# Einführung in Datenbanken

## Übung 2: DB-Managementsysteme und DBMS-Funktionen

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

Wintersemester 2023/24

http://www.informatik.uni-halle.de/~brass/db23/

# Inhalt

# Zum Konzept von Präsenzaufgaben (1)

- In den letzten Jahren hatten wir Aufgaben, die in den Übungen gelöst werden mussten, und wie Hausaufgaben einige Punkte für die Studienleistung brachten.
- Die Lösungen konnten im Prinzip nur während der Übungszeit in die Übungsplattform hochgeladen werden.

  Es war ausdrücklich nicht erwünscht, dass Nicht-Teilnehmer der Übung auch Lösungen abgegeben haben. Die (wenigen) Punkte für die Präsenzaufgaben sollten auch die Anwesenheit in der Übung belohnen.
- Im Gegensatz zu den Hausaufgaben war bei den Präsenzaufgaben Gruppenarbeit erwünscht.

  Als die Übungen wegen Corona online gehalten werden mussten, wurden die Gruppen vom Videokonferenz-System zufällig zusammengestellt. So ergab sich die Gelegenheit, wenigstens einige sozialen Kontakte zu haben, und auch neue Studierende kennenzulernen.

# Zum Konzept von Präsenzaufgaben (2)

- Wenn die Übung im Hörsaal gehalten wird, ist das etwas schwierig, da nicht jeder ein Notebook dabei hat.
- Wer hat ein Notebook oder ein ähnliches Gerät dabei?

  Im Prinzip würde es auch mit einem Handy gehen, aber das wäre umständlich. Man sollte schon einen etwas größeren Bildschirm und eine Tastatur haben. Es würde reichen, wenn es in jeder Gruppe ein Notebook gibt.
- Wir denken auch darüber nach, eine Abgabe auf Papier zu ermöglichen.

  Dann müssten die Abgaben aber von Tutoren abgetippt werden, was keinen Spaß macht. Die Tutoren sind ohnehin sehr beschäftigt mit den Hausaufgaben. Wir sind auch etwas unsicher, ob das mit der ganz neuen Übungsplattform überhaupt geht.
- Wäre hätte gerne Präsenzaufgaben in Gruppenarbeit?

# Präsenzaufgabe aus früherem Semester (1)

- Gehen Sie auf die Webseite
  [https://dbup2date.uni-bayreuth.de/wetterdaten.html]

  Es wird eine MySQL-Datenbank offenbar auf einem Linux-Rechner verwendet. Die Groß-/Kleinschreibung der Tabellennamen ist wichtig. Dies ist also nicht der Adminer und nicht PostgreSQL, aber auch eine Web-Schnittstelle zu einer SQL-Datenbank.

- Gesucht sind die Standorte aller Wetterstationen, deren Höhe über 1000m ist.

- Geben Sie die SQL-Anfrage für diese Aufgabe ab.

- Sie müssen als Gruppen aus den „Breakout-Rooms“ abgeben.

  Bitte in der Übungsplattform in StudIP abgeben (als Gruppe). Es reicht eine ASCII `.txt` Datei. Sie z.B. über die „geteilten Notizen“ kommunizieren. Oder [https://sharelatex.informatik.uni-halle.de/].

# Präsenzaufgabe aus früherem Semester (2)

| Wetterstation | | | | |
|---|---|---|---|---|
| S_ID | Standort | Geo_Breite | Geo_Laenge | Hoehe |
| 102 | Leuchtturm Alte Weser | 53.863 | 8.127 | 32 |
| 164 | Angermünde | 53.032 | 13.991 | 54 |
| 183 | Arkona | 54.679 | 13.434 | 42 |
| 232 | Augsburg | 48.425 | 10.942 | 462 |
| 282 | Bamberg | 49.874 | 10.921 | 240 |
| 403 | Berlin-Dahlem (FU) | 52.454 | 13.302 | 51 |
| 430 | Berlin-Tegel | 52.564 | 13.308 | 36 |
| 433 | Berlin-Tempelhof | 52.468 | 13.402 | 48 |
| 691 | Bremen | 53.045 | 8.798 | 4 |
| 722 | Brocken | 51.799 | 10.618 | 1135 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 346 | Feldberg/Schwarzwald | 47.875 | 8.004 | 1490 |
| 1358 | Fichtelberg | 50.428 | 12.954 | 1213 |
| 5792 | Zugspitze | 47.421 | 10.985 | 2965 |

# Präsenzaufgabe aus früherem Semester (3)

- In der obigen Tabelle wurde die Spalte „Betreiber“ weggelassen (kein Platz).

