣

Ja/Nein-Abstimmung pro Zeile. „␣“ symbolisiert ein Leerzeichen.

LIKE zum Mustervergleich (2)

Lösung:

- Das Ergebnis ist:

ID	S
1	A%B_
2	ABC
3	ABBBC

- Bei PostgreSQL kann man eine Tabelle auch direkt in die Anfrage schreiben (wenn man keine Tabelle anlegen kann):

```
SELECT ID, S
FROM   (VALUES(1, 'A%B_'),
               (2, 'ABC'),
               ...) AS TEST(ID,S)
WHERE  S LIKE 'A%B_'
```

Warnung!

- Es gibt offenbar Tutorials, die `LIKE` grundsätzlich zum String-Vergleich verwenden, auch wenn rechts gar kein Musterzeichen `%` oder `_` verwendet wird.
- Glauben Sie nicht alles, was Sie im Internet lesen.
- Ich ziehe Ihnen einen Punkt ab, wenn Sie `LIKE` verwenden, wo ein einfaches = auch gehen würde.
- Wenn Sie darüber diskutieren wollen, bitte.

  Ich glaube, dass = sehr viel klarer ist, wenn man eine exakte Gleichheit meint (bis eventuell auf Leerzeichen am Ende). Wenn Sie `LIKE` verwenden, muss man rechts nach Musterzeichen suchen. Bei = haben % und _ dagegen keine besondere Bedeutung. Falls die rechte Seite auch ein Spaltenwert ist, wird es besonders schwierig, denn Sie können oft nicht wissen, ob er vielleicht % oder _ enthalten könnte.

Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

String-Funktionen

- s_1 `||` s_2: Konkatenation zweier Zeichenketten.
- `CHARACTER_LENGTH(`s`)`, `CHAR_LENGTH(`s`)`:
  Länge einer Zeichenkette in Zeichen.
- `LOWER(`s`)`: Zeichenkette in Kleinbuchstaben umgewandelt.
- `UPPER(`s`)`: Zeichenkette in Großbuchstaben umgewandelt.
- `POSITION(`s_1 `IN` s_2`)`:
  Position des ersten Vorkommens von s_1 in s_2.
- `SUBSTRING(`s `FROM` p `FOR` n`)`:
  Teilzeichenkette der Länge n von s, die an Position p beginnt.

  `SUBSTRING(`s `FROM` p`)`: Ganzer Rest von s ab Position p.
- `TRIM(`s`)`: Teilzeichenkette von s, ohne Leerzeichen am Anfang oder Ende.

Numerische Funktionen (1)

- `ABS(x)`: Absolutwert von x.
- `MOD(n,m)`: Rest bei Division von n durch m.
- `LN(x)`: Natürlicher Logarithmus: $\ln(x)$.
- `EXP(x)`: Exponentialfunktion: e^x.
- `POWER(x,y)`: Potenz: x^y.
- `SQRT(x)`: Quadratwurzel: $\sqrt{x}$.
- `FLOOR(x)`: Abrundung: $\lfloor x \rfloor$.
- `CEILING(x)`, `CEIL(x)`: Aufrundung: $\lceil x \rceil$.
- `ROUND(x)`, `ROUND(x,n)`: Rundung (auf n Stellen)

Numerische Funktionen (2)

- `WIDTH_BUCKET(x,y,z,n)`: In welchem von n Teilintervallen des Intervalls $[y, z]$ liegt x?
- `SIN(x)`, `COS(x)`, `TAN(x)`, `SINH(x)`, `COSH(x)`, `TANH(x)`, `ASIN(x)`, `ACOS(x)`, `ATAN(x)`: Trigonometrische Funktionen.

  Winkel werden in Bogenmaß (`rad`) gemessen: $180° = \pi = 3.14159$.

  PostgreSQL, MariaDB/MySQL und MS SQL Server kennen nicht `SINH`, `COSH`, `TANH`.
- `LOG(b,x)`: Logarithmus zur Basis b: $\log_b(x)$.
- `LOG10(x)`: Logarithmus zur Basis 10: $\log_{10}(x)$.

