Programmierung von Datenbanken - SS2022

Name:	Programmierung von Datenbanken - SS2022
Kursbereiche:	SoSe 22
Kurskontakt:	Patrick Bonack, Emmanuela Georgoula, Lanouar Dominik Jaouani, Robert Macu, Anastasia Navodkina, Chantal Schneevoigt, Andreas Scholl, Margarita Syrtlanova, Oliver Theobald, Karsten Tolle, Adrian Welcker

Vorlesung 1

Teil:PDB-1

Dr. Karsten Tolle – PDB – SS 2022

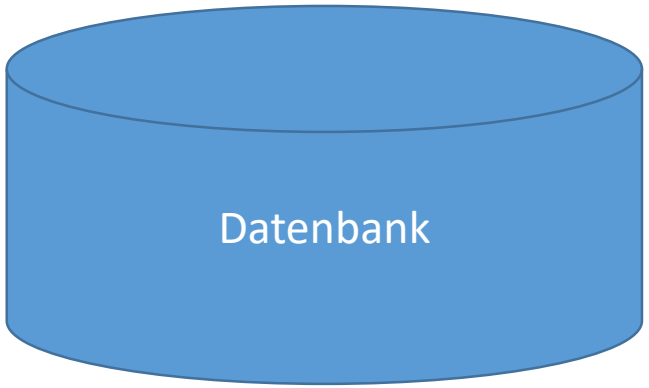
Inhalt heute …

- Motivation und Begriffe
- Tools und Links
- SQL (survival pack)
  - create database (datenbank anlegen)
  - create table (Tabelle erzeugen)
  - insert into (Einfügen)
  - select (Anfragen)
    - Struktur
    - where-Bedingungen

Datenbank

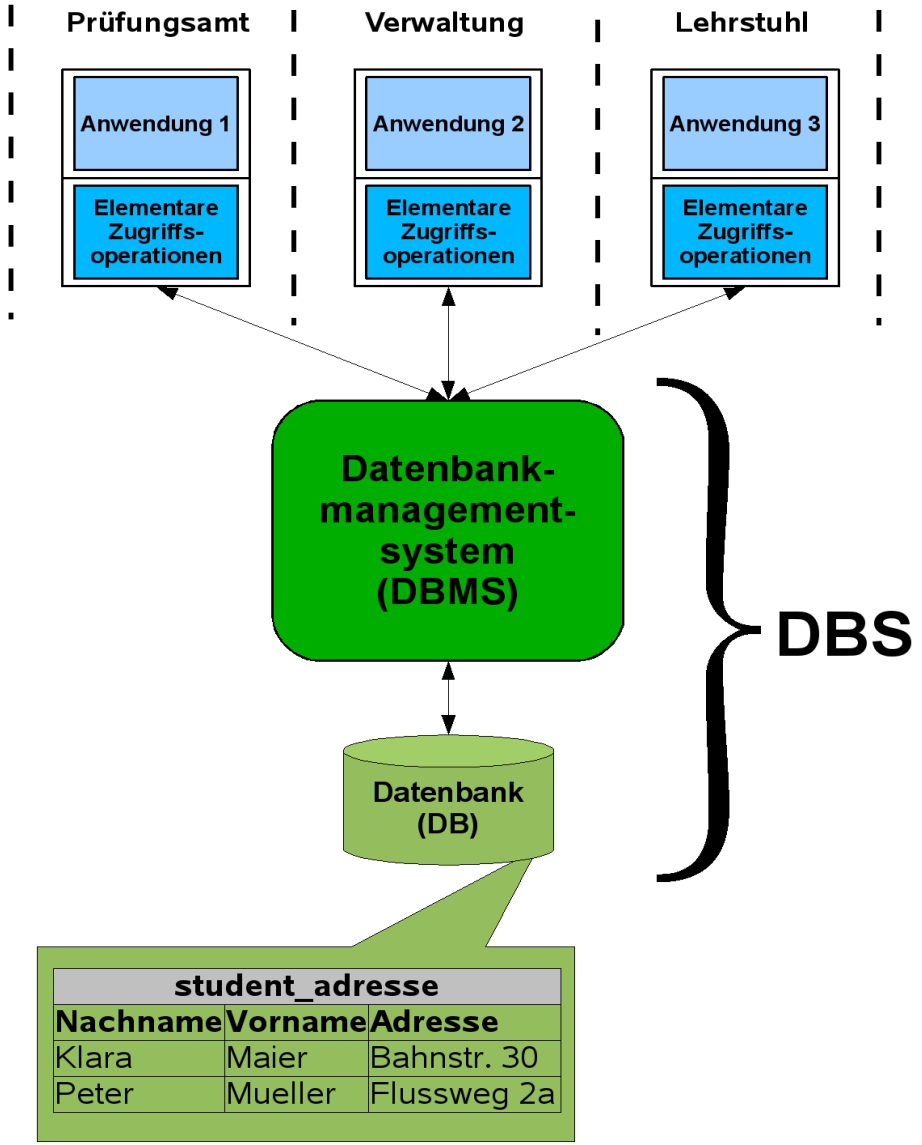
Definition (Duden):

Elektronisches System, in dem große Mengen an Daten zentral gespeichert werden können.



Was ist der Unterschied einer Datenbank zu einer Festplatte?

