UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Software Entwicklung & Programmierung

Datenbanken

PALUNO
The Ruhr Institute for Software Technology

Disclaimer

Bilder und Texte der Veranstaltungsfolien und -unterlagen sowie das gesprochene Wort innerhalb der Veranstaltung und Lehr-Lern-Videos dienen allein dem Selbst- bzw. Gruppenstudium. Jede weiterführende Nutzung ist den Teilnehmenden der Moodle-Kurse untersagt, z.B. Verbreitung an andere Studierende, in sozialen Netzwerken, dem Internet!

Darüber hinaus ist ein studentischer Mitschnitt von Webkonferenzen im Rahmen der Lehre nicht erlaubt.

Zielsetzung

Am Ende dieser Präsenzstunde könnt Ihr:

- Erläutern was ein **Datenbanksystem** ist und welche Komponenten es umfasst
- die grundsätzlichen **SQL-Konzepte** verstehen und den Einsatz von SQL in den Bereichen **Datendefinition**, **Datenmanipulation** und **Datenkontrolle** erläutern
- Mittels SQL einfache und komplexe **Datenbankanfragen** selber erstellen

Agenda

1. **Datenbanksysteme**
2. Relationale Datenbanken
3. Einführung in SQL
4. Anwendungsbeispiel
5. SQL in Java
6. Objektrelationale Abbildung

Datenbanksysteme

- Die Menge an Daten, auf die ein Programm zugreift und die es generiert, muss verarbeitet werden
- Die einfachste Möglichkeit dafür ist die **elektronische Datenverarbeitung** mittels Datenbanksystemen
- Datenbanksysteme ermöglichen die dauerhafte, zentrale **Speicherung** von Daten und deren **Verwaltung**
- Das Speichern und Abfragen erfolgt mit einer speziellen Skriptsprache (meistens **SQL**)

Datenbanksysteme

- Ein Datenbanksystem besteht aus zwei Komponenten:
  - Datenbankmanagementsystem (DBMS)
  - Datenbank (DB)

- Das DBMS ist zuständig für die **Organisation** und **Strukturierung** der Daten sowie die Kontrolle der lesenden und schreibenden Zugriffe auf die Datenbank

- Die Datenbank ist ein logisch zusammenhängender **Datenbestand**, der effizient, widerspruchsfrei und permanent gespeichert ist

→ ermöglicht das Verwalten von großen Datenmengen

Datenbanksysteme

Datenbankmanagementsystem (DBMS)

- Das DBMS ist die Systemsoftware eines Datenbanksystems
- SQL gilt als **Standarddatenbanksprache,** mit der man auf die Daten des DBMS zugreifen und diese verändern kann
- Beispiele für Datenbankmanagementsysteme:
  - H2
  - MariaDB
  - MySQL
  - Etc.

Datenbanksysteme

Datenbank (DB)

- An die Datenspeicherung in der Datenbank stellen sich folgende Anforderungen:
  - Datenintegrität (Daten innerhalb einer Datenbank bestimmte Regeln einhalten, damit die Korrektheit der Daten gesichert und die Geschäftslogik der Datenbank definiert ist)
  - Datensicherheit (Unbefugte keinen Zugriff auf gespeicherte Daten bekommen)
  - Abfrageoptimierung (Indizes dienen dazu, schnell einen bestimmten Datensatz zu finden)
  - Mehrbenutzerfähigkeit
  - Redundanzarmut

- Um den Anforderungen gerecht zu werden, gibt es definierte **Regeln** und **Strukturen**

- Diese werden von dem DBMS für die Datenbank festgelegt, man spricht von **Datenbankmodellen**

Datenbanksysteme

Datenbankmodelle

- Theoretische Grundlage für die Strukturierung der Daten und ihrer Beziehung zueinander ist das Datenbankmodell, man unterscheidet:
  - **Hierarchisches** Datenbankmodell
  - **Netzwerkartiges** Datenbankmodell
  - **Relationales** Datenbankmodell
  - **Objektorientiertes** Datenbankmodell
  - **Dokumentenorientiertes** Datenbankmodell
- Das bekannteste und meist verbreitete Datenbankmodell ist das **relationale Datenbankmodell**