  Sie ist auch nicht besonders spannend, der Wert ist „DWD“ in jeder Zeile.

- Man bekommt die ganze Tabelle angezeigt mit dieser SQL-Anfrage:

```
SELECT *
FROM   Wetterstation
```

- Das Schema der Tabelle war oben auf der Seite so gezeigt:

```
Wetterstation(S_ID: int, Standort: varchar,
              Geo_Breite: double,
              Geo_Länge: double,
              Hoehe: int, Betreiber: varchar)
```

Wir verwenden diese Kurznotation auch in dieser Vorlesung, allerdings meist ohne Datentypen (und die Notation für Fremdschlüssel ist anders).

# SQL-Anfragen am Beispiel (1)

- Eine korrekte Lösung war:

```
SELECT Standort
FROM   Wetterstation
WHERE  Hoehe > 1000
```

- Es ist auch möglich, alles in eine Zeile zu schreiben:

```
SELECT Standort FROM Wetterstation WHERE Hoehe > 1000
```

- Die Groß-/Kleinschreibung von Schlüsselworten (wie `SELECT`) und Bezeichnern (Tabellennamen, Spaltennamen, ...) ist in SQL normalerweise egal.
- Bei MySQL auf Linux-Rechnern ist die Groß-/Kleinschreibung von Tabellennamen aber wichtig (Tabellen entsprechen Dateien auf dem Server).

# SQL-Anfragen am Beispiel (2)

- Es ist z.B. auch möglich:

```
select standort
from   Wetterstation
where  hoehe > 1000
```

- Das korrekte Ergebnis der Anfrage ist:

| Standort |
|---|
| Brocken |
| Feldberg/Schwarzwald |
| Fichtelberg |
| Großer Arber |
| Zugspitze |

- Da nicht explizit eine Sortierung vorgegeben war, wäre jede andere Reihenfolge auch möglich.

# SQL-Anfragen am Beispiel (3)

- Es wird ein halber Punkt abgezogen, wenn nicht nur die verlangte Spalte „Standort“ ausgegeben wurde, sondern alle Spalten (`SELECT *`).
- Die Aufgabe war: „Gesucht sind die Standorte aller Wetterstationen, deren Höhe über 1000m ist.“

  Natürlich macht es Sinn, sich am Anfang mindestens auch die Höhe ausgeben zu lassen, um zu sehen, dass die Auswahl richtig geklappt hat. Aber vor der Abgabe der Lösung muss man es auf die verlangten Spalten reduzieren.
- Wichtig ist auch die Unterscheidung zwischen „(echt) größer“ und „größergleich“. Die Bedingung `Hoehe >= 1000` würde auch zum Punktabzug führen.
- Die Aufgabe war: „Gesucht sind die Standorte aller Wetterstationen, deren Höhe über 1000m ist.“

# SQL-Anfragen am Beispiel (4)

- Zu den Kompetenzen, die man für die Klausur erlernen muss, gehört auch, die Aufgabenstellung genau zu lesen und auf solche Feinheiten zu achten.

  Es ist beabsichtigt, dass es nur eine Abbildung von Datenbank-Zuständen auf Antworten gibt, die den Aufgabentext erfüllt. Sehr selten bemerken wir allerdings bei der Korrektur, dass die Aufgabenstellung nicht eindeutig war. Dann müssen wir Anfragen akzeptieren, die nicht äquivalent zur Musterlösung sind (zwei Anfragen sind äquivalent, wenn Sie die gleiche Abbildung von Zuständen auf Antworten beschreiben).

- Selbstverständlich könnte man in SQL auch schreiben:

```
WHERE 1000 < Hoehe
```

- Auch so würde es funktionieren, riskiert aber Punktabzug wegen unnötiger Verkomplizierung (und Index-Problem):

```
WHERE Hoehe - 1000 > 0
```

# SQL-Anfragen am Beispiel (5)

- Die folgende Abgabe ist nicht in Ordnung:

```
Präsenzaufgabe Übung 1

SELECT Standort
FROM   Wetterstation
WHERE  Hoehe>1000
```

- Bei automatischer Korrektur würde die erste Zeile einen Syntaxfehler geben, und die Abgabe automatisch mit 0 Punkten bewertet.

  Die freundlichen Tutoren werden das (diesmal noch) nicht tun ...

