FUELS-Seminar am 5. Januar 2022
Charles E. Müller (FU/FUELS)
charles.mueller@fu-berlin.de

Freie Universität Berlin

DatabaseConstructor – Softwarelösung zum Aufbau wissenschaftlicher Datenbanken

Mit einer Einführung in die relationale Datenmodellierung

FUELS
EMPIRICAL LEGAL STUDIES CENTER

A. Einführung in die relationale Datenmodellierung

I. Einleitung

- Relationale Datenbanktechnologien haben seit den 1980ern weite Verbreitung gefunden
- Zentraler Gedanke: Vermeidung von **Redundanzen**
- Ein relationales Datenmodell beschreibt Objekte (Entitäten) und die Beziehungen zwischen ihnen

II. Datenmodell – 1. Grundbegriffe

- **Datenbank** setzt sich aus **Tabellen** zusammen
- Eine Tabelle entspricht einer Entität im Datenmodell

Feld / Attribut

Datensatz

Zu unterscheiden vom statistischen Datensatzbegriff

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

II. Datenmodell – 2. Beziehungen – a) 1:n

Jedes Buch hat einen Autor, aber ein Autor kann mehrere Bücher schreiben

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

ID	Autor	Titel	Jahr
1	1	System d. h. röm. R., Band 1	1840
2	3	Römische Geschichte, Band 1	1854
3	3	Römische Geschichte, Band 2	1855
4	3	Römische Geschichte, Band 3	1856
5	2	Pandekten, Band 1	1862
6	2	Pandekten, Band 2,1	1865
7	2	Pandekten, Band 2,2	1866
8	2	Pandekten, Band 3	1870
9	4	Scherz und Ernst i. d. Jurispr.	1884

II. Datenmodell – 2. Beziehungen – a) 1:n

Primärschlüssel (Primary Key)

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

Fremdschlüssel (Foreign Key)

ID	Autor	Titel	Jahr
1	1	System d. h. röm. R., Band 1	1840
2	3	Römische Geschichte, Band 1	1854
3	3	Römische Geschichte, Band 2	1855
4	3	Römische Geschichte, Band 3	1856
5	2	Pandekten, Band 1	1862
6	2	Pandekten, Band 2,1	1865
7	2	Pandekten, Band 2,2	1866
8	2	Pandekten, Band 3	1870
9	4	Scherz und Ernst i. d. Jurispr.	1884

II. Datenmodell – 2. Beziehungen – b) n:m

Jedes Buch kann mehrere Autoren haben und jeder Autor mehrere Bücher

ID	Vorname	Nachname
1	Friedrich Carl	von Savigny
2	Bernhard	Windscheid
3	Theodor	Mommsen
4	Rudolf	von Jhering

Einfügen einer Hilfstabelle zur Abbildung einer n:m-Beziehung

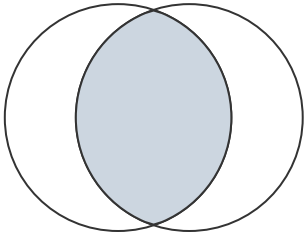
ID	Autor	Buch
1	1	1
2	2	5
3	2	6
4	2	7
5	2	8
6	3	2
7	3	3
8	3	4
9	4	9

ID	Titel	Jahr
1	System d. h. röm. R., Band 1	1840
2	Römische Geschichte, Band 1	1854
3	Römische Geschichte, Band 2	1855
4	Römische Geschichte, Band 3	1856
5	Pandekten, Band 1	1862
6	Pandekten, Band 2,1	1865
7	Pandekten, Band 2,2	1866
8	Pandekten, Band 3	1870
9	Scherz und Ernst i. d. Jurispr.	1884

II. Datenmodell – 3. Joins

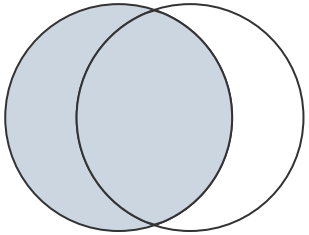
Inner Join

Die Zeilen aus beiden Tabellen, die mit der anderen verbunden werden können



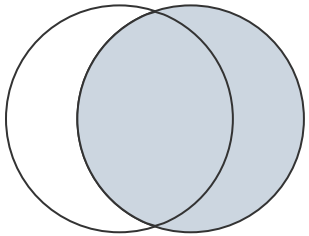
Left Join

Alle Zeilen der linken Tabelle, wo möglich mit der rechten Tabelle verbunden



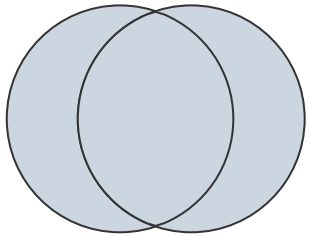
Right Join

Alle Zeilen der rechten Tabelle, wo möglich mit der linken Tabelle verbunden



Full Join

Alle Zeilen aus beiden Tabellen, verbunden wo möglich



II. Datenmodell – 3. Joins – a) Inner Join

ID	Autor	Titel	Jahr
1	1	System d. h. röm. R., Band 1	1840
2	3	Römische Geschichte, Band 1	1854
3	3	Römische Geschichte, Band 2	1855
4	6	Allgemeine Staatslehre	1900