- Datenbank-management-system (DBMS)
- Datenbank (DB)
- Datenbanksystem (DBS)

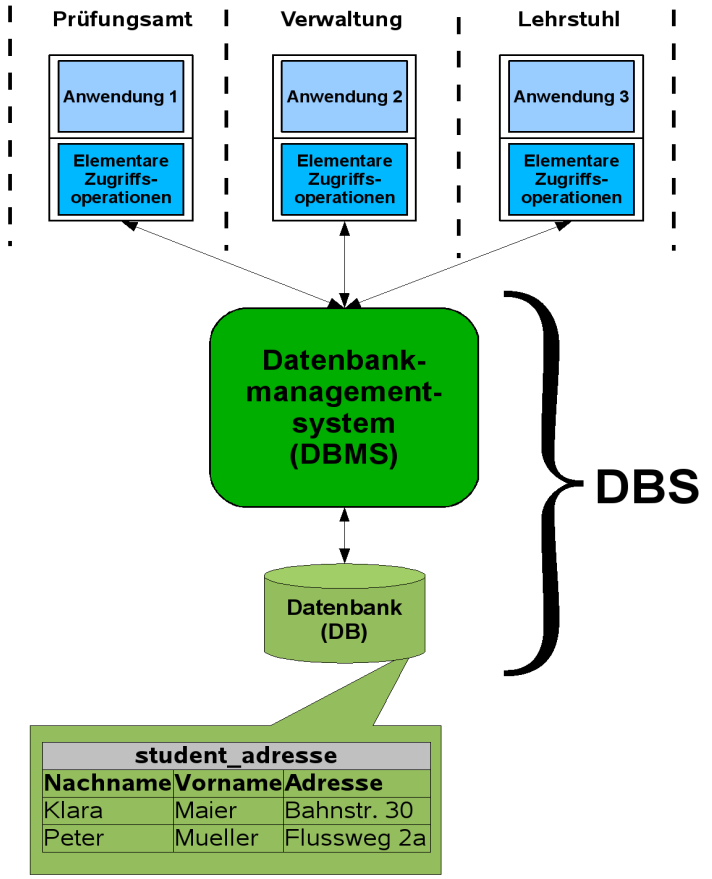


Fragen um herauszufinden, ob man ein DBMS benötigt:

1. Liegt eine **große Datenmenge** vor, die schwer zu managen ist?
   - Braucht es lange das Dokument zu öffnen?
   - Ist das Betrachten der Daten schwierig, muss man viel scrollen und ist es schwer die gesuchten Daten zu finden?
2. Arbeiten **verschiedene Personen/Anwendungen** mit den Daten?
3. Gibt es weitere Daten, die mit den gegebenen Daten in Relation stehen und ebenfalls gespeichert werden?
   - Ziehen Änderungen an einer Stellen Änderungen an anderen Stellen nach sich?
4. Werden die gleichen Daten an **unterschiedlichen Orten** verwendet?

Vorteile DBS

- **Redundanz und Inkonsistenz**
  Können durch die zentrale Datenverwaltung und Datenhaltung vermieden werden.
- **Sicherheit gegen Datenmissbrauch**
  Durch die zentrale Benutzerverwaltung können Zugriffsrechte gut kontrolliert werden.
- **Datenkonsistenz auch bei Ausfall**
  Durch Recovery-Strategien kann sicher gestellt, dass auch nach einem unerwartetem Ausfall die Daten konsistent bleiben (Transaktionen).
- … und weitere!



Erstellung einer Datenbank: Erster Schritt …

- Was sind die Anforderungen?
- Was sind die Ziele?
- Ist bekannt was gespeichert werden soll?

→ **Design des *Datenmodels***

Erstellung einer Datenbank: Zweiter Schritt …

- Welches DBMS wird genutzt?
- Wer soll wie auf die Daten zugreifen?

→ **Umsetzung des *Datenmodels im DBMS***

- Wir verwenden in PDB:
  - **relationales** DBMS – insb. MariaDB
  - Zugriff über die Anfragesprache **SQL**

Relationales Datenbankmodell

E.F. Codd, 1970 (Grundbegriffe)

- **Tabellen** mit Zeilen und Spalten um die Daten darzustellen.

Attribute

employee

EMPNO	FIRSTNME	LASTNME	PHONENO	SALARY
001	Jon	Lucas	2983	2000
003	Jon	Smith	2980	3588
103	Lucas	Jon	4444	3980
999	Jon	Smith	3987	1500

Tupel

Schema bzw. Relationenschema:

employee (EMPNO, FIRSTNME, LASTNME, PHONENO, SALARY)

Downloads

- **MariaDB**: https://mariadb.org/

  Für die Übungsblätter gilt als Grundlage Version 10.6.7:

  https://mariadb.com/de/downloads/

  für Mac User entweder zurück zu Version 10.1.44

  https://mariadb.org/download/?t=mariadb&o=true&p=mariadb&r=10.1.44&os=macosx&cpu=x86_64

