

Datenbanken 1

Sommersemester 2017

Übung 1

Übersicht

- Entwurf
 - Anforderungsanalyse -> Dokumentation der Anford.
 - Konzeptuelles Modell -> ER-Diagramm
 - Logisches Modell -> Relationales Datenmodell
 - Physisches Modell (DB-spezifisch) -> DB2, MySQL, Oracle, etc.
- Operativ
 - MySQL: Installation, Workbench
 - Rel. Anfragesprache -> SQL (Structured Query Language)

Anforderungsanalyse I

Für ein Krankenhaus gibt es folgende Anforderungen, die relevant für die Entwicklung einer Datenbanklösung sind:

1. Das Krankenhaus beschäftigt **Ärzte**. Diese haben einen Namen, eine Personalnummer, eine Adresse, ein Geschlecht, ein Geburtsdatum und ein Fachgebiet.
2. Ein Arzt betreut mehrere Patienten. Ein **Patient** hat einen Namen, ein Geschlecht, ein Geburtsdatum und eine Patientennummer.
3. Ein Patient, welcher eingewiesen wurde, wird genau einer **Station** zugeordnet.
4. Ein Arzt kann eine **Krankheit** bei einem Patienten diagnostizieren. Eine Krankheit hat einen Umgangsnamen und einen Lateinischen Fachnamen, sowie eine Liste von üblichen **Symptomen**. Über einen Status soll vermerkt werden, ob die Krankheit geheilt wurde oder diese noch vorliegt.
5. Ein Patient kann ein oder mehrere **Medikamente** durch einen Arzt verordnet bekommen. Ein Medikament hat einen Namen, einen Produzenten, einen Preis (pro Einheit) und einen Bestand. Zu jeder dieser Verordnungen ist eine Dosierung anzugeben. (Zur Vereinheitlichung werden Bestand und Dosierung in sogenannten „Einheiten“ angegeben – je nach Medikament kann dies eine Pille, gewisse Gramm Anzahl oder Milliliter bei Flüssigkeiten sein. Diese Umrechnung bzw. Problematik kann hier jedoch ignoriert werden.)
6. Einige Medikamente können einer oder mehreren Krankheiten zugeordnet werden.
7. Es gibt Medikamente mit denselben Wirkstoff-Zusammensetzungen, die von unterschiedlichen Produzenten hergestellt werden.
8. Krankenhäuser beschäftigen auch **Pfleger**. Dies haben einen Namen, eine Adresse, ein Geschlecht, ein Geburtsdatum, eine Station und eine Personalnummer.

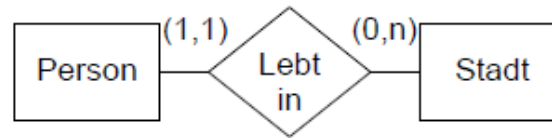
Anforderungsanalyse II

Ein Beispiel aus der Praxis: Insg. 1050 Anforderungen an eine neue Webanwendung. Daraus muss der Data-Architect Entitäten und Beziehungen extrahieren... viele Diskussionen mit Fachbereichen (die techn. Aspekte sind vergleichsweise uninteressant!)