  PostgreSQL und Oracle kennen nicht `LOG10`.

  Bei PostgreSQL liefert `LOG(x)` den Logarithmus zur Basis 10.

Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

Datenbank: Komponisten

- Die Aufgaben dieses Übungsblattes beziehen sich auf die Komponisten-Tabelle in einer Datenbank mit Informationen über Musik-CDs (klassische Musik).

KOMPONIST				
KNr	Name	Vorname	geboren	gestorben
11	Händel	Georg Friedrich	1685	1759
⋮	⋮	⋮	⋮	⋮

- Sie finden Sie im Adminer, Schema „komponist_public“:

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

Übungsblatt 2, Aufgabe 1 (1)

- Schreiben Sie eine SQL-Anfrage, die den Vornamen des Komponisten „Vivaldi“ bestimmt.
- Versuchen Sie, sich dabei an die eingeschränkte Syntax aus Kapitel 3 der Vorlesung zu halten.
- Das erwartete Ergebnis ist:

vorname
Antonio

- Schreiben Sie Ihre Bereitschaft, vorzurechnen, in einen Kommentar (ebenso eventuelle weitere Angaben zu Quellen und Arbeitszeit).

  Da die Fähigkeit, Kommentare in SQL zu schreiben, wichtig ist, wird ein Punkt abgezogen, wenn die Angabe fehlt. Dies gilt auch für die anderen Aufgaben.

Übungsblatt 2, Aufgabe 1 (2)

- Mögliche Lösung:

```
SELECT Vorname
FROM   Komponist
WHERE  Name = 'Vivaldi'
```

- Die Groß-/Kleinschreibung von Schlüsselworten wie „SELECT“ und Tabellen- und Spaltennamen ist egal:

```
select vorname
from   komponist
where  name = 'Vivaldi'
```

- Bei String-Konstanten ist die Groß-/Kleinschreibung dagegen wichtig. Mit `'vivaldi'` wäre das Ergebnis leer.

Übungsblatt 2, Aufgabe 1 (3)

- RTF, PDF, oder Word als Abgabeformat ist falsch.
- SQL ist eine Programmiersprache wie Java.

  Natürlich ist SQL speziell für Datenbanken gemacht, und deklarativ im Gegensatz zu Java, was eher imperativ ist. Aber wenn es um die Codierung von „Programmen“ geht, sind beide im wesentlichen Folgen von ASCII-Zeichen (oder UTF-8, wenn Sie auch Umlaute verwenden wollen). Es gibt z.B. keinen Fettdruck in SQL-Anfragen oder die Möglichkeit, einen Teil in einer größeren Schrift anzuzeigen. Ein Editor, der die SQL-Syntax kennt, wird automatisch Schlüsselworte wie „`SELECT`“ in einer anderen Farbe anzeigen als z.B. die Tabellennamen. In der Datei gibt es die Angaben zu Farben aber nicht. Sie können die Farbe auch nicht einzeln für jedes Wort wählen.
- Verwenden Sie einen Editor, wie Sie ihn auch für Java verwenden würden (z.B. Notepad++, Visual Studio Code).

Übungsblatt 2, Aufgabe 1 (4)

- Sie brauchen Quellen wie W3Schools nur dann anzugeben, wenn die Anfragen nichttrivial waren und Ihnen die Quelle für die konkrete Hausaufgabe mehr genutzt hat als ein allgemeines Datenbank-Lehrbuch (oder mein Skript).

  Dass Sie mein Skript verwenden, weiß ich ja (oder hoffe ich).

- Bei einer komplexen Anfrage gibt es eine ganze Reihe von Möglichkeiten, sie zu formulieren. Wenn dann zwei Studierende von der gleichen Anfrage ausgegangen sind, und nur Tabellen- und Spaltennamen ausgetauscht haben, könnte vielleicht der Eindruck eines Plagiats entstehen.

- Bei dieser Aufgabe gibt es aber im wesentlichen nur eine einzige Lösung.