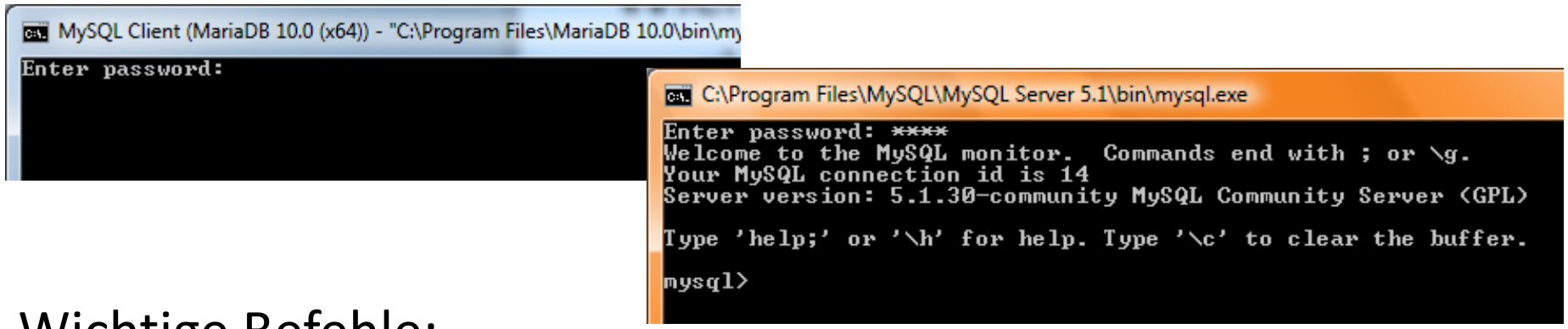
  oder (BESSER) über Homebrew (siehe Video „ Installation von MariaDB unter macOS via Homebrew“ im Moodle-Kurs) … dort sogar Verision 10.7.3 ☺

- (alternativ) MySQL: https://dev.mysql.com/downloads/mysql/ Community Server

  Workbench extra
  https://dev.mysql.com/downloads/workbench/

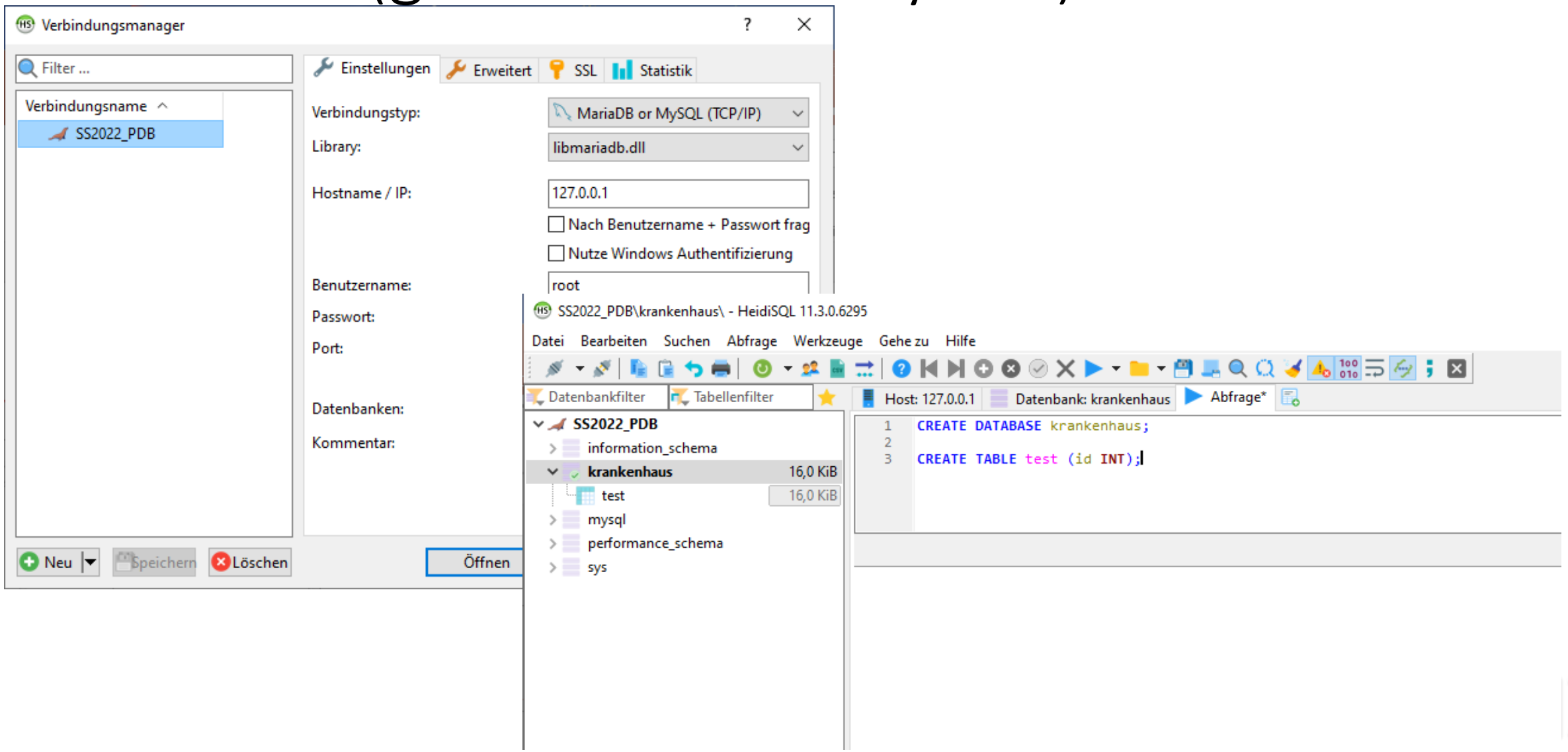
  (oder als Packet)