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

ID	Autor	Titel	Jahr	ID	Vorname	Nachname	Geburtstag
1	1	System d. h. röm. R., Band 1	1840	1	Friedrich Carl	von Savigny	21.02.1779
2	3	Römische Geschichte, Band 1	1854	3	Theodor	Mommsen	30.11.1817
3	3	Römische Geschichte, Band 2	1855	3	Theodor	Mommsen	30.11.1817

II. Datenmodell – 3. Joins – b) Left Join

ID	Autor	Titel	Jahr
1	1	System d. h. röm. R., Band 1	1840
2	3	Römische Geschichte, Band 1	1854
3	3	Römische Geschichte, Band 2	1855
4	6	Allgemeine Staatslehre	1900

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

ID	Autor	Titel	Jahr	ID	Vorname	Nachname	Geburtstag
1	1	System d. h. röm. R., Band 1	1840	1	Friedrich Carl	von Savigny	21.02.1779
2	3	Römische Geschichte, Band 1	1854	3	Theodor	Mommsen	30.11.1817
3	3	Römische Geschichte, Band 2	1855	3	Theodor	Mommsen	30.11.1817
4	6	Allgemeine Staatslehre	1900	*NULL*	*NULL*	*NULL*	*NULL*

II. Datenmodell – 3. Joins – c) Full Join

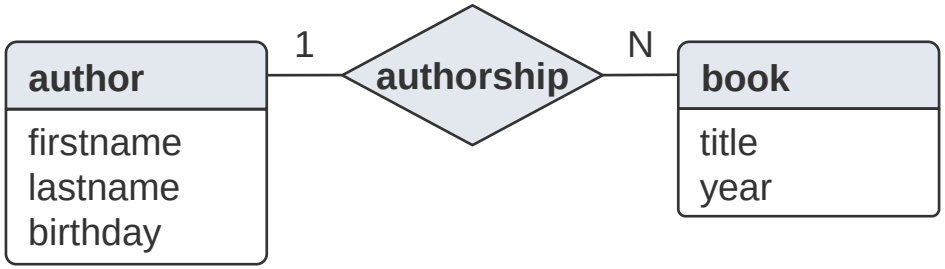
ID	Autor	Titel	Jahr
1	1	System d. h. röm. R., Band 1	1840
2	3	Römische Geschichte, Band 1	1854
3	3	Römische Geschichte, Band 2	1855
4	6	Allgemeine Staatslehre	1900

ID	Vorname	Nachname	Geburtstag
1	Friedrich Carl	von Savigny	21.02.1779
2	Bernhard	Windscheid	26.06.1817
3	Theodor	Mommsen	30.11.1817
4	Rudolf	von Jhering	22.08.1818

ID	Autor	Titel	Jahr	ID	Vorname	Nachname	Geburtstag
1	1	System d. h. röm. R., Band 1	1840	1	Friedrich Carl	von Savigny	21.02.1779
N.	*NULL*	*NULL*	*NULL*	2	Bernhard	Windscheid	26.06.1817
2	3	Römische Geschichte, Band 1	1854	3	Theodor	Mommsen	30.11.1817
3	3	Römische Geschichte, Band 2	1855	3	Theodor	Mommsen	30.11.1817
4	6	Allgemeine Staatslehre	1900	*NULL*	*NULL*	*NULL*	*NULL*
N.	*NULL*	*NULL*	*NULL*	4	Rudolf	von Jhering	22.08.1818