  Bis auf Groß-/Kleinschreibung und Zeilenaufteilung/Formatierung (man kann z.B. alles in eine Zeile schreiben). Leerzeichen links/rechts von „=“: optional.

Übungsblatt 2, Aufgabe 1 (5)

- Verwenden Sie nur Daten, die in der Aufgabenstellung genannt werden.

  Man muss natürlich wissen, dass Vivaldi Nachname eines Komponisten ist.

- Es ist z.B. nicht ok, auch die Geburtsdaten in der Anfrage zu verwenden:

```
SELECT Vorname
From   komponist
WHERE  name = 'Vivaldi'
AND    geboren = '1678' AND gestorben = '1741'
```

- Außerdem ist falsch, dass die Geburtsdaten hier als Strings angegeben sind, nicht als Zahlen.

  Die meisten Datenbanksysteme sind sehr großzügig mit Typ-Umwandlungen, so dass es keine Fehlermeldung gibt. Wir ziehen aber dennoch Punkte für einen Typfehler ab. Die Spalten `geboren` und `gestorben` sind numerisch.

Übungsblatt 2, Aufgabe 1 (6)

- Es ist insbesondere falsch, das erwartete Ergebnis direkt in die Anfrage zu schreiben:

```
SELECT vorname
FROM   komponist
WHERE  vorname='Antonio'
```

- Auch folgende Anfrage würde ja das richtige Ergebnis liefern, wäre aber trotzdem falsch:

```
SELECT 'Antonio'
```

- Anfragen müssen in allen möglichen DB-Zuständen das richtige Ergebnis liefern.

  Natürlich wird sich der Vorname von Vivaldi nicht ändern, aber falls in der DB ein falscher Vorname eingetragen wäre, müsste dieser geliefert werden. Wenn man den Vornamen schon weiß, braucht man keine DB-Anfrage.

Übungsblatt 2, Aufgabe 1 (7)

- Es war verlangt, Anfragen auszuprobieren.
- Die folgende Anfrage bekommt voraussichtlich 0 Punkte:

```
select vorname
from   komponist
where  nachname = 'vivaldi'
```

- Diese Anfrage liefert eine Fehlermeldung:

  Error in query (7):
  ERROR: column "nachname" does not exist
  LINE 3: where nachname = 'vivaldi'

  „(7)“ ist vermutlich ein Fehlercode. Eventuelle Hinweise wären willkommen.

- Korrigiert man den Fehler, bekommt man ein leeres Ergebnis wegen der Kleinschreibung von `'vivaldi'`.

  Dafür würde es nur einen Punkt Abzug geben.

Übungsblatt 2, Aufgabe 1 (8)

- Die folgende Anfrage bekommt auch 0 Punkte:

```
SELECT vorname
FROM   komponisten
WHERE  name = 'Vivaldi'
```

- Diese Anfrage liefert:

```
Error in query (7):
ERROR: relation "komponisten" does not exist
LINE 2: FROM komponisten
```

- Die Tabelle heißt „`Komponist`“ (Singular).
- Probieren Sie Ihre Anfragen im Adminer aus und verwenden Sie Copy&Paste (von dort in die `.sql`-Datei oder von einer `.sql`-Datei in den Adminer).

Übungsblatt 2, Aufgabe 1 (9)

Warum so streng?

- Es ist für die Tutoren viel leichter, die volle Punktzahl zu geben, statt die Anfrage zu debuggen und Fehler zu erklären.

  Das Übungsblatt hatte 5 Aufgaben mit ca. 70 Abgaben pro Aufgabe. Außerdem ist eine Präsenzübung mit ca. 50 Abgaben zu korrigieren, die wegen der Papierabgaben aufwändiger ist (manuelle Erfassung der Namen). Wir haben drei Tutoren. Die Tutoren werden für 25 Stunden im Monat bezahlt, wovon wöchentliche Besprechungen von jeweils ca. einer Stunde abgehen. Es bleiben also ca. 5 Stunden pro Tutor pro Woche. Wenn man zur einfachereren Rechnung einen Tutor ganz für die Präsenzaufgabe reserviert, bleiben 10 Stunden, d.h. 600 Minuten, für 350 Lösungen. Insgesamt also ca. 2 Minuten pro Lösung.