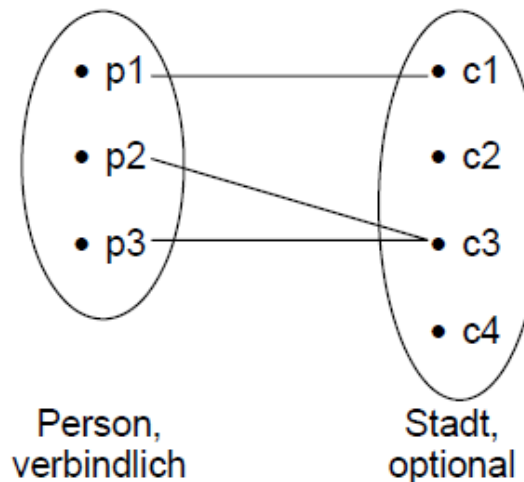
Req. ID*	Strategic pillar*	Functionality*	Block L1*	Block L2*	Use Case	Description*	Comment
R0075	Live Smarter	Portal Platform >> Foundation <<	Account preferences	User profile	profile setup	User must be able to create an user profile	Contact information have to be provided. Name, location, contact number, email... APS: Existing: APS other countries: high => We have to follow market standards
R0076	Live Smarter	Portal Platform >> Foundation <<	Account preferences	User profile	profile setup	User must be able to choose which information is displayed in his public profile	some information is per default public. E.g.: Name APS: Existing: APS other countries: high => We have to follow market standards
R0077	Live Smarter	Portal Platform >> Foundation <<	Account preferences	User profile	profile setup	User must be able to edit his/her existing user profile	including profile picture APS: Existing: APS other countries: high => We have to follow market standards
R0078	Live Smarter	Portal Platform >> Foundation <<	Account preferences	User profile	profile setup	User must be able to select which information he wants to share with contacts	hidden und unhidden information APS: Existing: APS other countries: high => We have to follow market standards
R0079	Live Smarter	Portal Platform >> Foundation <<	Account preferences	User profile	profile setup	User must be able to upload a photo of himself/herself	APA: It is important and if mobile device used APS: Existing: APS other countries: high => We have to follow market standards
R0805	Market Smarter	Certification - Audit Based	Certification	Audit	General	System must be able to create a report with compliance requirements to be met by a specific customer which can be send to him	APA: unable to understand APS: ??? FICD: Could be relevant for <Segment>
R0806	Market Smarter	Certification - Audit Based	Certification	Audit	General	User must be able to answer to the compliance requirements with messages, attached files and pictures.	APA: unable to understand APS: ??? FICD: Could be relevant for <Segment>
R0807	Market Smarter	Certification - Audit Based	Certification	Audit	Data exchange	System must be able to exchange data with the external farm management systems e.g. <company> audit system	APA: unable to understand APS: ??? FICD: Could be relevant for <Segment>
R1006	Grow smarter	Customer Navigator	Customer Search	Customer Search Criteria		User must be able to search for a customer from the customer navigator list view by a number of fields such as (name, address ...).	

→ In der Praxis ist es sinnvoll jeder Anford. eine ID zuzuweisen. Dies hilft bei der Identifizierung in Diskussionen, Dokumenten etc.

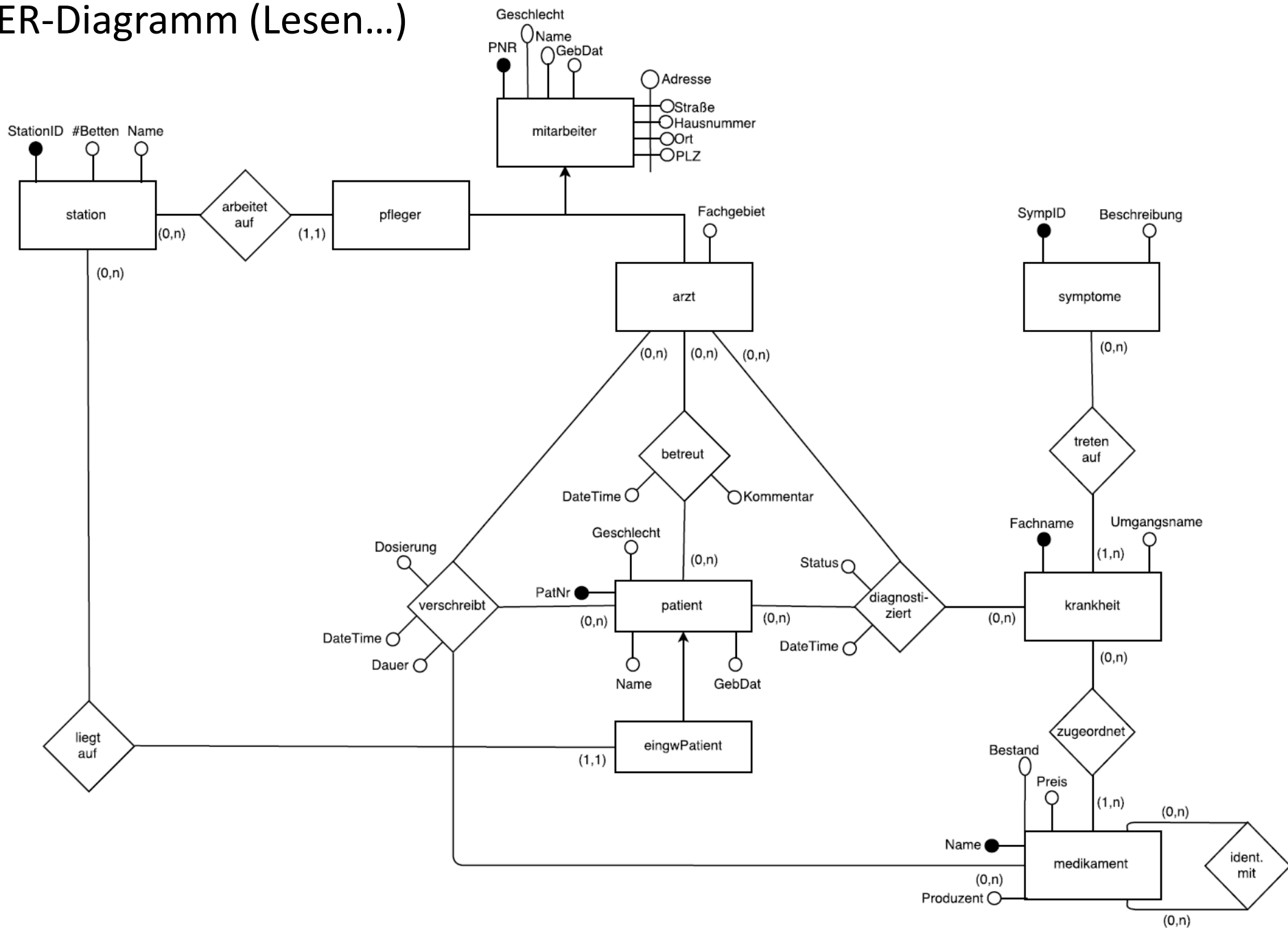
ER-Diagramm (Beispiel: Kardinalitäten)