- In SQL beginnen Kommentare mit „`--`“ und erstrecken sich bis zum Ende der Zeile. So wäre es korrekt:

```
-- Präsenzaufgabe Übung 1
```

# SQL-Anfragen am Beispiel (6)

- Für folgende Abgabe würde es einen heftigen Punktabzug geben:

```
select *
from   wetterstation
where  hoehe > 1000
```

- Die Anfrage wurde offensichtlich nicht getestet, da sie eine Fehlermeldung liefert:

```
Table 'wetterdaten.wetterstation' doesn't exist
```

  Wie oben erläutert, ist das eine Spezialität von MySQL. Bei allen anderen mir bekannten Systemen (und vermutlich nach dem SQL-Standard) sind Tabellen-Namen case-insensitiv. Da aber genau diese Webschnittstelle vorgegeben war, muss man sich auch an die Syntax dieses Systems halten.

- Testen Sie alle Anfragen vor der Abgabe.

# SQL-Anfragen am Beispiel (7)

- Folgende Abgabe enthält einen Typfehler:

```
select Standort
from   Wetterstation
where  Hoehe > '1000m'
```

- Die Spalte Höhe hat den Datentyp `int` (ganze Zahl) und kann nicht mit einer Zeichenkettenkonstante `'1000m'` verglichen werden.
- In MySQL gibt das aber keinen Fehler.

  In MySQL ist `1 = '1abc2345'` true (nur numerischer Präfix wird konvertiert).

- In PostgreSQL schon:

```
ERROR: invalid input syntax for integer: "1000m"
LINE 3: where Hoehe > '1000m';
                      ^
```

# SQL-Anfragen am Beispiel (8)

- Auch dies ist ein Typfehler, würde aber in PostgreSQL funktionieren (MySQL sowieso):

```
select Standort
from   Wetterstation
where  Hoehe > '1000'
```

- Nach dem SQL-Standard ist es ein Typfehler.

  Die verglichenen Werte müssen von kompatiblen Datentypen sein:
  Alle numerischen Typen sind kompatibel und alle String-Typen ebenfalls,
  aber numerische Typen sind nicht kompatibel mit String-Typen.

- In DB2 und Access gibt es auch eine Fehlermeldung.
- Es wird wahrscheinlich ein halber Punkt abgezogen, obwohl es funktioniert.

# SQL-Anfragen am Beispiel (9)

- Dies ist stilistisch nicht schön:

```
select Wetterstation.Standort
from   Wetterstation
where  Hoehe > 1000
```

- Man darf Tabellennamen vor den Spaltennamen schreiben.

  Tatsächlich ist es der Name einer Tupelvariable, die standardmäßig so heißt wie die zugehörige Tabelle. Das kommt alles in der Vorlesung erst später. Versuchen Sie, nur in der Vorlesung eingeführte Konstrukte zu verwenden.

- Es gibt keinerlei Notwendigkeit für die Verkomplizierung.

  Wenn sich eine Anfrage auf mehr als eine Tabelle bezieht, kann es notwendig sein, die Spaltenreferenzen eindeutig zu machen.

- Warum wird der Spaltenname dann nicht auch bei `Hoehe` vorangestellt? (Dann wäre es wenigstens einheitlich.)

# SQL-Anfragen am Beispiel (10)

- Es ist schon auffällig, wenn sehr spezielle Lösungen zweimal in den Abgaben auftauchen:
  - Z.B. die Lösung mit `'1000m'`.
  - Auch die Lösung mit `Wetterstation.Standort`, aber `Hoehe` ohne den Zusatz.
- Wenn jemand Probleme hat, und Sie sind schon fertig, beraten Sie Ihren Nächsten/Ihre Nächste.

  Das ist natürlich ethisch wertvoll. Lassen Sie sich das Problem erklären, und geben Sie Tipps, bzw. lösen Sie dieses Problem. Wenn Sie Ihre Lösung zum Abschreiben geben, nehmen Sie dem/der Betreffenden die Chance, sich selbst mit der Aufgabe zu beschäftigen (nicht ethisch wertvoll!).
- Wenn die Quelle nicht korrekt angegeben wird, können Plagiate harte Konsequenzen haben!

# Inhalt

# Hinweise zur ersten Hausaufgabe (1)

- Es gibt ein erstes Hausaufgabenblatt (10 Punkte).

  Sie finden es in der Übungsplattform im [StudIP-Eintrag der Vorlesung], (Reiter „LTI-Tool“) und auch auf der Webseite der Vorlesung (im Menu unter Übung). Das Blatt war noch am Montag (16.10.2023) auf der Webseite (allerdings recht spät).

- Abgabeschluss ist nächster Montag, 23.10.2023, 18:00.
- Hausaufgaben sind einzeln zu bearbeiten!

  Oder, wenn Sie unbedingt eine nicht selbst erstellte Lösung übernehmen wollen, müssen Sie wenigstens die Quelle korrekt angeben.