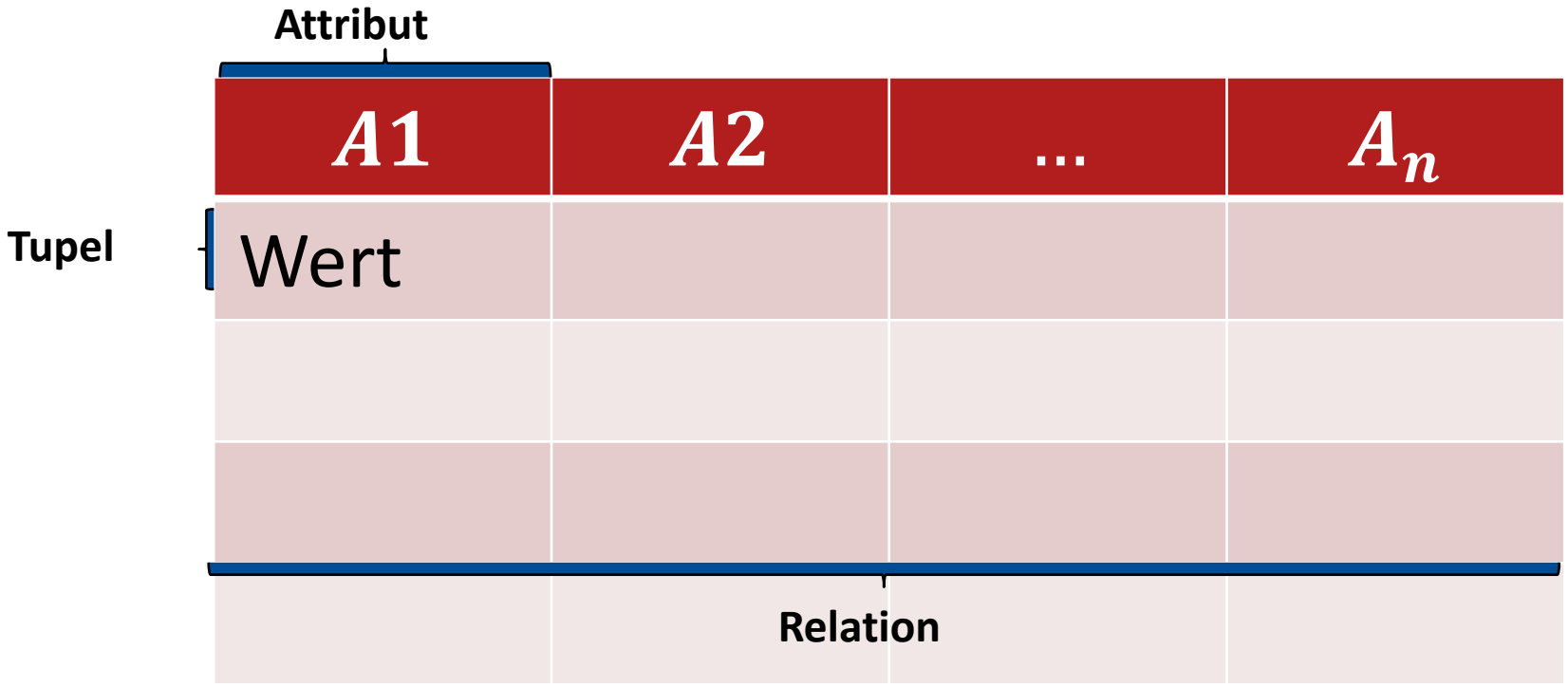
Agenda

1. Datenbanksysteme
2. **Relationale Datenbanken**
3. Einführung in SQL
4. Anwendungsbeispiel
5. SQL in Java
6. Objektrelationale Abbildung

Relationales Datenbankmodell

- Daten werden in **Tabellen** (*Relationen*) gespeichert

- Die Tabellen bestehen aus **Spalten** (*Attribute*) und **Zeilen** (*Tupel*)



Relationales Datenbankmodell

- In einer relationalen Datenbank befinden sich **beliebig viele Tabellen**, die untereinander verknüpft sind

- **Zur Erinnerung:**
  - Unser Ziel ist die effiziente und permanente Datenspeicherung sowie die Vermeidung von Redundanz

- Wie schafft man es, eine beliebige Anzahl an Tabellen zu verknüpfen und gleichzeitig die Abwesenheit von Redundanzen aufrecht zu erhalten?