- $\text{min_card}(\text{Person}, \text{Lebt_in}) = 1$
- $\text{max_card}(\text{Person}, \text{Lebt_in}) = 1$
- $\text{min_card}(\text{Stadt}, \text{Lebt_in}) = 0$
- $\text{max_card}(\text{Stadt}, \text{Lebt_in}) = n$



ER-Diagramm (Lesen...)



Relationales Datenmodell

E.F. Codd, 1970 (Grundbegriffe)

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

Employee

Attribute

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

Schema bzw. Relationenschema:

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

ER-Diagramm (Transform. in rel. Modell)

mitarbeiter (PNR, Geschlecht, Name, GebDat, Straße, Hausnummer, PLZ, Ort)

arzt (PNR, Fachgebiet)

FK PNR auf mitarbeiter:PNR

pfleger (PNR, StationID)

FK StationID auf station:StationID

Jeder Pfleger hat genau eine Station

FK PNR auf mitarbeiter:PNR

station (StationID, AnzahlBetten, Name)

patient (PatNr, Name, Geschlecht, GebDat)

eingwPatient (PatNr, StationID)

FK StationID auf station:StationID

Jeder Patient hat genau eine Station

FK PatNr auf patient:PatNr

betreut (PatNr, PNR, DateTime, Kommentar)

Datum ebenfalls im PK, damit ein Arzt einen Patienten öfter betreuen kann

FK PatNr auf patient:PatNr

FK PNR auf arzt:PNR

medikament (Name, Produzent, Preis, Bestand)

verschreibt (PatNr, PNR, Name, DateTime, Dauer, Dosierung)

FK PatNr auf patient:PatNr

FK PNR auf arzt:PNR

FK Name auf medikament:Name

ident_mit (Name_1, Name_2)

FK Name_1 auf medikament:Name

FK Name_2 auf medikament:Name

krankheit (Fachname, Umgangsname)

symptome (SympID, Beschreibung)

treten_auf (Fachname, SympID)

FK Fachname auf krankheit:Fachname

Jeder Fachname muss mind. 1x vorkommen

FK SympID auf symptome:SympID

diagnostiziert (Fachname, PatNr, PNR, DateTime, StatusID)

