Einführung in Datenbanken

Übung 4: Datentypen in Logik und in SQL

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

Präsenzaufgabe: Syntaxgraph für Adventure-Spiele

- Zeichnen Sie Syntaxdiagramme für eine ganz einfache Kommandosprache eines Textadventure-Spiels.
- Typische Kommandos bestehen aus einem Verb und einem Objekt, z.B. “`take lamp`”.

  Verben: “`take`”, “`drop`”, “`examine`”, “`use`”.

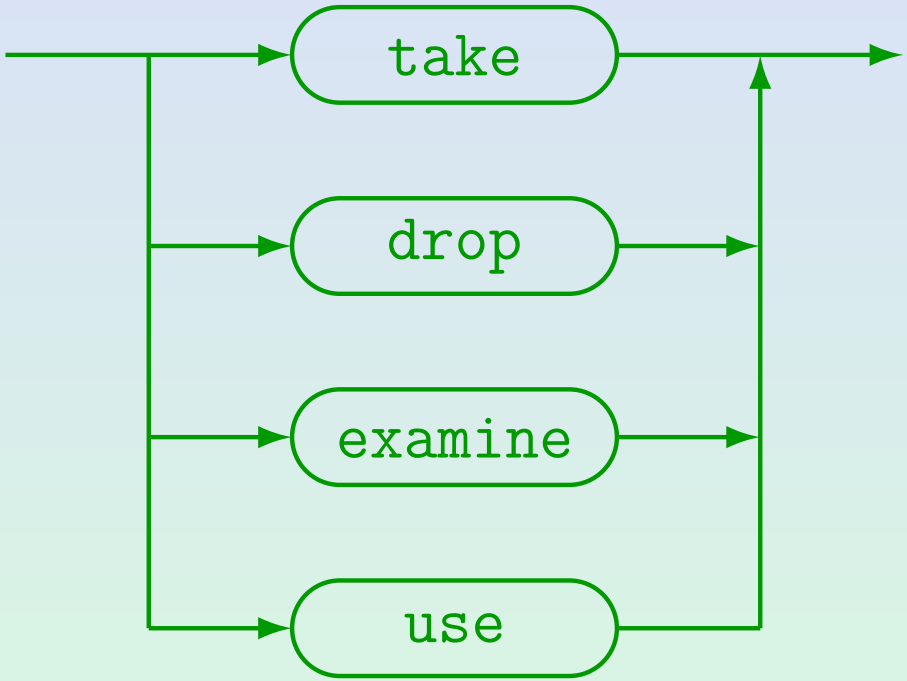
  Objekte: “`lamp`”, “`sword`”, “`rope`”.
- Man kann optional einen Artikel verwenden: „`take the lamp`“.
- Ein Verb kann auf mehrere Objekte angewendet werden, durch „and“ getrennt: “`take the lamp and the rope`”.

  Dies ist gleichbedeutend mit: “`take lamp`”, “`take rope`”.
- Sie müssen als Gruppen aus den „Breakout-Rooms“ abgeben.

  Geben Sie als PDF ab, oder auch ASCII (.txt).

Präsenzaufgabe: Musterlösung (1)

Verb:

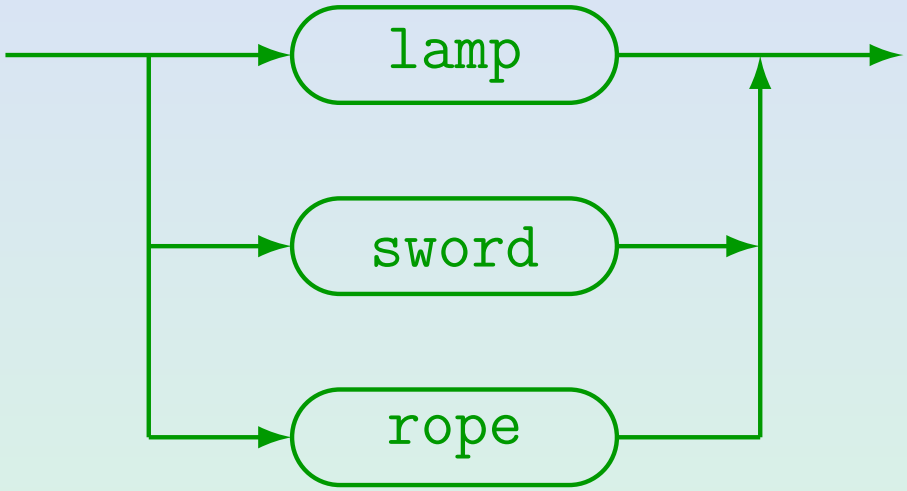


- Man kann sich eins der vier vorgegebenen Verben aussuchen.