- Thema des Blattes ist das CSV-Format („comma-separated values“) zum Austausch von Tabellen.
- Außerdem sollen Sie sich im WWW umschauen, und fünf interessante Webseiten zu Datenbanken nennen.

# Hausaufgabe 1: CSV, Link-Liste (1)

- Es sollten Daten für diese Tabelle im CSV-Format abgeliefert werden:

| WEBSEITEN | | |
|---|---|---|
| STUD | NR | URI |
| abcde | 1 | http://db-engines.com/ |
| abcde | 2 | https://www.postgresql.org/ |

- webseiten.csv:

```
abcde,1,https://db-engines.com/
abcde,2,https://www.postgresql.org
```

- In der ersten Spalte verwenden Sie bitte Ihren fünfstelligen Nutzer-Account.

  So können wir alle Dateien zusammenfügen und eine große Datei in eine Datenbank laden.

# Hausaufgabe 1: CSV, Link-Liste (2)

- Bevor wir die Daten allgemein zur Verfügung stellen, werden wir die Accounts verändern (z.B. `abcNN`).
- Man darf Datenwerte in `"..."` einschließen.
  - Man muss das tun, wenn ein Datenwert ein Komma, einen Zeilenumbruch oder ein Anführungszeichen enthält.
  - Ein Anführungszeichen im Innern von `"..."` muss verdoppelt werden.
- Beispiel:

```
abcde,1,"https://db-engines.com/"
"abcde","2","https://www.postgresql.org"
```

- Auf dem Übungsblatt sind [RFC 4180], [englische Wikipedia] und [deutsche Wikipedia] verlinkt.

# Hausaufgabe 1: CSV, Link-Liste (3)

- Beachten Sie bitte, dass viele Varianten des CSV-Formats benutzt werden, z.B. auch mit Semikolon statt Komma.

  Sie riskieren 0 Punkte, wenn Sie eine Datei mit einem Syntaxfehler abgeben. Insbesondere sollten Sie, wenn Sie die Datei aus Excel im CSV-Format exportiert haben, sie noch einmal in einem Texteditor anschauen (z.B. Notepad++ [https://notepad-plus-plus.org/]).

- Codieren Sie die Zeichen bitte als UTF-8.

  Also Nicht-Umlaute wie ASCII (ein Byte), Umlaute dagegen mit zwei Byte. Das CSV-Format sagt leider nichts über die Zeichencodierung aus.

- Verwenden Sie keine Kopfzeile mit den Spaltennamen.

  Die Kopfzeile ist im CSV-Format erlaubt, aber nicht vorgeschrieben. Leider kann man nicht an der Datei erkennen, ob sie verwendet wird. Man kann sie mit der ersten Datenzeile verwechseln. Im MIME-Medientyp gibt es einen Parameter dafür.

# Inhalt

# Lernziele von Kapitel 2: DBMS-Funktionen

- Die Verwendung eines DBMS zur Verwaltung persistenter Daten mit der (direkten) Verwendung von Dateien vergleichen.
  - Einige Vorteile der Lösung mit DBMS nennen.

    Und auch mögliche Nachteile.
  - Vor-/Nachteile für ein konkretes Projekt bewerten.
- „Datenunabhängigkeit“ erklären. Was ist ein Index?

  In diesem Kapitel wird hauptsächlich physische Datenunabhängigkeit behandelt.
- Vorteile deklarativer Anfrage/Programmiersprachen benennen.
- Den Begriff der Transaktion erklären, Vorteile der Transaktionsverwaltung im DBMS richtig einschätzen.

# Persistente Daten: Wie persistent sind Platten?

- Ein DBMS dient zur Verwaltung persistenter Daten, die Sie dauerhaft speichern wollen.

  Speziell strukturierte Daten, nicht einfach Texte oder Bilder.

- Meist werden letztendlich Platten dafür verwendet.
- Hatten Sie schon einmal einen Plattenschaden, bei dem alle Daten auf der Platte nicht mehr lesbar waren?

  A. Ich selbst hatte eine defekte Platte.

  B. Jemand aus meiner unmittelbaren Bekanntschaft.

  C. Nein. Bislang hatten ich und meine Bekannten Glück.

- Und wie ist es mit USB-Sticks?

# Aufgabe: Algorithmen im DBMS

## Aufgabe:

- Wenn Sie ein DBMS als Software-Bibliothek für Ihr Programm sehen, für welche Aufgaben bekommen Sie Algorithmen in der Bibliothek (die Sie sonst selbst programmieren müssten)?