- **Lösung:**
  - Identifizierung von Datensätzen in einer Datenbank anhand eines **einzigartigen Schlüssels**

Relationales Datenbankmodell

Kunden-Nummer	Name	Vorname	Ort
001	Meier	Max	Köln
002	Huber	Jens	Berlin
003	Müller	Stephan	Bochum
004	Schmitt	Sophie	Frankfurt

Der Rechnungsposition wird über den einzigartigen Primärschlüssel einem Kunden zugeordnet

Fremdschlüssel

Primärschlüssel

Rechnungspos.	Artikel	Anzahl	Kunden Nr.
V1	Tisch	1	001
V2	Stuhl	4	001
V3	Gardinen	2	003
V4	Teppich	1	004

Agenda

1. Datenbanksysteme
2. Relationale Datenbanken
3. **Einführung in SQL**
4. Anwendungsbeispiel
5. SQL in Java
6. Objektrelationale Abbildung

Einführung in SQL (Structured Query Language)

- SQL ist eine deskriptive Anfragesprache

- Untergliedert in:
  - **Data Definition Language** (Datendefinitionssprache)
    - Hauptfunktion: Datenbankschema definieren
  - **Data Manipulation Language** (Datenmanipulationssprache)
    - Hauptfunktion: Einfügen, Löschen und Verändern von Datensätzen
  - **Data Retrieval Language** (Datenbankanfragesprache)
    - Hauptfunktion: Anfragen an die DB

Data Definition Language (DDL)

- Die DDL definiert das **Schema** der Datenbank (Struktur)
- Dient dazu Datenstrukturen oder verwandte Elemente innerhalb der Datenbank zu **beschreiben**, **ändern** oder zu **entfernen**

Befehle:

CREATE	Erstellen eine Tabelle
ALTER	Hinzufügen einer neuen Spalte zu einer Tabelle
DROP	Löschen einer kompletten Tabelle

Data Manipulation Language (DML)

- Die DML ist der Teil der SQL-Sprache, der für die **Datenbearbeitung** verantwortlich ist
- Wird genutzt, um Daten innerhalb der Datenbank zu **schreiben**, **zu lesen, abzuändern oder zu löschen**

Befehle:

INSERT	Hinzufügen eines Eintrags in eine Tabelle
SELECT	Abfrage der Daten aus einer Tabelle
UPDATE	Ändern von Tabelleneinträgen
DELETE	Löschen von Tabelleneinträgen

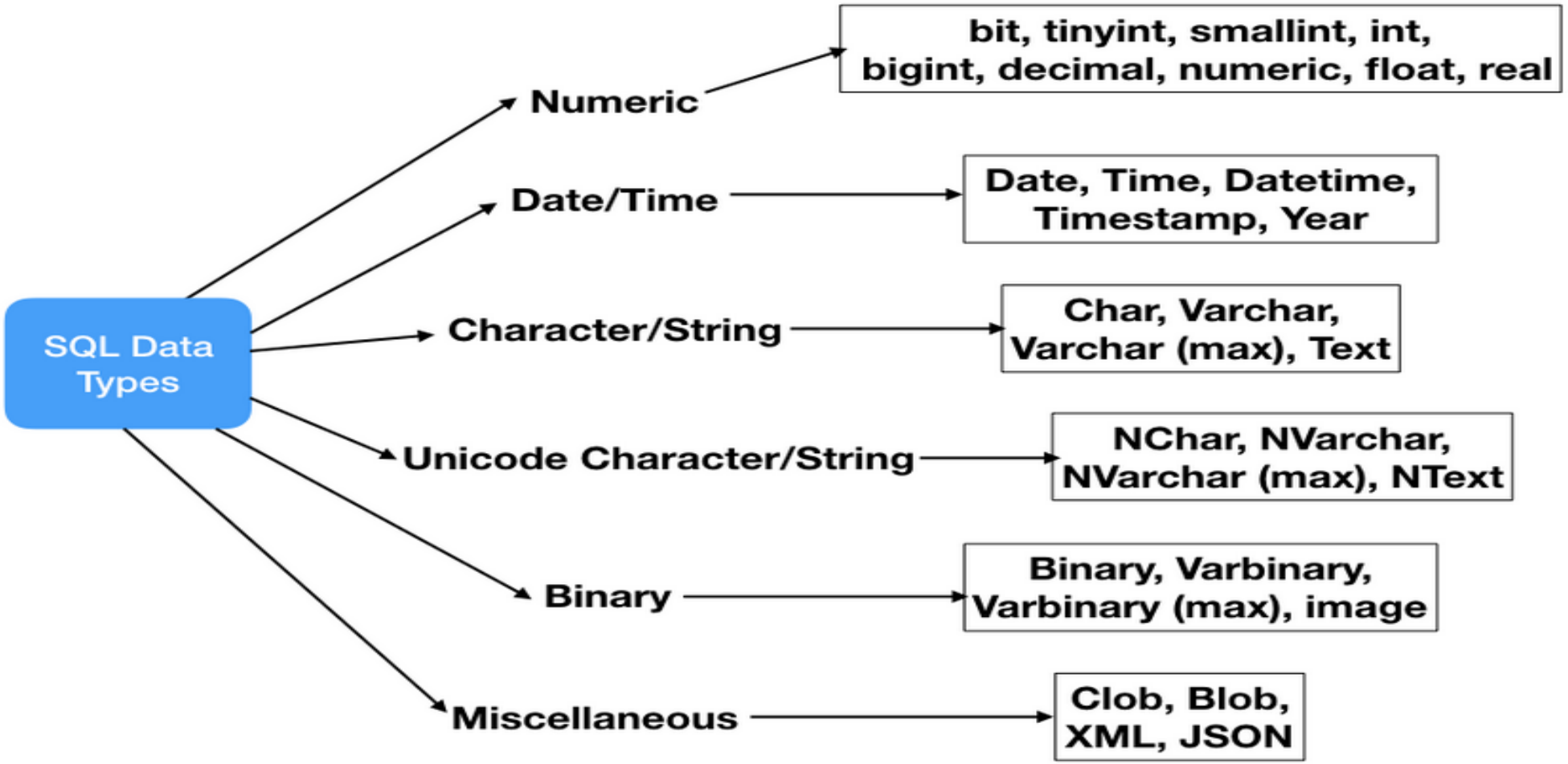
Data Retrieval Language (DRL)