  Nach der Liste in [https://zork.fandom.com/wiki/Command_List] hatte Zork ungefähr 28 normale Verben, plus 12 Kommandos zur Bewegung und 10 Steuerbefehle wie „`quit`“. Außerdem noch einige magische Befehle.

Präsenzaufgabe: Musterlösung (2)

Objekt:

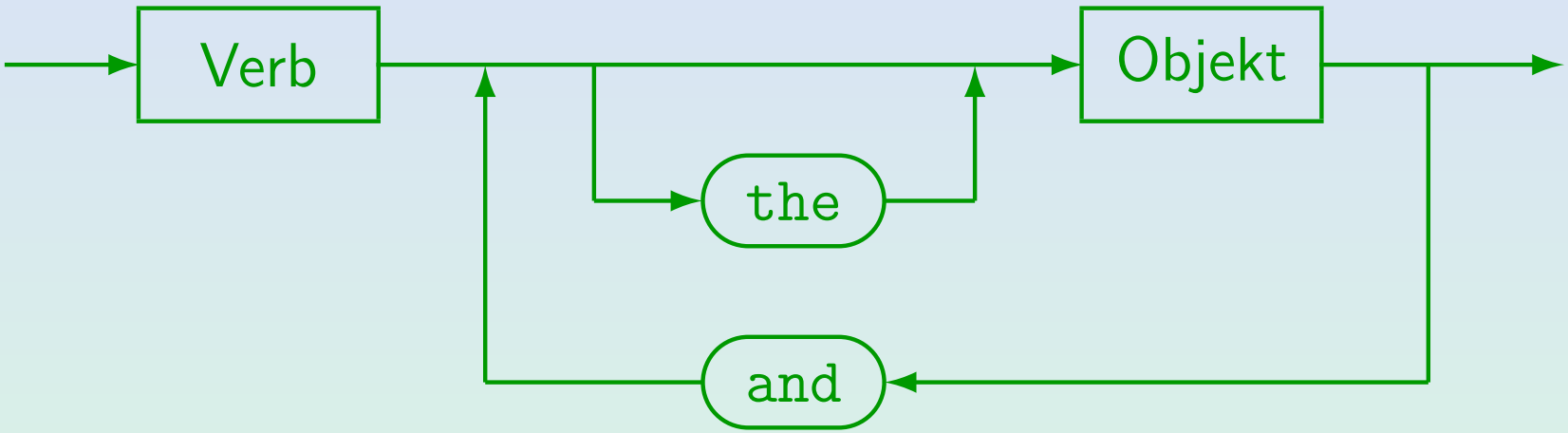


- Man kann eins der drei vorgegebenen Nomen wählen.

  Ein reales Spiel hätte natürlich deutlich mehr Objekte. Teils müssen Objekte auch über Adjektive oder andere Zusätze unterschieden werden, z.B. könnte es mehrere verschiedene Schwerter geben. Solange im aktuellen Raum und im Besitz des Spielers aber nur ein Schwert ist, kann man den Zusatz weglassen: Dann ist aus dem Kontext eindeutig, welches gemeint ist.

Präsenzaufgabe: Musterlösung (3)

Kommando:



- Beispiel: `take lamp`
- Beispiel: `take the lamp and the rope`
- Man könnte das Diagramm auch noch weiter zerlegen, z.B. „Nomen mit optionalem Artikel“ als syntaktische Kategorie einführen. Oder alles in ein Diagramm.