  Tatsächlich ist ein DBMS mehr als eine Unterprogrammbibliothek, weil es z.B. auch als Kontrollinstanz die Zugriffsrechte verschiedener Nutzer verwaltet. Auch die deklarative Sprache zur Formulierung von Anfragen und den Anfrageoptimierer können Sie hier außer Acht lassen.

- _____
- _____
- _____
- _____

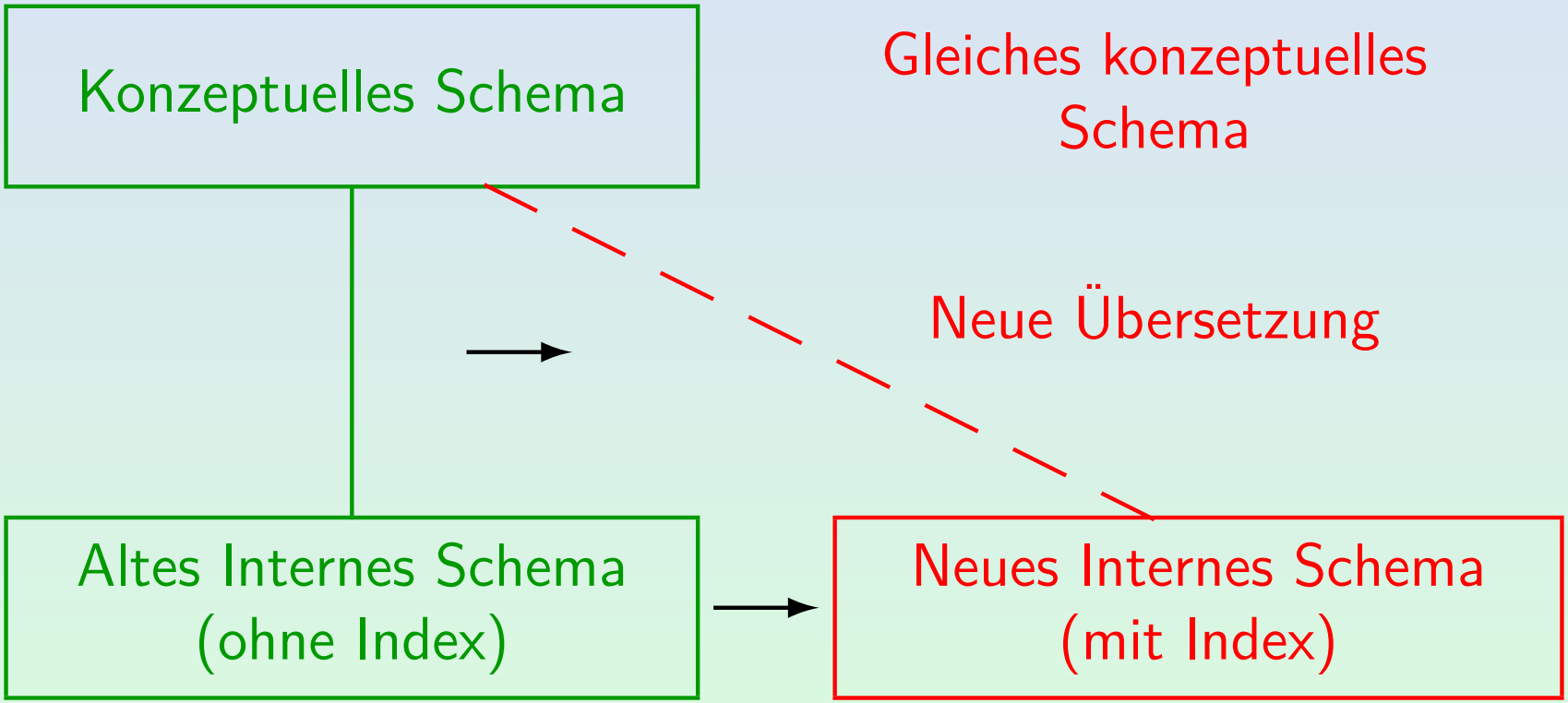
# Indexe

- Ein Index über einer Spalte einer Tabelle hilft u.a., Zeilen mit einem bekannten Wert in dieser Spalte schnell zu finden.
- Es ist also im wesentlichen eine Abbildung von Datenwerten auf physische Adressen von Tabellenzeilen.