Command Line Client

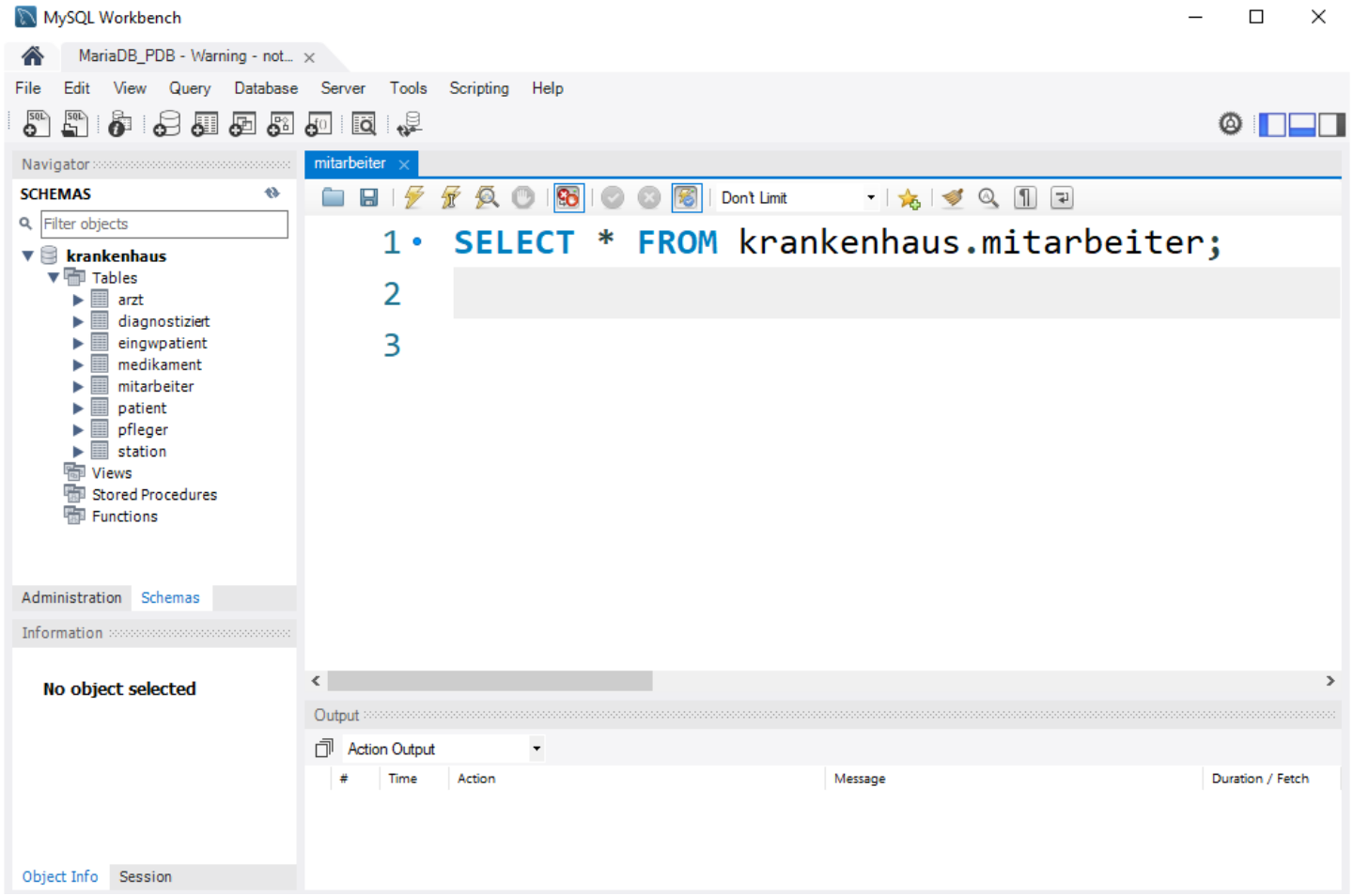


- Wichtige Befehle:
  - show databases; -- zeigt die Datenbanken an
  - **create database <database_name>; -- erzeugt eine leere DB**
  - drop database <database_name>; -- löscht eine DB
  - use <*database_name*>; -- erzeugt eine Verbindung zur Datenbank
  - show tables; -- zeigt die Tabellen der Datenbank an
  - explain <*table_name*>; -- gibt Informationen über die Tabelle
  - show variables; -- zeigt die aktuellen Einstellungen an

HeidiSQL (geht auch mit MySQL)