- Die DRL wird genutzt um **spezifische Daten** von einer Datenbank abzufragen
- Um die Daten zu erhalten, wird der Befehl **SELECT** genutzt und der spezifische Ort in der Datenbank, an dem die Daten zu finden sind

Befehle:

FROM	Auswahl einer bestimmten Tabelle
WHERE	Auswahl einer bestimmten Tabellenspalte

Datentypen



Agenda

1. Datenbanksysteme
2. Relationale Datenbanken
3. Einführung in SQL
4. **Anwendungsbeispiel**
5. SQL in Java
6. Objektrelationale Abbildung

Anwendungsbeispiel

Students	
PK	matrikelnr
	name
	telefon
	stadt
	studiengang

Aufgabe: „Erstelle eine Liste mit Matrikelnummer (matrikelnr) und Name (name) aller Studenten (Students), die Wirtschaftsinformatik studieren (studiengang).“

matrikelnr	name	telefon	stadt	studiengang
9999999	Rahn	34010	Essen	Wirtschafsinformatik
9999991	Klopp	78091	Dortmund	Wirtschafsinformatik
9000000	Melches	98002	Essen	Angewandte Informatik

Anwendungsbeispiel - Lösung

„Erstelle eine Liste mit Matrikelnummer (matrikelnr) und Name (name) aller Studenten (Students), die Wirtschaftsinformatik studieren (studiengang).“

Lösung:

```
SELECT matrikelnr, name
FROM Students
WHERE studiengang = 'Wirtschaftsinformatik';
```

matrikelnr	name
9999999	Rahn
9999991	Klopp

Agenda

1. Datenbanksysteme
2. Relationale Datenbanken
3. Einführung in SQL
4. Anwendungsbeispiel
5. **SQL in Java**
6. Objektrelationale Abbildung

Anwendungsbeispiel
SQL Verbindungsaufbau


```
public static void SQLExample()
{
    String url = "jdbc:postgresql://localhost/SEP";
    String user = "SEP_USER";
    String password = "test";

    try(Connection con = DriverManager.getConnection(url,user,password))
```

Parameter welche zur Verbindung zur Datenbank Benötigt werden

Verbindung zur Datenbank
(Hier als Resource im Try-Catch-Block)