FK Fachname auf krankheit:Fachname

FK PatNr auf patient:PatNr

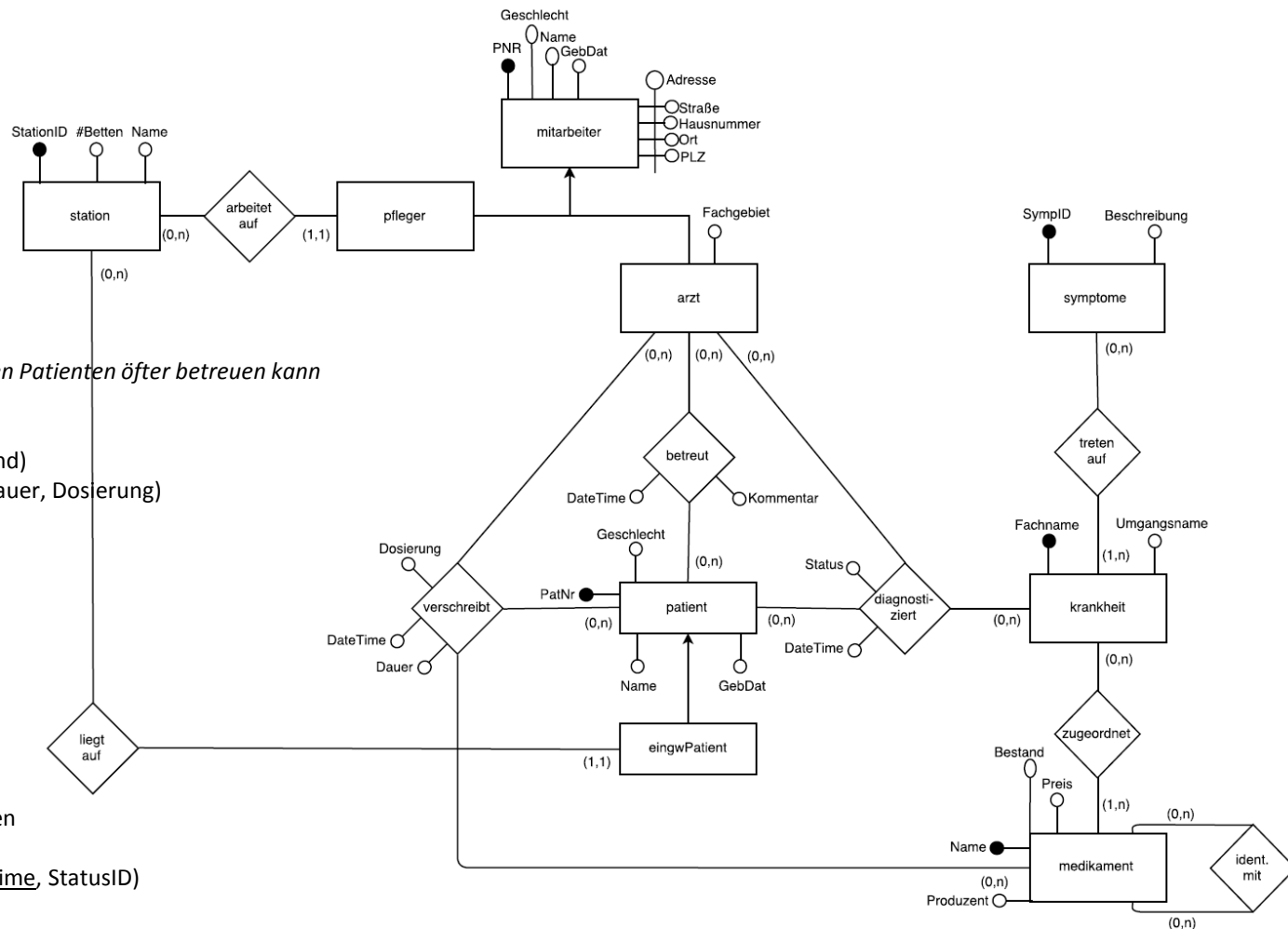
FK PNR auf arzt:PNR

zugeordnet (Fachname, Name)

FK Fachname auf krankheit:Fachname

FK Name auf medikament:Name

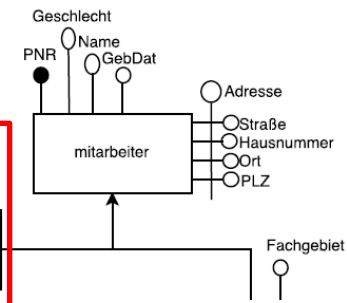
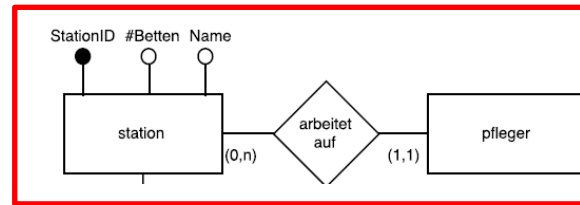
Jedes Medikament muss mind. 1x vorkommen



orange = n:n-Beziehung (führt zu eigener Relation)

blau = Entität

Ausgedrückt in SQL (Auszug):