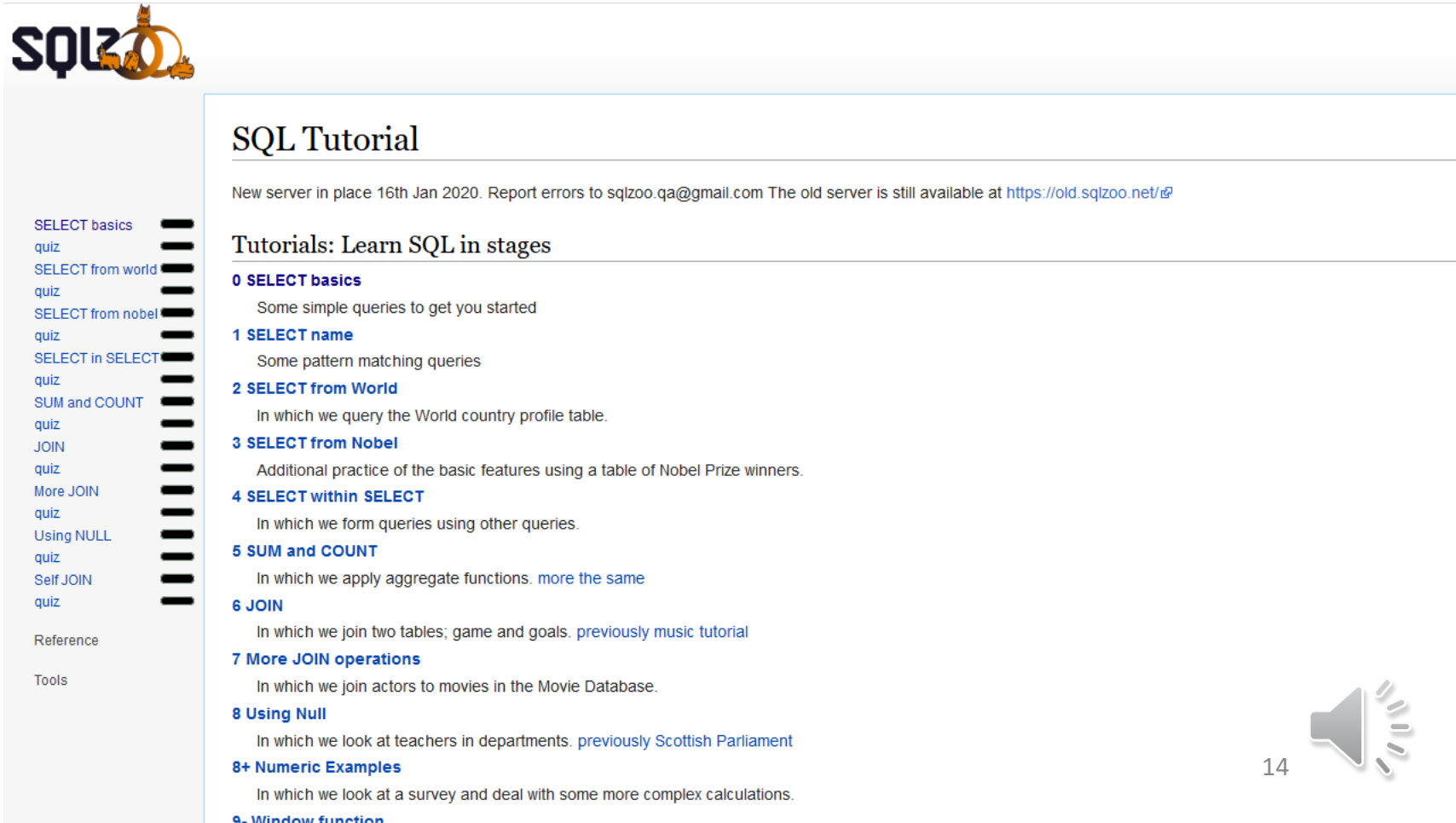


MySQL Workbench (geht auch mit MariaDB)