Anwendungsbeispiel
SQL Tabelle erstellen


```
try(Connection con = DriverManager.getConnection(url,user,password))
{
    String query = "Create Table Student(" +
            "vorname varchar(255)," +
            "nachname varchar(255)," +
            "Studiengang varchar(255)," +
            "Matrikelnummer int Primary Key," +
            "Stadt varchar(255)" +
            ")";

    Statement CreateTable =  con.createStatement();
    CreateTable.execute(query);
    CreateTable.close();
```

String welcher SQL Befehl
Zum erstellen einer Tabelle enthält

Matrikelnummer ist der Primärschlüssel
Der Tabelle Student

Statement wird von der Connection erstellt
und das Statement führt die SQL-Anfrage aus

Anwendungsbeispiel
SQL Daten einfügen


```
public static void insertStudent(Student student, Connection con) throws SQLException {
    String query = "INSERT INTO Student (vorname, nachname, Studiengang, Matrikelnummer, Stadt) VALUES " +
            ""+"('"+student.getVorName()+"','"+student.getNachName()+"','"+student.getStudiengang()+"','"+student.getMatrikelnummer()+"','"+student.getStadt()+"')";

    Statement Insert = con.createStatement();
    Insert.execute(query);
    Insert.close();
}
```

Methode nimmt Objekt vom Typ Student an

Erstellt aus Studentenobjekt passende SQL Anfrage

```
insertStudent(new Student( vorname: "Peter", nachname: "Parker", studiengang: "Fotografie", matrikelnummer: 1962, stadt: "New-York"),con);
insertStudent(new Student( vorname: "Clark", nachname: "Kent", studiengang: "Journalismus", matrikelnummer: 1938, stadt: "Smallville"),con);
insertStudent(new Student( vorname: "Barry", nachname: "Allen", studiengang: "Forensik", matrikelnummer: 1940, stadt: "Central City"),con);
insertStudent(new Student( vorname: "Matt", nachname: "Murdock", studiengang: "Rechtswissenschaft", matrikelnummer: 1964, stadt: "New-York"),con);
```

Anwendungsbeispiel

SQL Daten abfragen


```
query = "SELECT vorname,nachname,Stadt from Student WHERE Stadt= 'New-York'";
Statement getData = con.createStatement();
ResultSet RS = getData.executeQuery(query);
while(RS.next())
{
    System.out.println(RS.getString( columnLabel: "vorname")+" "+RS.getString( columnLabel: "nachname")+" kommt aus "+RS.getString( columnLabel: "Stadt"));
}
```

SQL Select-Abfrage

Ergebnis der Anfrage wird in einem Resultset gespeichert

	vorname character varying (255)	**nachname** character varying (255)	**studiengang** character varying (255)	**matrikelnummer** [PK] integer	**stadt** character varying (255)
1	Clark	Kent	Journalismus	1938	Smallville
2	Barry	Allen	Forensik	1940	Central City
3	Peter	Parker	Fotografie	1962	New-York
4	Matt	Murdock	Rechtswissenschaft	1964	New-York

```
Peter Parker kommt aus New-York
Matt Murdock kommt aus New-York
```

Ausgabe auf der Konsole

Agenda

1. Datenbanksysteme
2. Relationale Datenbanken
3. Einführung in SQL
4. Anwendungsbeispiel
5. SQL in Java
6. **Objektrelationale Abbildung**

Objektrelationale Abbildung (Object-relational mapping)

- Eine objektrelationale Abbildung ist eine Technik in der Softwareentwicklung, bei der Objekte einer objektorientierten Programmiersprache (z.B. **Java**) auf eine relationale Datenbank abgebildet werden

- Beispielsoftware zur Nutzung von ORM :
  - Hibernate
  - ORMLite
  - EclipseLink
  - Etc.

- Ermöglicht das Verwalten von Daten einer relationalen Datenbank direkt aus Java(Quellcode) heraus

Anwendungsbeispiel
ORMLite Verbingungsgaufbau


```
public static void ORMExample()
{
    String url = "jdbc:postgresql://localhost/SEP";
    String user = "SEP_USER";
    String password = "test";

    try(JdbcPooledConnectionSource connectionSource = new JdbcPooledConnectionSource(url,user,password))
    {
```

Parameter, welche zur Verbindung zur Datenbank Benötigt werden

Verbindung zur Datenbank
(Hier als Resource im Try-Catch-Block)