Präsenzaufgabe: Bemerkungen (1)

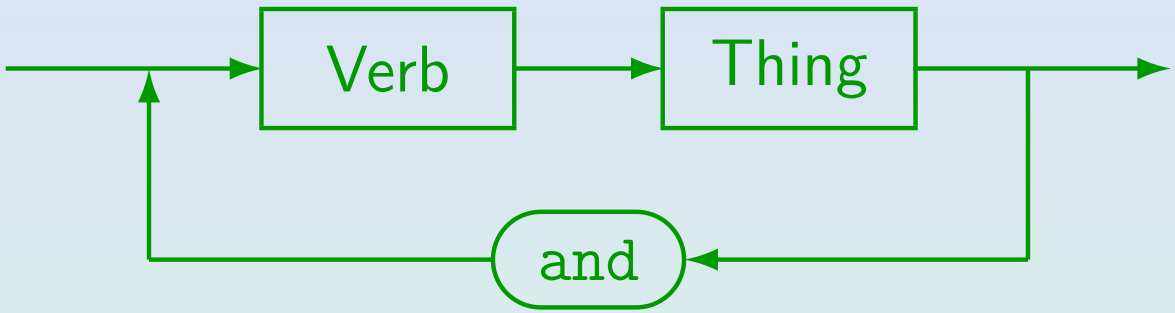
- Es gab überraschend viele nicht korrekte Lösungen (gefühlt mehr als die Hälfte).
- Diagramme brauchen einen eingehenden Pfeil („Start“) und einen ausgehenden Pfeil („Ende“).
- Man muss unterscheiden zwischen:
  - Rechtecken (syntaktische Kategorien),

    Dazu muss es ein weiteres Diagramm geben mit dem Namen, der in dem Rechteck steht.
  - Ovalen (Terminalsymbolen),

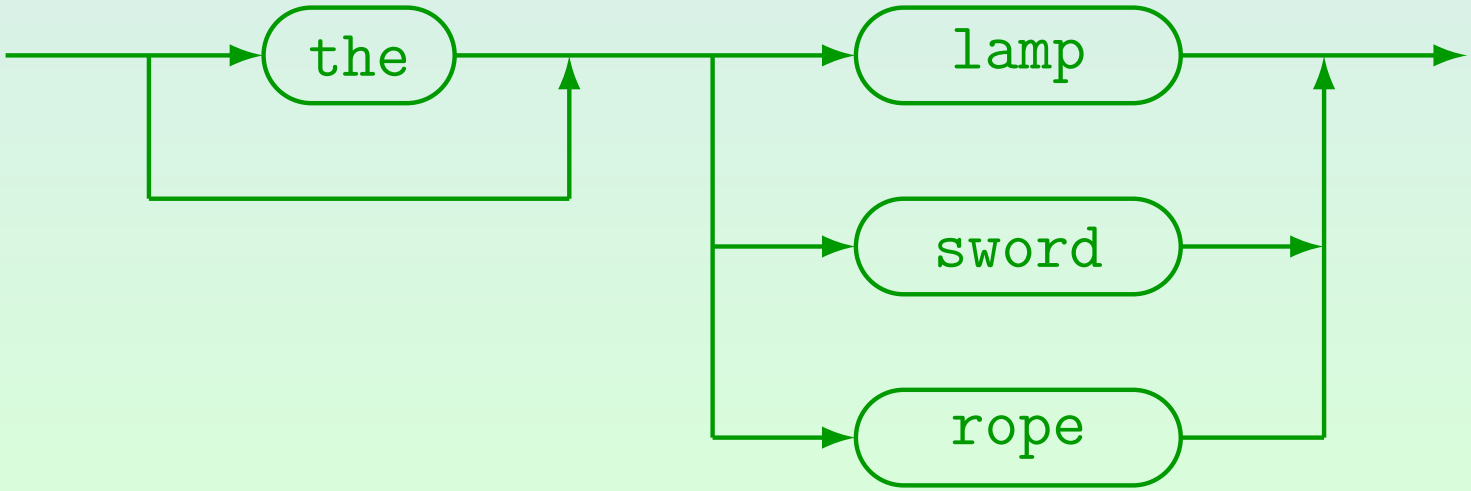
    Die Zeichen darin müssen genau so im Text erscheinen.
- Wenn man z.B. „`and`“ in ein Rechteck schreibt, muss es ein weiteres Diagramm mit Namen „`and`“ geben.