- Oft verdecken Syntaxfehler nur weitere Fehler.
- In der Klausur müssen Sie mit Syntaxfehlern umgehen können.

Übungsblatt 2, Aufgabe 1 (10)

- Datenbank-Spalten sollten nicht in Anführungszeichen `"..."` geschrieben werden, wenn die `CREATE TABLE`-Anweisungen nicht auch diese Anführungszeichen enthalten:

```
SELECT "vorname"
FROM   komponist
WHERE  vorname = 'Antonio'
```

- Die Anführungszeichen erzwingen eine bestimmte Groß-/Kleinschreibung. Wozu?

  Das funktioniert zufällig in PostgreSQL (weil dort nicht in Anführungszeichen eingeschlossene Bezeichner in Kleinbuchstaben umgewandelt werden), aber z.B. nicht in Oracle (dort würden Großbuchstaben funktionieren).

- Die inkonsistente Schreibweise der gleichen Spalte ist auch stilistisch schlecht.

Übungsblatt 2, Aufgabe 1 (11)

- Bitte schreiben Sie keine Namen in die Anfragen!
- Wir verwenden die Daten für ein Forschungsprojekt (natürlich ohne die Beziehung zu den realen Namen aus der Übungsplattform).
- Wenn Sie Ihren Namen als Kommentar in die Abgabe schreiben, kann es passieren, das Ihr Klartext-Name in dem Datensatz landet, der am Ende auf einer Forschungs-Webseite veröffentlicht wird.

  Forschungsergebnisse sollen ja reproduzierbar sein.
- Wir werden also offenbar nach den Namen aller Studierenden als Substrings in allen SQL-Anfragen suchen müssen (mit SQL `LIKE`).

Übungsblatt 2, Aufgabe 1 (12)

- Halten Sie sich bitte an die vorgegebene Codierung der Daten.
- Folgende Varianten werden voraussichtlich verstanden (die erste ist, was gewünscht war):
  - `VORRECHNEN:5`
  - `VORRECHNEN: 5`
  - `Vorrechnen:5`
  - `Vorrechnen: 5`
- Was soll „`LZ:2`“ bedeuten?
- Bitte „`-- ChatGPT:2`“ statt

  `Codierung: Quelle: ChatGPT4o (modifiziert)`
- Schreiben Sie es als Kommentar! Ernstes Risiko: 0 Punkte.

Übungsblatt 2, Aufgabe 2 (1)

- Schreiben Sie eine SQL-Anfrage, die die Lebensdaten von Wolfgang Amadeus Mozart liefert.
- Dabei ist „Wolfgang Amadeus“ der Vorname und „Mozart“ der Nachname (Spalte „`Name`“).
- Das erwartete Ergebnis ist:

geboren	gestorben
1756	1791

Übungsblatt 2, Aufgabe 2 (2)

- Mögliche Lösung:

```
SELECT geboren, gestorben
FROM   Komponist
WHERE  Vorname = 'Wolfgang Amadeus'
AND    Name = 'Mozart'
```

Übungsblatt 2, Aufgabe 3 (1)

- Geben Sie alle Komponisten aus, die zwischen 1500 und 1599 geboren wurden (jeweils einschließlich der Grenzen). Drucken Sie alle Spalten der Tabelle.
- Das erwartete Ergebnis ist (die Sortierung könnte anders sein):

knr	name	vorname	geboren	gestorben
13	Monteverdi	Claudio	1567	1643
19	Byrd	William	1543	1623
42	Franck	Melchior	1580	1639
44	Mainerio	Giorgio	1545	1582
47	Da Nola	Giovan ...	1510	1592
48	Azzaiolo	Filippo	1530	1569
49	Susato	Tilman	1500	1561

Übungsblatt 2, Aufgabe 3 (2)

- Solange Sie keine bestimmte Sortierung in Ihrer SQL-Anfrage verlangen, darf das DBMS die Ergebniszeilen in irgendeiner Reihenfolge ausgeben (mathematisch: Menge/Multimenge).