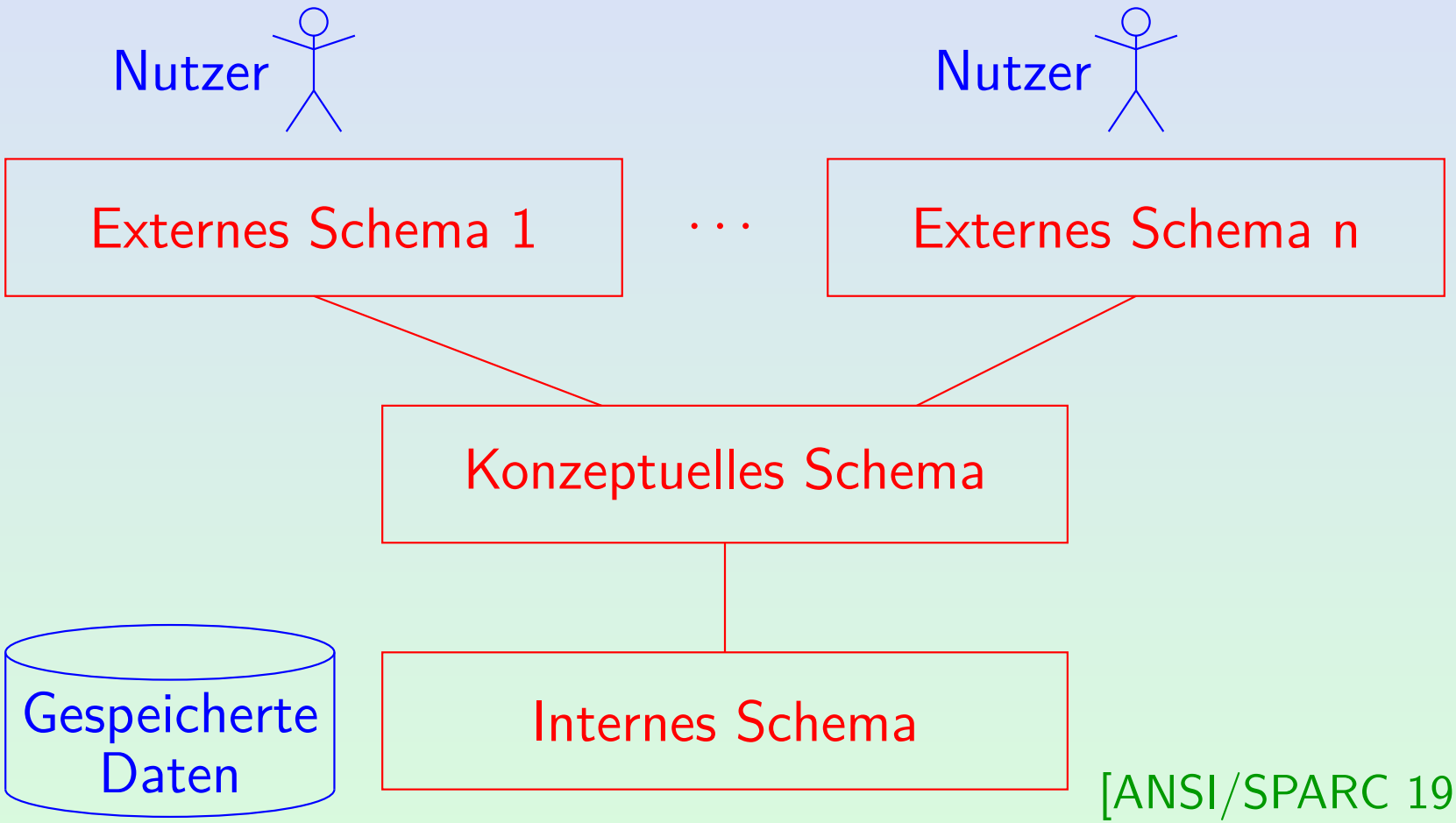
  Es kann auch mehr als eine Zeile zu einem Datenwert geben. Diese Abstraktion ist etwas vereinfacht, manche Indexe können auch andere Arten von Anfragen beschleunigen, z.B. mit < und >-Bedingungen.
- Kennen Sie binäre Suchbäume?

  Kennt jemand auch Probleme von binären Suchbäumen?
- Kennen Sie B-Bäume oder $B^+$-Bäume oder $B^*$-Bäume?
- Kennen Sie Hashtabellen?

# Physische Datenunabhängigkeit



# Drei-Schema Architektur




[ANSI/SPARC 1978]

# Beispiel-Datenbank: Konzeptuelles Schema

**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 |

# Beispiele zur Drei-Schema-Architektur

Aufgabe:

- Für welche Nutzer könnte es z.B. eigene externe Schemata geben? Wie würden diese aussehen?
- Welche nachträglichen Erweiterungen an den Tabellen könnten Sie sich vorstellen (z.B. eine zusätzliche Spalte)?