Präsenzaufgabe: Bemerkungen (2)

Kommando (korrekt?):



Thing:



Inhalt

Hausaufgabe 2a: EMP-DEPT-Datenbank (1)

- DEPT(DEPTNO, DNAME°, LOC°)
- EMP(EMPNO, ENAME°, JOB°, MGR° → EMP,
  HIREDATE°, SAL°, COMM°, DEPTNO° → DEPT)
- SALGRADE(GRADE, LOSAL°, HISAL°)
- Groß-/Kleinschreibung ist egal.

  PostgreSQL wandelt alle Bezeichner ohne `"..."` in Kleinbuchstaben um, Oracle alle in Großbuchstaben. Dieses Beispiel kommt original von Oracle.

- Es ist schlecht, dass für alle Attribute außer Primärschlüsseln Nullwerte erlaubt sind.

  Tatsächlich Nullwerte enthalten sind in nur in `EMP`, und dort in den Spalten `MGR` (Mr. King hat keinen Vorgesetzten), `COMM` (nur Angestellte mit `JOB = 'SALESMAN'` bekommen eine Provision/„commission“) und `DEPTNO` (Mr. Smith ist keiner Abteilung zugeordnet).

Hausaufgabe 2a: EMP-DEPT-Datenbank (2)

- Die Datentypen sollten nicht mit angegeben werden.

  Man gibt sie in dieser Notation nur bei Bedarf an. Es wird dadurch nicht übersichtlicher.

- Die Datentypen wurden gegenüber dem Original verändert.

  [https://livesql.oracle.com/apex/livesql/file/content_O5AEB2HE08PYEPTGCFLZU9YCV.html]

- Der Typ `text` aus PostgreSQL ist nicht portabel.

  Der SQL-Standard kennt keinen solchen Typ. Manche anderen DBMS haben ihn, aber die mögliche maximale Länge ist unterschiedlich.
  Bei PostgreSQL sind es ca. 1 GB. Bei MySQL 64 KB.

- Der Typ `integer` ist im SQL Standard vorgesehen.

  Aber der Standard lässt den Wertebereich offen. Üblich sind 32-Bit Zahlen.

Hausaufgabe 2a: EMP-DEPT-Datenbank (3)

- Teils wurde die Reihenfolge der Spalten von `EMP` geändert.
  - Streng genommen ist das ein Fehler:
    Die Spalten einer Tabelle haben eine definierte Reihenfolge.

    Die Zeilen dagegen nicht.
- Öfters wurden die Markierungen für den möglichen Nullwert vergessen (° oder ?).
- Manchmal wurden bei Fremdschlüsseln nicht nur die referenzierte Tabelle, sondern auch die Spalte mit angegeben.
  - Es soll eine Kurz-Notation sein.
    Dabei werden immer die Primärschlüssel referenziert.

    Ich würde das nur im Notfall angeben (wenn andere Reihenfolge oder Sekundärschlüssel).

Hausaufgabe 2b: CREATE TABLE (1)

VORLESUNGS_ANGEBOT			
VORLESUNG	SEMESTER	DOZENT	UEB_GRUPPEN
OOP	WS 2018/19	Stefan Brass	7
OOP	WS 2019/20	Matthias Hagen	7
OOP	WS 2020/21	Matthias Hagen	
Einf. in DBen	WS 2019/20	Stefan Brass	2
Einf. in DBen	WS 2020/21	Stefan Brass	4

```
CREATE TABLE VORLESUNGS_ANGEBOT(
       VORLESUNG    VARCHAR(80) NOT NULL,
       SEMESTER     VARCHAR(12) NOT NULL,
       DOZENT       VARCHAR(30) NOT NULL,
       UEB_GRUPPEN NUMERIC(2),
       PRIMARY KEY(VORLESUNG, SEMESTER))
```

Hausaufgabe 2b: CREATE TABLE (2)

- Der gegebene Beispiel-Wert für den Vorlesungs-Titel
  „`Objektorientierte Programmierung`“
  hat 32 Zeichen. Der Typ `VARCHAR(`n`)` mit kleinerem n wäre ein Fehler. MGdI/KdM hat 69 Zeichen!
- Der Typ `CHAR(100)` ist schlecht: Er benötigt immer Platz für 100 Zeichen.