  Das hängt von dem Auswertungsplan ab, für den sich der Optimierer entschieden hat. Bei der Beispiel-Ausgabe entspricht die Komponisten-Nummer `knr` wohl der Speicherreihenfolge, und der Optimierer hat einen einfachen „Full Table Scan“ gewählt. Das kann in einer anderen PostgreSQL-Version oder bei einer größeren Tabelle aber anders sein.

- Wenn Sie wollen, können Sie

  ```
  ORDER BY geboren
  ```

  an die Anfrage anhängen, um eine Sortierung nach dem Geburtsjahr zu bekommen.

  Wenn Sie für den Vergleich mit der erwarteten Ausgabe eine Sortierung nach der `knr` erzwingen wollen, schreiben Sie entsprechend „`ORDER BY knr`“.

Übungsblatt 2, Aufgabe 3 (3)

- Mögliche Lösung:

```
SELECT *
FROM   Komponist
WHERE  geboren >= 1500 AND geboren <= 1599
```

- Man kann auch alle Spalten ausschreiben:

```
SELECT KNr,Name,Vorname,geboren,gestorben
FROM   Komponist
WHERE  geboren >= 1500 AND geboren <= 1599
```

- Bei SQL-Anfragen in Programmen sollte man das machen, da der Tabelle vielleicht später nachträglich Spalten hinzugefügt werden.
- Bei „ad hoc“ Anfragen ist das unnötig.

Übungsblatt 2, Aufgabe 4 (1)

- Ein Studienkollege hat folgende Anfrage geschrieben:

```
SelecT GEBOREN
FROM   komponist
WHERE  vorname = "Peter", geboren > 1800.
```

- Leider enthält sie Syntaxfehler.
- Versuchen Sie zuerst, die Fehler zu finden, ohne die Anfrage auszuführen.
- Anschließend führen Sie die fehlerhafte Anfrage aus.

  Es ist auch nützlich, sich mit den Reaktionen von PostgreSQL auf Syntaxfehler vertraut zu machen.

- Dann korrigieren Sie die Anfrage bitte.

  Geben Sie die korrigierte Anfrage mit Kommentaren zu den Syntaxfehlern ab.

  Achten Sie darauf, nur echte Syntaxfehler zu nennen, keine Stilfehler.

Übungsblatt 2, Aufgabe 4 (2)

- Fehler in der gegebenen Anfrage:
  - Die Zeichenketten-Konstante muss in einfachen Anführungszeichen geschrieben werden: `'Peter'`.
  - Ein Komma unter `WHERE` ist nicht erlaubt, gemeint ist `AND`.
- Dagegen ist der Punkt am Ende kein (echter) Fehler. Die Zahlkonstante darf mit „`.`“ enden (schlechter Stil).
- Korrigierte Anfrage:

```
SelecT GEBOREN
FROM   komponist
WHERE  vorname = 'Peter' AND geboren > 1800.
```

Übungsblatt 2, Aufgabe 5 (1)

- Welchen Wert gibt folgende Anfrage aus?

```
SELECT char_length('\r\n''abc')
```

- Die Funktion `char_length` liefert die Länge einer Zeichenkette in Zeichen.

  Es gibt auch `octet_length` für die Länge in Bytes).

- Sie finden diese Funktion in der PostgreSQL-Dokumentation:
  [https://www.postgresql.org/docs/current/functions-string.html]
- Überlegen Sie zuerst selbst, bevor Sie die Anfrage ausführen.
- Geben Sie eine kurze Erklärung des Ergebnisses ab, und zwar als `.txt`-Datei. Bitte kein Word, und auch kein PDF.