Anwendungsbeispiel

ORMLite Daten einfügen


```
TableUtils.createTable(connectionSource, Student.class);

Dao<Student, Integer> StudentDao = DaoManager.createDao(connectionSource, Student.class);
StudentDao.create(new Student( vorname: "Peter", nachname: "Parker", studiengang: "Fotografie", matrikelnummer: 1962, stadt: "New-York"));
StudentDao.create(new Student( vorname: "Clark", nachname: "Kent", studiengang: "Journalismus", matrikelnummer: 1938, stadt: "Smallville"));
StudentDao.create(new Student( vorname: "Barry", nachname: "Allen", studiengang: "Forensik", matrikelnummer: 1940, stadt: "Central City"));
StudentDao.create(new Student( vorname: "Matt", nachname: "Murdock", studiengang: "Rechtswissenschaft", matrikelnummer: 1964, stadt: "New-York"));
```

Tabelle wird in Datenbank
mit der Klasse Student erstellt

Mittels Data Access Objects(DAO)
werden nun Studenten in die Datenbank inseriert

Anwendungsbeispiel

ORMLite Klasse

```
@DatabaseTable(tableName = "Student")
public class Student
{
    @DatabaseField(canBeNull = false)
    private String vorName;

    @DatabaseField(canBeNull = false)
    private String nachName;

    @DatabaseField(canBeNull = false)
    private String Studiengang;

    @DatabaseField(id = true)
    private int Matrikelnummer;

    @DatabaseField(canBeNull = false)
    private String Stadt;

    public Student()
    {}
```

Definiert wie die Tabelle
In der Datenbank heißen wird

„@DatabaseField“ gibt an, dass in der Tabelle entsprechende Spalten erstellt werden müssen

Hier wird definiert, dass die Matrikelnummer der Primärschlüssel der Tabelle ist

Um aus Klassen Tabellen zu erstellen benötigt ORMLite einen leeren Konstruktor der Klasse

```
    public Student(String vorname, String nachname, String studiengang, int matrikelnummer, String stadt) {
        this.vorName = vorname;
        this.nachName = nachname;
        this.Studiengang = studiengang;
        this.Matrikelnummer = matrikelnummer;
        this.Stadt = stadt;
    }
```

Anwendungsbeispiel

ORMLite Datenabfrage

```
List<Student> StudentenListe = StudentDao.queryForAll();
for (Student student : StudentenListe )
{
    if(student.getStadt().equals("New-York"))
    {
        System.out.println(student.getVorName()+" "+student.getNachName()+ " Stammt aus New-York");
    }
}
```

Mit dem DAO können auch Elemente aus der Datenbank gelesen werden

Einträge der Tabelle in der Datenbank:

	vorName character varying (255)	nachName character varying (255)	Studiengang character varying (255)	Matrikelnummer [PK] integer	Stadt character varying (255)
1	Peter	Parker	Fotografie	1962	New-York
2	Clark	Kent	Journalismus	1938	Smallville
3	Barry	Allen	Forensik	1940	Central City
4	Matt	Murdock	Rechtswissenschaft	1964	New-York

Ausgabe auf der Konsole:

```
Peter Parker Stammt aus New-York
Matt Murdock Stammt aus New-York
```

ORMLite – Und jetzt?

- https://ormlite.com/javadoc/ormlite-core/doc-files/ormlite.html#Top
  - Vollständige Dokumentation
- https://github.com/j256/ormlite-jdbc
  - Beispielprojekt mit den wichtigsten Funktionalitäten von ORMLite

Quellen

- https://www.tinohempel.de/info/info/datenbank/sql_tabelle.htm
- https://www.datenbanken-verstehen.de/datenbank-grundlagen/dbms/dml-ddl-dcl-kommandos/
- https://www.bigdata-insider.de/was-ist-eine-relationale-datenbank-a-643028/
- https://www.ionos.de/digitalguide/hosting/hosting-technik/relationale-datenbanken/
- https://t3n.de/news/eigentlich-relationale-datenbanken-683688/2/
- https://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Datenmanagement/Datenbanksystem/Datenbankmanagementsystem/index.html
- http://www.gitta.info/RelQueryLang/de/text/RelQueryLang.pdf

Verwendete Grafiken

- https://www.journaldev.com/16774/sql-data-types
- https://thenounproject.com/
  - Light Bulb by Alexander Skowalsky

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Vielen Dank für Ihre Aufmerksamkeit

PALUNO
The Ruhr Institute for Software Technology