  Das sind Zeichenketten fester Länge. Die eingefügten Zeichenketten werden nach rechts mit Leerzeichen aufgefüllt. Für `SEMESTER` macht `CHAR(10)` vielleicht Sinn.
- `VARCHAR(14)` für `DOZENT` funktioniert gerade für die Beispiel-Daten. Aber denken Sie an `Matthias Müller-Hannemann` (25 Zeichen).
- `VARCHAR(127)` für `SEMESTER` ist überdimensioniert.

Hausaufgabe 2b: CREATE TABLE (3)

- Der Typ `INT(1)` ist nicht portabel: Bei PostgreSQL gibt es einen Syntaxfehler.

  In MySQL ist der Parameter die maximale Ausgabebreite.

- Der Typ `NUMERIC(3,1)` für die Anzahl Übungsgruppen ist falsch: Es kann doch keine halben Übungsgruppen geben?

- `NUMERIC(1)` für die Übungsgruppen könnte etwas wenig sein (es funktioniert aber für die Beispiel-Daten).

- Ziemlich häufig wurde der Schlüssel vergessen (kostet einen Punkt).

- Ein Mal `UNIQUE` statt `PRIMARY KEY`.

  Wozu? Macht Fremdschlüssel schwerer (man muss Spalten angeben).

Deklarative Anfragesprache am Beispiel

- Stellen Sie sich vor, die Hausaufgabenpunkte sind in einer Textdatei gespeichert mit dem Format

  Vorname,Nachname,Aufgabennummer,Punkte

  (d.h. ein Tupel der Tabelle `ABGABEN` pro Zeile).

- Welchen Aufwand schätzen Sie für die Entwicklung eines Java-Programms, das die Gesamtpunktzahl je Student/-in ausgibt (alphabetisch geordnet)?

  Anzahl Zeilen: _____ (ohne Kommentare)

  Arbeitszeit: _____

- Trauen Sie sich das zu? (A: ja, B: mit viel Zeit, C: nein)

- In SQL braucht man 4 Zeilen und 2 Minuten Zeit.

Bonusaufgabe: Anfrage in Java (1)

- Tabelle für Ergebnisse:

```
CREATE TABLE H2_BONUS(
        ABGABE_NR   NUMERIC(6)    NOT NULL,
        ZEIT        NUMERIC(3)    NULL,
        ZEILEN      NUMERIC(3)    NOT NULL,
        CODEZEILEN  NUMERIC(3)    NOT NULL,
        PUNKTE      NUMERIC(2,1)  NOT NULL,
        KOMMENTAR   VARCHAR(200)  NOT NULL,
        TUTOR       CHAR(2)       NOT NULL,
        PRIMARY KEY(ABGABE_NR));
```

- Anzahl Abgaben:

```
SELECT COUNT(*) FROM H2_BONUS;
```

  Ergebnis: 37

Bonusaufgabe: Anfrage in Java (2)

- Anzahl Abgaben mit voller Punktzahl:

```
SELECT COUNT(*)
FROM   H2_BONUS
WHERE  PUNKTE >= 5;
```

  Ergebnis: 28

- Intervall für Zeit bei voller Punktzahl:

```
SELECT COUNT(*),
       MIN(ZEIT), MAX(ZEIT), AVG(ZEIT)
FROM   H2_BONUS
WHERE  PUNKTE >= 5 AND ZEIT IS NOT NULL
```

  Ergebnis: `COUNT`: 24, `MIN`: 20, `MAX`: 180, `AVG`: 80

  Bei Aggregationsfunktionen werden Nullwerte automatisch ignoriert.

  Die Bedingung, dass `ZEIT` nicht Null ist, beeinflusst hier nur `COUNT(*)`.