  Externe Schemata für eine Anwendung würden helfen, dass die Anwendung nach so einer Erweiterung unverändert weiterläuft. Mit etwas Disziplin bei der Formulierung der SQL-Anweisungen ist für zusätzliche Spalten aber noch kein externes Schema nötig.
- Geben Sie ein Beispiel für eine Spalte, bei der das interne Schema einen Index enthalten sollte.

# Hinweis zur Drei-Schema-Architektur

- Die Drei-Schema-Architektur der ANSI/SPARC Arbeitsgruppe ist ein theoretisches Modell.
- Wenn man z.B. in PostgreSQL eine relationale Datenbank anlegt, definiert man das konzeptuelle Schema und kann darauf natürlich auch zugreifen.
- Wenn man will, kann man weitere Schemata mit virtuellen Tabellen („Views“, „Sichten“) anlegen, und anderen Nutzern nur auf ein solches Schema Zugriff geben.

  Man muss das aber nicht so strukturieren. Die virtuellen Tabellen können auch im gleichen Schema angelegt werden wie die ursprünglichen Tabellen.
- Das interne Schema wird nicht explizit angelegt und nicht ganz strikt getrennt. Z.B. gibt man zusätzliche Speicherparameter beim `CREATE TABLE` Befehl mit an.

# DBMS-Funktionen am Beispiel

Aufgabe:

- Angenommen, Sie sollen ein System zur Evaluation der Lehre an dieser Universität entwickeln:
  Studierende stimmen über die Vorlesungs-Qualität ab.

  Es gibt ein Formular im Internet, in das Studierende ihre Daten eingeben können. Diese werden auf dem Web-Server abgespeichert.
  Später werden die gesammelten Daten ausgewertet,
  z.B. Durchschnittswerte berechnet.

- Vorschlag: Die Daten werden in einer Datei gespeichert (z.B. im CSV-Format).

- Welche Argumente gibt es, stattdessen ein DBMS zu verwenden?

# Nachteile von DBMS

- Was könnten mögliche Nachteile der Verwendung eines DBMS gegenüber einfachen Dateien sein?
  - _____
  - _____
  - _____
- In welchen Situationen lohnt sich die Verwendung eines DBMS eher nicht? D.h. wann kann es seine Vorteile nicht ausspielen?
  - _____
  - _____
  - _____

# Deklarative Anfragesprache am Beispiel (1)

- 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.

# Deklarative Anfragesprache am Beispiel (2)

- Wir hatten das letztes Jahr als Bonusaufgabe.
- Die Studierenden mit voller Punktzahl hatten zwischen 20 min und 180 min angegeben, Durchschnittswert war 80 min (Median: 60 min).

  Ich habe es auch gelöst und habe 67 min gebraucht, kannte aber schon die Lösungen von Studierenden.
- Die Lösungen hatten zwischen 34 und 181 Zeilen (Durchschnitt: 82) (Muster-Lösung: 78).
- Reine Codezeilen waren zwischen 28 und 139 (Durchschnitt: 64) (Muster-Lösung: 59).
- Man kann die Aufgabe z.B. mit einer `TreeMap` lösen, die Studierenden-Namen auf Punkte abbildet.

# 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

# Open-Source Datenbanken

- Sie haben schon verwendet (und es wurden in der Vorlesung besprochen):
  - PostgreSQL
  - MySQL: Sehr verbreitet, z.B. in Web Hosting Paketen.
- Es gibt inzwischen viele weitere Open-Source Datenbank-Managementsysteme:
  - MariaDB (von MySQL abgespalten)
  - SQLite: Wird in vielen Programmen verwendet, z.B. Firefox.

    [https://stackoverflow.com/questions/7610896/]
  - HSQLDB: Einfach in Java-Programme zu integrieren.

# Kommerzielles DBMS: Oracle

- Verbreitetes kommerzielles System. Vermutlich Marktführer.

  Ich persönlich habe mit Oracle gearbeitet seit meinem Studium.

  Es ist das System, was ich am besten kenne. Recht gut standard-kompatibel.

  Oracle war die erste kommerzielle SQL-Datenbank überhaupt (1979).

- Die „Express Edition“ kostet nichts:
  [https://www.oracle.com/database/technologies/appdev/xe.html]

  Gibt es für Windows und Linux. Bei Linux funktioniert jedenfalls die Installation unter CentOS problemlos (ähnlich zu Red Hat Linux).

  Maximal 4 GB/11 GB Datenbank, 1 Kern, 1 GB RAM, u.a. Einschränkungen.

- Oracle ist auch Anbieter von betriebswirtschaftlicher Standard-Software (Konkurrent zu SAP).

  Durch viele Zukäufe, u.a. Siebel Systems (CRM), PeopleSoft, JD Edwards.