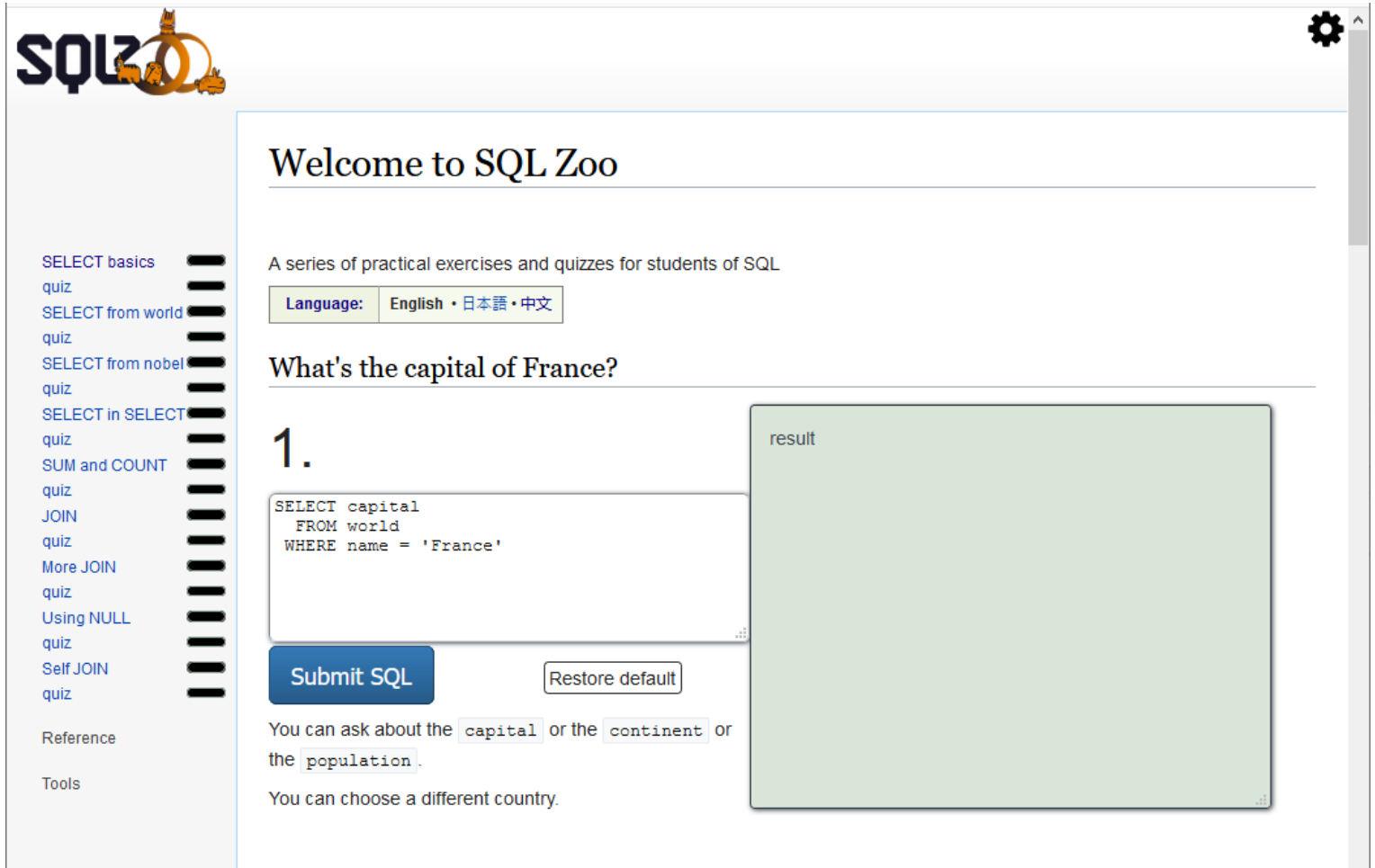
SQL-Online-Tutorial

- http://sqlzoo.net/



SQL Online Tutorial

- Anfragen können interaktiv ausgeführt werden.



SQL – Web Links

- **SQL Online ausprobieren:** http://www.w3schools.com/sql/

  und http://sqlzoo.net/

- **Mobile Apps, z.B. Learn SQL**

  https://play.google.com/store/apps/details?id=com.sololearn.sql&hl=de

  https://itunes.apple.com/de/app/learn-sql/id953775305?mt=8

- **Literatur – eBooks allgemein:** http://www.ub.uni-frankfurt.de/datenbanken/ebooks_gesamt.html