Bonusaufgabe: Anfrage in Java (3)

- Durchschnittswert ist 80min, aber Medianwert ist ca. 60min:

```
SELECT COUNT(*),
       MIN(ZEIT), MAX(ZEIT), AVG(ZEIT)
FROM   H2_BONUS
WHERE  PUNKTE >= 5 AND ZEIT <= 60
```

  Ergebnis: 12 Studierende (von 24)

- Anzahl Zeilen:

```
SELECT COUNT(*), MIN(ZEILEN), MAX(ZEILEN),
       AVG(ZEILEN), ..., AVG(ZEILEN/CODEZEILEN)
FROM   H2_BONUS
WHERE  PUNKTE >= 5
```

  28 Einsendungen, Zeilen von 34 bis 181 (∅ 82),
  Codezeilen von 28 bis 139 (∅ 64), Zeilen/Codezeilen: ∅ 1.28

Bonusaufgabe: Lösung mit TreeMap (1)

```
(1)  import java.io.BufferedReader;
(2)  import java.io.FileReader;
(3)  import java.io.IOException;
(4)  import java.util.Map;
(5)  import java.util.TreeMap;
(6)
(7)  public class Query {
(8)
(9)      // Constants:
(10)     private static final String FILE =
(11)             "abgaben.csv";
(12)     private static final String DELIM =
(13)             ",";
(14)
(15)     public static void main(String args[]) {
(16)
```

Bonusaufgabe: Lösung mit TreeMap (2)

```
(17)        // Open data file:
(18)        BufferedReader br;
(19)        try {
(20)            br = new BufferedReader(
(21)                new FileReader(FILE));
(22)        }
(23)        catch(IOException e) {
(24)            System.err.println(
(25)                "Error opening file: " + FILE);
(26)            System.err.println(e);
(27)            //e.printStackTrace();
(28)            return;
(29)        }
(30)
```

Bonusaufgabe: Lösung mit TreeMap (3)

```
(31)        // Read data into TreeMap:
(32)        TreeMap<String,Integer> map =
(33)                new TreeMap<>();
(34)        while(true) {
(35)            // Read a line from the file:
(36)            String line = null;
(37)            try {
(38)                line = br.readLine();
(39)            } catch(IOException e) {
(40)                System.err.println(
(41)                    "Error while reading");
(42)                System.err.println(e);
(43)                System.exit(1);
(44)            }
(45)
```

Bonusaufgabe: Lösung mit TreeMap (4)

```
(46)            // Stop if nothing read:
(47)            if(line == null)
(48)                break;
(49)
(50)            // Split line into data values:
(51)            String[] row = line.split(DELIM);
(52)            if(row.length != 4) {
(53)                System.err.println(
(54)                    "Bad format: " + line);
(55)                System.exit(2);
(56)            }
(57)            String first = row[0];
(58)            String last = row[1];
(59)            // row[2] is exercise number
(60)            // row[3] is number of points
```

Bonusaufgabe: Lösung mit TreeMap (5)

```
(61)            int points = 0;
(62)            try {
(63)                points =
(64)                    Integer.parseInt(row[3]);
(65)            }
(66)            catch(NumberFormatException e) {
(67)                System.err.println(
(68)                    "No number: " + row[3]);
(69)                System.exit(3);
(70)            }
(71)
(72)            // Composed student name (key):
(73)            String name = last + ", " + first;
(74)
```

Bonusaufgabe: Lösung mit TreeMap (6)

```
(75)            // Add student or increment points:
(76)            Integer oldPoints = map.get(name);
(77)            if(oldPoints == null)
(78)                map.put(name, points);
(79)            else
(80)                map.put(name,
(81)                    oldPoints + points);
(82)        } // End of loop over input lines
(83)
```

- Der Vorteil der `TreeMap` ist das die Methode
  `public Set<Map.Entry<K,V>> entrySet()`
  eine `Set`-Sicht auf die Daten liefert, deren Iterator die Einträge in Sortierreihenfolge liefert.

  [https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html]

  Außerdem ist die logarithmische Komplexität garantiert.

Bonusaufgabe: Lösung mit TreeMap (7)

```
(84)            // Print results:
(85)            for(Map.Entry<String,Integer> entry :
(86)                    map.entrySet()) {
(87)                String name = entry.getKey();
(88)                Integer points = entry.getValue();
(89)                System.out.println(
(90)                    name + ": " + points);
(91)            }
(92)        }
(93)    }
```

- Ich habe 67 min gebraucht, kannte aber schon Lösungen von Studierenden. (Original: 78 Zeilen, 59 Codezeilen)
- Kleines Problem: Nachname „`Von Strauch`“ wird vor Nachname „`Von`“ eingeordnet.