Entitäts-Tabelle:

```
CREATE TABLE krankenhaus.pfleger (PNR INT NOT NULL, StationID INT NOT NULL, PRIMARY KEY (PNR));
```

1:n-Mapping-Tabelle:

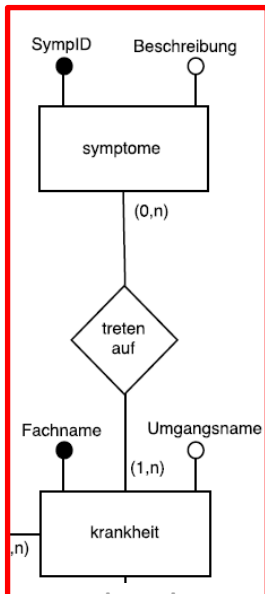
Nicht nötig! Bei 0:n schon oder NULL-Werte sind möglich.

Entitäts-Tabelle:

```
CREATE TABLE krankenhaus.station (StationID INT NOT NULL AUTO_INCREMENT, Name VARCHAR(45), AnzahlBetten INT, PRIMARY KEY (StationID));
```

FK-Definition:

```
ALTER TABLE krankenhaus.pfleger ADD CONSTRAINT pfleger_mitarbeiter FOREIGN KEY (PNR) REFERENCES krankenhaus.mitarbeiter (PNR) ON DELETE NO ACTION ON UPDATE NO ACTION, ADD CONSTRAINT pfleger_station FOREIGN KEY (StationID) REFERENCES krankenhaus.station (StationID) ON DELETE NO ACTION ON UPDATE NO ACTION;
```



Entitäts-Tabelle:

```
CREATE TABLE krankenhaus.symptome (SympID INT NOT NULL AUTO_INCREMENT, Beschreibung TEXT, PRIMARY KEY (SympID));
```

n:n-Mapping-Tabelle:

```
CREATE TABLE krankenhaus.treten_auf (Fachname VARCHAR(100) NOT NULL, SympID INT NOT NULL, PRIMARY KEY (Fachname, SympID));
```

Entitäts-Tabelle:

```
CREATE TABLE krankenhaus.krankheit (Fachname VARCHAR(100) NOT NULL, Umgangname VARCHAR(100), PRIMARY KEY (Fachname));
```

FK-Definition:

```
ALTER TABLE krankenhaus.treten_auf ADD CONSTRAINT tretenAuf_symptome FOREIGN KEY (SympID) REFERENCES krankenhaus.symptome (SympID) ON DELETE NO ACTION ON UPDATE NO ACTION, ADD CONSTRAINT tretenAuf_krankheit FOREIGN KEY (Fachname) REFERENCES krankenhaus.krankheit (Fachname) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Praktischer Teil: MySQL Installation/Workbench

- MySQL Community Server:

<https://dev.mysql.com/downloads/mysql/>



- MySQL Workbench:

<https://dev.mysql.com/downloads/workbench/>

- MariaDB: <https://mariadb.org/>



... im RBI

URL: <http://www.rbi.informatik.uni-frankfurt.de/RBI/de/nachrichten/datenbank-server-fuer-datenbank-vorlesungen-und-praktika>

GOETHE UNIVERSITÄT FRANKFURT AM MAIN | Rechnerbetriebsgruppe Informatik

Website durchsuchen

nur im aktuellen Bereich

Startseite über uns Service FAQ Lehre FB12 Informatik Goethe-Uni Impressum

Startseite » Nachrichten » Datenbank Server für Datenbank-Vorlesungen und -Praktika

Datenbank Server für Datenbank-Vorlesungen und -Praktika

Unter dem Namen *dbprakt* bzw. dem Alias *dbserver* steht für Studierende zur Übung für die Datenbank-Vorlesungen und -Praktika ein Server mit installierter SQL-Datenbank (MariaDB) zur Verfügung.

Ein remote Zugriff auf das Datenbanksystem mittels eines `mysql`-Client ist möglich über alle PC-Pool Rechner der RBI.

Voraussetzung zur Nutzung der Datenbank ist neben einem RBI Benutzerkonto das Anlegen einer Datenbank und eines Datenbankadmin-Account für den Nutzer auf `dbprakt` bzw. `dbserver` durch die RBI.

Für die Installation der Datenbank oder weitere Informationen kontaktieren Sie bitte [Azadeh Djahedi](#) per E-Mail oder persönlich in Raum R011.

Der Zugriff per `mysql`-Client sieht wie folgt aus:

```
mysql -u benutzername -h dbserver.rbi.cs.uni-frankfurt.de -p datenbankname
```

Nachrichten

- Sicherheitsüberprüfung der Account-Passwörter durch die RBI
19.04.2017
- Dreamspark Premium wird zu Microsoft Imagine
08.09.2016
- DreamSpark Premium steht wieder zur Verfügung!
24.02.2016
- Datenbank Server für Datenbank-Vorlesungen und -Praktika
24.02.2016

© 2017 Johann Wolfgang Goethe-Universität Frankfurt
CONTACT: Johann Wolfgang Goethe-Universität Frankfurt am Main, Institut für Informatik (RBI), Robert-Mayer-Straße 11-15, 60325 Frankfurt am Main.
Phone: +49 69 798 28351, Fax: +49 69 798 28851, E-Mail: rbi@cs.uni-frankfurt.de

[Übersicht](#) [Barrierefreiheit](#) [Kontakt](#)

Powered by Plone & Python

MySQL Workbench (geht auch mit MariaDB)

The screenshot displays the MySQL Workbench interface with the following components:

- Navigator:** Shows the instance 'Local instance MySQL56' and the 'prg2' schema.
- Schemas:** Lists various schemas including 'afe', 'airport', 'bank', 'employees', 'geonames', 'laden', 'prg2', and 'tutor_student_blat'.
- Query Editor:** Contains the query: `SELECT * FROM prg2.student;`
- SQL Additions:** Displays the syntax for the `SELECT` statement.
- Result Set Filter:** Shows the columns: `MatrNum`, `TutorID`, `UebungConfirm`, `UebungID`, and `KlausurConfirm`.
- Output:** Shows the execution results in a table format.

Time	Action	Message	Duration / Fetch
1 15:45:18	SELECT * FROM prg2.student	364 row(s) returned	0.000 sec / 0.000 sec

Table: student

Columns:

- MatrNum**: bigint(20) PK
- TutorID**: int(11)
- UebungConfirm**: int(1)
- UebungID**: int(11)
- KlausurConfirm**: int(1)
- FirstName**: varchar(255)
- LastName**: varchar(255)

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-Name (  
    Attribut-Name Domäne { , Attribut-Name  
    Domäne }*  
);
```

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

Zunächst wird ein neues Schema („Datenbank“) über z.B. den Workbench angelegt (funktioniert über die Menüleiste).
In diesem Schema (darauf achten, dass auch wirklich dieses Schema ausgewählt ist) wird mit folgendem SQL-Befehl eine Tabelle angelegt:

```
CREATE TABLE kunde (  
    name VARCHAR(30) ,  
    vorname VARCHAR(20) ,  
    strasse VARCHAR(50) ,  
    stadt VARCHAR(25) ,  
    kinder INT ,  
    gebdatum DATE  
);
```

Einfügen von Tupeln

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

```
INSERT INTO kunde VALUES ('Schmidt', 'Elias',  
    'Hauptstrasse 12', 'Frankfurt', 3, '2017-04-  
14');
```

```
INSERT INTO kunde VALUES ('Müller', 'Marie',  
    'Bergstrasse 5', 'Frankfurt', 3, '2017-03-  
15');
```

→ Zwei (sehr junge...) Kunden werden eingefügt.

Anfragen

KUNDE(NAME, VORNAME, STRASSE, STADT)

```
SELECT name, vorname FROM kunde;
```

```
SELECT stadt FROM kunde;
```

```
SELECT * FROM kunde WHERE vorname  
LIKE 'Elias';
```

Mehr zu SQL in späteren Übungen...

Gutes Tutorial vorab:

<https://www.w3schools.com/sql/>