  Die Datenbank war aber die Basis der Firma. Jetzt auch Cloud Anbieter.

# Kommerzielles DBMS: Microsoft SQL Server

- Verbreitetes kommerzielles System, gut standard-kompatibel
- Ursprünglich war es das Sybase-Datenbanksystem, für das Betriebssystem OS/2 (Microsoft/IBM) portiert.

  Die erste Version erschien 1989. Die Kooperation mit Sybase lockerte sich ab ca. 1993, inzwischen ist Microsoft SQL Server ein eigenständiges System. Es war lange Zeit nur für Microsoft Betriebssysteme erhältlich, seit 2017 auch für Linux.
- SQL Server 2019 Express kostet nichts.

  [https://www.microsoft.com/de-de/sql-server/sql-server-downloads]
  [https://docs.microsoft.com/de-de/sql/sql-server/editions-and-components-of-sql-server-version-15]

  Bei der Express-Edition ist die Datenbankgröße begrenzt auf 10 GB, die Puffergröße im Hauptspeicher auf 1.4 GB, maximal 4 Kerne, keine Hoch-Verfügbarkeit (Failover), und andere Einschränkungen.

# Kommerzielles DBMS: IBM DB2

- Weiteres wichtiges kommerzielles DBMS.

  Edgar F. Codd, der Erfinder des relationalen Datenbankmodells, arbeite im IBM Forschungslabor in San Jose, als der diese Ideen entwickelt hat. Sein Artikel „A Relational Model of Data for Large Shared Data Banks“ erschien 1970 (in den Communications der ACM). 1981 bekam er den Turing Preis. SEQUEL, eine frühere Version von SQL, wurde von Chamberlin, Boyce et al. 1974 im gleichen Forschungslabor entwickelt. System/R, einer der beiden ersten Forschungs-Prototypen einer relationalen Datenbank, wurde dort 1976/77 entwickelt. Der andere war Ingres von Michael Stonebraker (Berkeley).

- Kostenlose Express-C Version (läuft aus?)

  [https://www-01.ibm.com/marketing/iwm/iwm/web/pickUrxNew.do?source=swg-db2expressc]

  Für Entwickler gibt es diese neueren Versionen, die zum Test kostenlos sind (?):
  [https://www.ibm.com/us-en/marketplace/ibm-db2-direct-and-developer-editions/purchase]

# Installations-Erfahrungen (1)

- Sie können in dieser Vorlesung mit dem Adminer-Webinterface arbeiten. Sie müssen keine Software installieren.

  Außer später den Oracle SQL Developer Data Modeler oder ein anderes Zeichenprogramm (für 1–2 Hausaufgaben zum ER-Modell).

- Allerdings hat man auch Administrator-Rechte und kann mehr ausprobieren, wenn man einen eigenen DB-Server hat.

- Haben Sie PostgreSQL schon auf Ihrem Rechner installiert?

  A. Ja, direkt.

  B. Ja, über Docker,

  C. Habe es versucht, hat aber nicht geklappt.

  D. Nein.

# Installations-Erfahrungen (2)

- Haben Sie eine andere Datenbank auf Ihrem Rechner?
  A. MySQL oder MariaDB
  B. Microsoft Access
  C. Microsoft SQL Server
  D. Anderes DBMS
  E. Nein.
- Weiteres Web-Interface (sogar mit verschiedenen DBMS):
  [http://sqlfiddle.com/]

  Links geben Sie `CREATE TABLE`-Befehle und `INSERT`-Anweisungen ein, z.B. [https://users.informatik.uni-halle.de/~brass/db21/sql/db0.sql]. Dann klicken Sie auf „Build Schema“. Anschließend geben Sie rechts Ihre Anfrage ein und klicken auf „Run Query“.

# Inhalt

# Präsenzaufgabe: DBMS Daten und Webseiten

- Geben Sie eine Liste mit drei bekannten DBMS ab:
  - Name
  - Webadresse der Homepage des Projekts
  - Webadresse der Download-Seite (falls möglich)
  - Webadresse der offiziellen Dokumentation (falls möglich)
  - Webadresse des Wikipedia-Artikels
  - Aktuelle Versions-Nummer
  - Jahr des Projekt-Starts (soweit möglich, ggf. ungefähr).
- Sie müssen als Gruppen aus den „Breakout-Rooms“ abgeben.

  Bitte in der Übungsplattform in StudIP abgeben (als Gruppe). Es reicht eine ASCII `.txt` Datei. Sie z.B. über die „geteilten Notizen“ kommunizieren. Oder [https://sharelatex.informatik.uni-halle.de/].