Weil das Leerzeichen im ASCII Code vor dem Trennzeichen „,“ steht.

Bonusaufgabe: Weitere Lösungen

- Eine hinsichtlich der Trennung von Vor- und Nachname sauberere Lösung ist:
  - Man legt eine Klasse für Studierende an.

    Mit den Attributen Vorname, Nachname, Punkte.
  - Die Daten werden in einer Liste (z.B. `ArrayList`) von Studierenden-Objekten gehalten.
  - Wenn man einen neuen Datensatz gelesen hat, durchsucht man die Liste linear, und inkrementiert die Punkte oder fügt ein neues Objekt an.
  - Die Studierenden-Klasse muss das Interface `Comparable` implementieren, dann kann man die Liste mit `Collections.sort(...)` sortieren.
- Es gab auch Lösungen mit einzelnen Listen für Namen und Punkte.

Bonusaufgabe: SQL zum Vergleich

- Tabelle:

ABGABEN			
VORNAME	NACHNAME	AUFGABE	PUNKTE
Lisa	Weiss	1	10
Michael	Grau	1	9
Daniel	Sommer	1	5
Lisa	Weiss	2	8

- Hier zum Vergleich die Lösung in SQL:

```
SELECT VORNAME, NACHNAME, SUM(PUNKTE)
FROM   ABGABEN
GROUP  BY VORNAME, NACHNAME
ORDER  BY NACHNAME, VORNAME
```

Aggregationsfunktionen wie `SUM` und die `GROUP BY` und `ORDER BY`-Klauseln werden später in dieser Vorlesung ausführlich besprochen.

Inhalt

Kapitel 4: SQL — Geschichte und Standards

Nach diesem Kapitel sollten Sie Folgendes können:

- Das Jahrzehnt nennen, in dem das relationale Modell und die erste Version von SQL entwickelt wurden.
- Den Namen des Forschers nennen, der das relationale Modell erfunden hat.
- Das Jahr nennen, in dem der erste SQL-Standard erschienen ist, sowie ein paar Worte zur weiteren Entwicklung sagen.

  Z.B. das Jahr des aktuellen Standards ungefähr benennen, oder auch, wie häufig ungefähr neue Standards erscheinen.
- Sich eine Version des Standards besorgen.

  Kostenpflichtig oder inoffizielle Vorversionen im Internet.
  Zusätzlich auch sich eine SQL-Grammatik im Internet besorgen.

Geschichte des Relationalen Modells (1)

- Wer gilt als Erfinder des relationalen Modells?
  - A. Donald E. Knuth
  - B. Jim Gray
  - C. Edgar F. Codd
  - D. Charles Bachmann
  - E. Michael Stonebraker

Alle sind Turing-Preisträger [https://de.wikipedia.org/wiki/Turing_Award]

Geschichte des Relationalen Modells (2)

- Wann erschien sein Artikel „A Relational Model of Data for Large Shared Data Banks“?

  A. 1950

  B. 1960

  C. 1970

  D. 1980

  E. 1990

SQL Standard (1)

- Wann erschien die erste Version des SQL-Standards?

A. 1970

B. 1980

C. 1986

D. 1992

E. 1999

SQL Standard (2)

- Welches ist die aktuelle Version des SQL-Standards?
  - A. SQL-92
  - B. SQL-99
  - C. SQL-2003
  - D. SQL-2016
  - E. SQL-2019

Inhalt

Signaturen

- Was ist der Zweck einer Signatur?
- Was sind die Bestandteile einer Signatur?
- Wie werden sie interpretiert?
- Was fehlt in diesem formalen Rahmen, um Datentypen in SQL vollständig zu beschreiben?
- Hatten Sie vor dieser Vorlesung
  - A. Gar keine Prädikatenlogik
  - B. Einsortige Prädikatenlogik
  - C. Mehrsortige Prädikatenlogik
  - D. Mehrsortige Prädikatenlogik mit Überladen
  - E. Mehrsortige Prädikatenlogik mit Subklassen (OCL?)

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

Präsenzaufgabe: PQ-Formel in SQL

- 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:

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 ...`