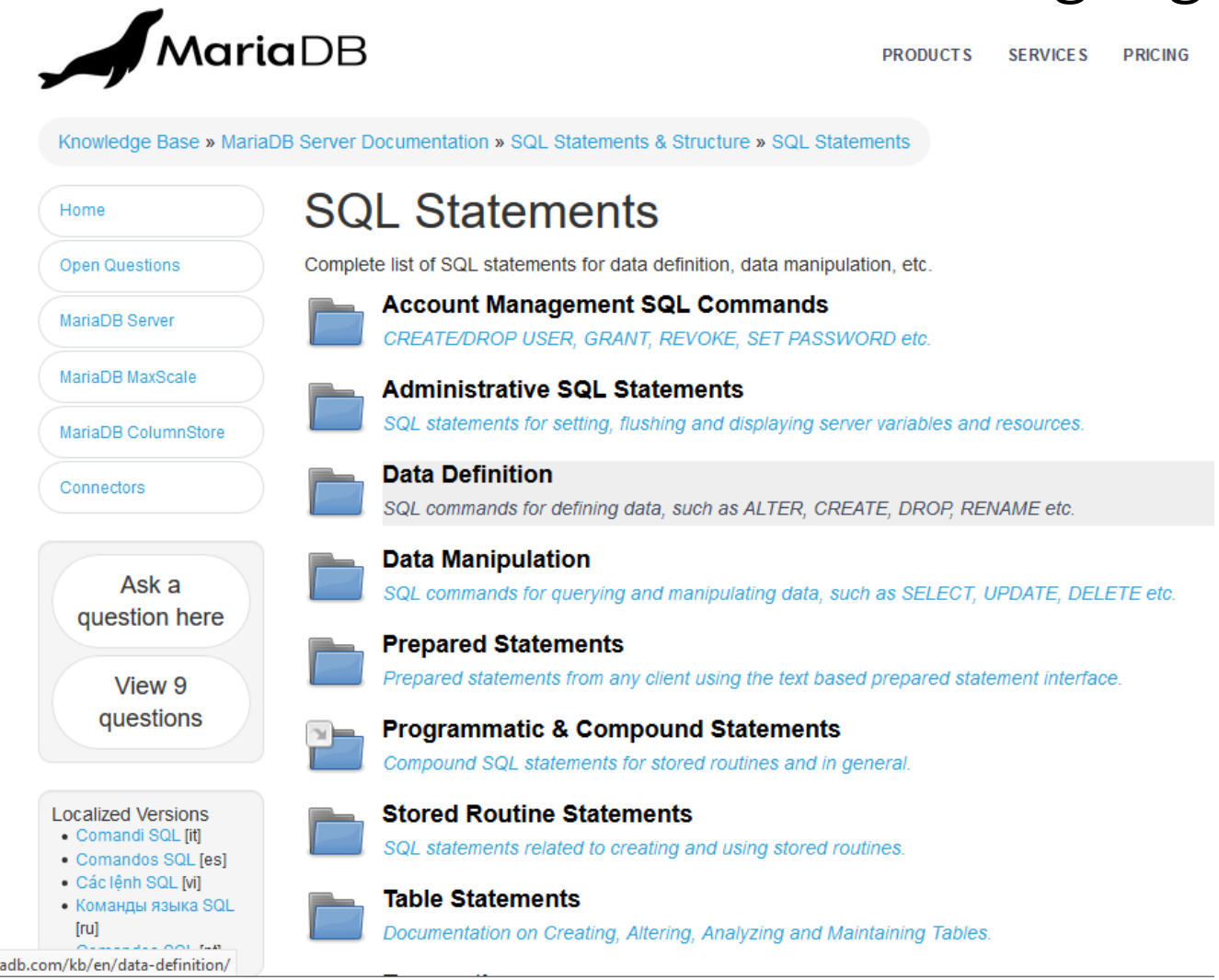
Structured Query Language

- SQL ist für Relationale Datenbanksysteme!

Standards:

- SQL-1 von 1986 bzw. 1989 (ca. 120 Seiten)
- SQL-2 (SQL92) von 1992 (ca. 580 Seiten)
  http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt
- SQL-3 (SQL99) von 2000 (ca. 1200 Seiten)
- SQL 2003 - ISO/IEC 9075:2003
- SQL:2006 - ISO/IEC 9075-14:2006 (SQL/XML)
- …

Data Definition Languge (DDL) und Data Manipulaiton Lanuguage (DML)



https://mariadb.com/kb/en/sql-statements/

Datenbank anlegen …

create database <name>;

alternativ: create schema <name>;

create database krankenhaus;

create schema krankenhaus;

Tabellen erstellen

Eine Tabelle wird im Minimalfall mit ihrem eindeutigen Namen sowie der Liste der zugehörigen Attribute samt Domänen nach folgendem Schema definiert:

create table *Relations*-N*ame* (
Attribut-Name Domäne { , *Attribut-Name Domäne*}*
) ;

z.B.: create table test (id int);

```
create table kunde (
        name varchar(30),
        vorname varchar(20),
        strasse varchar(50),
        stadt varchar(25),
        kinder int,
        gebdatum date
  ) ;
```

CREATE TABLE

Syntax

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...) [table_options    ]... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)] [table_options    ]... [partition_options]
    select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
   { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
    [IGNORE | REPLACE] [AS] SELECT ...   (Some legal select statement)
```

https://mariadb.com/kb/en/create-table/

die wichtigsten-SQL-Datentypen (Domänen)

- integer/int
- **double(m,d)**
- **float(m,d)**
- **decimal(m,d)**

- char(*n*)
- varchar(n)
- text

- date
- time
- datetime
- timestamp

- clob(n)
- blob(n)
- …

Siehe auch: https://mariadb.com/kb/en/data-types/

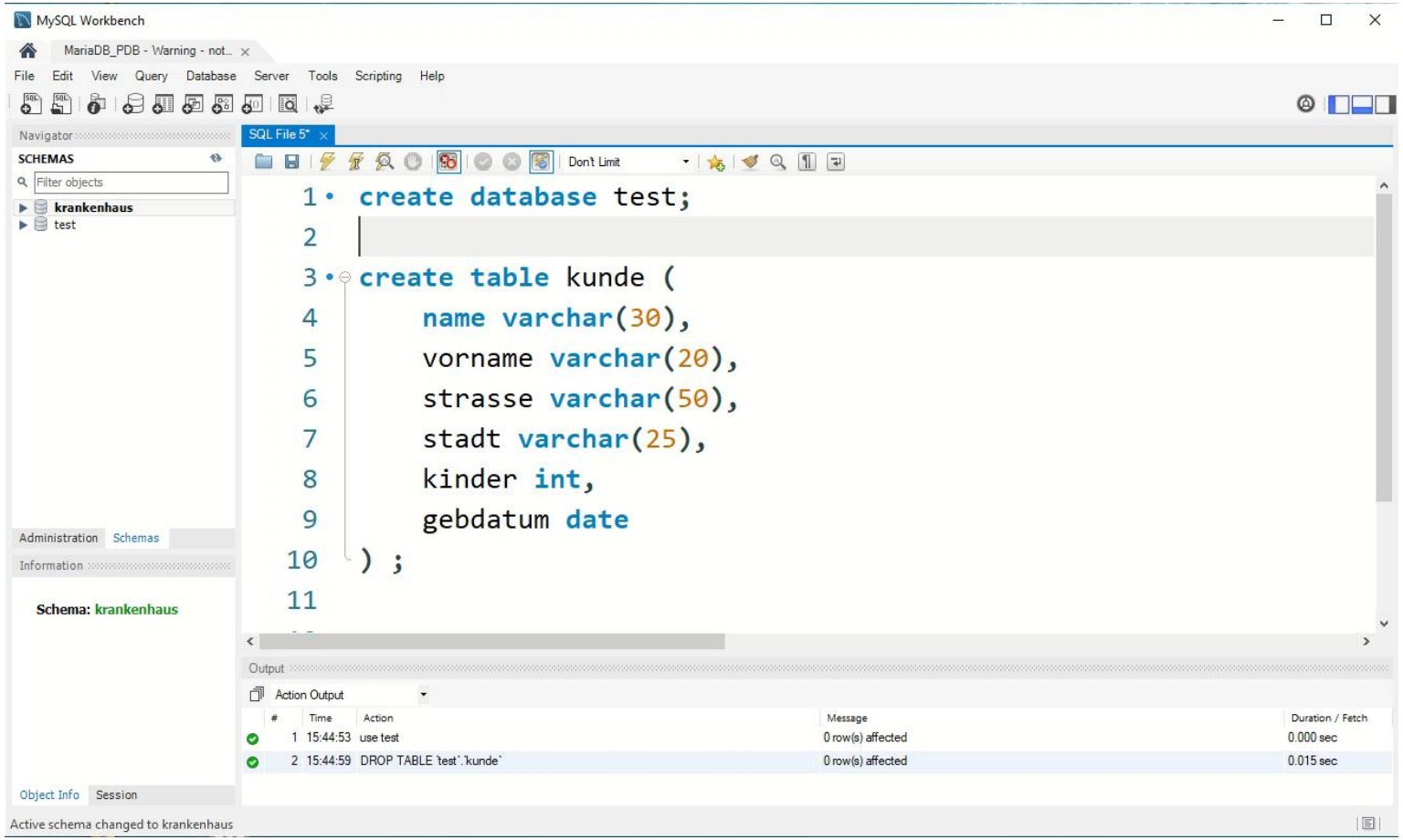
Einfügen von Tupeln I

Um Daten einzufügen, spezifiziert man das Tupel, welches eingefügt werden soll …

Die Werte für die Attribute der Tupel müssen aus der Domäne (Definitionsbereich) der Attribute sein.