Übungsblatt 2, Aufgabe 5 (2)

- Lösung:
  - Die Länge der Zeichenkette `'\r\n''abc'` ist 8.
  - Die äußeren `'...'` sind Begrenzer der Zeichenkette.
  - `\r` ist keine Escape-Sequence wie bei Java, sondern es sind zwei normale Zeichen.
  - `\n` entsprechend.
  - `''` ist ein einzelnes Hochkomma (einfaches Anführungszeichen, Apostroph).

    Der Compiler erkennt durch die Verdoppelung, dass die Zeichenkette noch nicht zu Ende ist. Escape-Sequenzen mit „\“ gibt es ja gerade nicht.

Allgemeine Hinweise (1)

- Anfragen, die in PostgreSQL nicht ausführbar sind (wegen Syntaxfehlern) werden automatisch mit 0 Punkten bewertet!

  Eventuelle Anmerkungen müssen als SQL-Kommentar geschrieben werden.

- Abweichungen vom erwarteten Ergebnis im Beispielzustand führen dagegen nicht automatisch zu 0 Punkten (aber schon zu Punktabzug).

  Sie müssen keine besonderen Tricks machen, um die oben gezeigten Ausgaben zu erreichen (das wäre eine Art von „mogeln“). Anfragen, die zwar im Beispielzustand des gewünschte Ergebnis ausgeben, aber mit der Aufgabenstellung kaum etwas zu tun haben, können auch mit 0 Punkten bewertet werden.

Allgemeine Hinweise (2)

- Bei Unklarheiten und Problemen können Sie Fragen im Forum in StudIP stellen.

  Bitte posten Sie aber keine Anfragen, die der Lösung nahe kommen. Bei komischen Syntaxfehlern, die Sie sich nicht erklären können, können Sie probieren, dem Dozenten oder dem Übungsleiter eine persönliche EMail mit der Anfrage zu schicken.

- Sie haben natürlich auch die Möglichkeit,
  - in der Übung zu fragen,
  - oder im Anschluss an die Vorlesung,
  - oder im Tutorium jeweils am Montag 12^{15} im Multimedia-Pool (Raum 3.02). Der Dozent ist anwesend so lange wie es Fragen gibt.

Allgemeine Hinweise (3)

- Schreiben Sie Ihren Namen (sowie Matrikelnummer etc.) bitte nicht in die Abgaben oder Dateinamen, da wir die Anfragen auch für ein Forschungsprojekt verwenden wollen.

  Das soll natürlich in anonymisierter Form geschehen, aber wir können nicht garantieren, dass wir identifizierbare Informationen auch im Text der Anfrage in allen Fällen finden und eliminieren.

- Vergessen Sie nicht die Angaben zum Vorrechnen (natürlich als SQL-Kommentar).

- Eventuell verwendete Quellen wie ChatGPT (oder auch die Lösung eines anderen Studierenden) müssen angegeben werden (mit der in Kapitel 0, Folien 22 bis 27 beschriebenen Codierung).

  Sie sind erwachsene Menschen und können sich ja denken, dass Sie nur durch eigene Beschäftigung mit den Aufgaben lernen.

Inhalt

1 Präsenzaufgabe 3
2 Hausaufgabe 3
3 Datentypen
4 LIKE
5 Funktionen
6 Hausaufgabe 2
7 Präsenzaufgabe

Präsenzaufgabe

- Es seien viele Übungsaufgaben zur Bestimmung der Nullstellen von Parabeln der folgenden Form gegeben:

$$y = x^2 + px + q$$

- Die Aufgaben stehen in folgender Tabelle (Schema parabel):

PARABEL		
ID	P	Q
1	-3	2
2	2	2
3	2	1
4	1	-12

- Schreiben Sie eine SQL-Anfrage, die für jede Parabel die Nullstellen bestimmt, soweit möglich (zwei Spalten für x_1, x_2).

Verwenden Sie die bekannte „pq-Formel“: $x_{1,2} = -\left(\frac{p}{2}\right) \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}$.

Form der Anfrage: `SELECT ID, ... AS X1, ... AS X2 FROM ... WHERE ...`