```
insert into kunde
values
('Otto', 'Hans', 'Bäckerweg 12', 'Frankfurt',
3, '1970-12-01') ;
```

→ Der Kunde "Hans Otto" wird eingefügt.


MySQL Workbench
MariaDB_PDB - Warning - not...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
krankenhaus
test
Administration Schemas
Information
Schema: krankenhaus
Object Info Session
SQL File 5*
Don't Limit
1 • create database test;
2
3 • create table kunde (
4 name varchar(30),
5 vorname varchar(20),
6 strasse varchar(50),
7 stadt varchar(25),
8 kinder int,
9 gebdatum date
10) ;
11
Output
Action Output
Time Action Message Duration / Fetch
1 15:44:53 use test 0 row(s) affected 0.000 sec
2 15:44:59 DROP TABLE `test`.`kunde` 0 row(s) affected 0.015 sec
Active schema changed to krankenhaus

Einfügen von Tupeln II

... unbekannte Werte können mit **NULL-Values** befüllt werden:

```
insert into kunde
values
('Otto', 'Hans', null, 'Frankfurt', 3, '1970-12-01') ;
```

→ Der Kunde "Hans Otto" wird eingefügt.

Einfügen von Tupeln III

... alternativ kann man die Attribute angeben, die gesetzt werden sollen:

insert into kunde (vorname, name, gebdatum)
values ('Hans', 'Otto', '1970-12-01') ;

→ Der Kunde "Hans Otto" wird eingefügt.

… welche Version ist besser?

1. **insert into** kunde
   values ('Otto', 'Hans', null, null, null, '1970-12-01') ;
2. **insert into** kunde (vorname, name, gebdatum)
   values ('Hans', 'Otto', '1970-12-01') ;

Einfügen von Tupeln IV

... es können auch mehrere Datensätze mit einem SQL-Statement eingefügt werden:

insert into kunde (vorname, name, gebdatum)
values

('Hans', 'Otto', '1970-12-01'),

('Hans', 'Otto', '1925-11-24'),

('Hans', 'Otto', '2012-12-24') ;

→ Der Kunde "Hans Otto" wird 3-mal eingefügt.

Anfragen ohne Bedingungen

kunde (name, vorname, strasse, stadt)

select name, vorname **from** kunde;

select vorname, name **from** kunde;

select stadt **from** kunde;

select * **from** kunde;

Nicht in Programmen!

SQL verwirklicht das Prinzip der „Vielfachmenge“ (engl. multiset). In den Ergebnismengen können demnach Duplikate auftreten.

Sind keine Duplikate erwünscht, müssen sie explizit durch den Zusatz **distinct** entfernt werden.

select distinct stadt **from** kunde;

where-Klausel

Bezüglich der Bedingung sind Vergleiche mit den üblichen Operatoren, den logischen Verknüpfungen **and** und **or** sowie Klammerungen gestattet.

kunde (name, vorname, strasse, stadt, kinder)

```
select * from kunde
where kinder > 0 or stadt = 'Frankfurt' and
name = 'Otto' ;
```

kunde (name, vorname, strasse, stadt, kinder)

```
select * from kunde
where kinder > 0 or stadt = 'Frankfurt' and
name = 'Otto' ;
```

```
select * from kunde
where kinder > 0 or (stadt = 'Frankfurt' and
name = 'Otto');
```

Ausführungsreihenfolge: erst AND dann OR … aber besser lesbar mit Klammern!