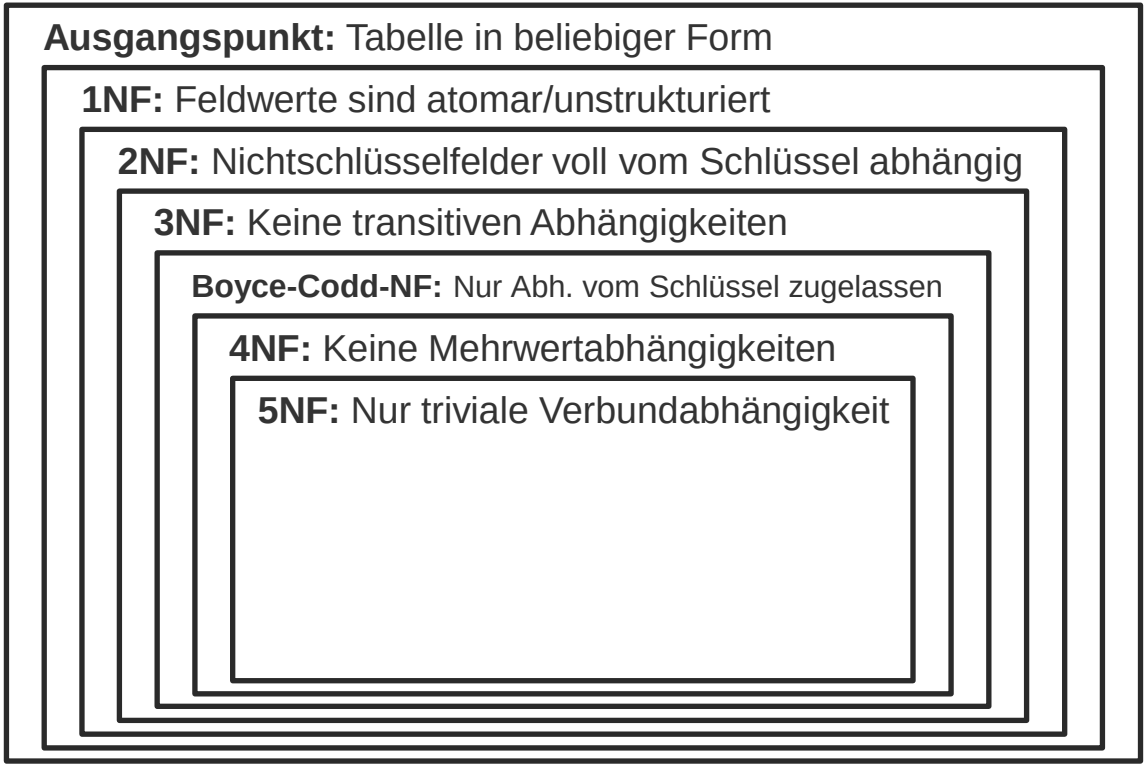
II. Datenmodell – 4. Visualisierung

- **Entity-Relationship Model** von Peter Chen (1976)
- Verschiedene Varianten
- Hier: **Chen-Notation** + Attribute



II. Datenmodell – 5. Normalisierung

Normalformen (NF): Abfolge von Regeln zur Prüfung eines relationalen Modells, u.a. zur Vermeidung von Redundanzen



Darstellung nach
Meier/Kaufmann[8], S. 38 = Meier[7], S. 40

III. Anwendung – 1. SQL

- Structured Query Language
- Aussprache: „S-Q-L“ oder auch „SEQUEL“
- Verschiedene Implementationen, u.a.:
  - **MySQL/MariaDB:** Datenbankserver
    - Inspektion: phpMyAdmin
  - **SQLite:** Datenbank wird in einer Datei gespeichert
    - Für kleinere Datenbanken; leichtere Handhabung
    - Inspektion: SQLBrowser

III. Anwendung – 1. SQL – a) Befehle

- Alle Interaktionen mit der Datenbank erfolgen durch SQL-Befehle:
  - **Struktur-aufbau**
  - Eingaben
  - Abfragen
  - Änderungen

Im Beispiel: MySQL

```
CREATE TABLE author (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(60) NOT NULL,
  lastname VARCHAR(60) NOT NULL,
  birthday DATE NOT NULL
) DEFAULT CHARSET=utf8mb4
```

`INT UNSIGNED`
: Zusatz `UNSIGNED` bedeutet keine Vorzeichen, Wertebereich verdoppelt sich (bei `INT` auf 0–4.294.967.295)

`AUTO_INCREMENT`
: Besorgt automatische laufende Nummerierung

`PRIMARY KEY`
: Kennzeichnung als Primärschlüssel

`DEFAULT CHARSET=utf8mb4`
: UTF-8 all the way through!

III. Anwendung – 1. SQL – a) Befehle

- Alle Interaktionen mit der Datenbank erfolgen durch SQL-Befehle:
  - Struktur-aufbau
  - **Eingaben**
  - Abfragen
  - Änderungen

```
CREATE TABLE author (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(60) NOT NULL,
  lastname VARCHAR(60) NOT NULL,
  birthday DATE NOT NULL
) DEFAULT CHARSET=utf8mb4
```

```
INSERT INTO author (firstname, lastname, year)
  VALUES ('Heinrichhh', 'Heine', '1797-12-13')
```

- `id` wird dank `AUTO_INCREMENT` automatisch befüllt
- Niemals Benutzereingaben direkt in den Query einfügen: Gefahr von SQL Injections

Im Beispiel: MySQL

III. Anwendung – 1. SQL – a) Befehle

- Alle Interaktionen mit der Datenbank erfolgen durch SQL-Befehle:
  - Struktur-aufbau
  - Eingaben
  - **Abfragen**
  - **Änderungen**

```
CREATE TABLE author (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(60) NOT NULL,
  lastname VARCHAR(60) NOT NULL,
  birthday DATE NOT NULL
) DEFAULT CHARSET=utf8mb4
```

```
INSERT INTO author (firstname, lastname, year)
  VALUES ('Heinrichhh', 'Heine', '1797-12-13')
```

```
SELECT * FROM author
```

```
SELECT firstname
  FROM author
  WHERE lastname='Heine'
```

```
UPDATE author SET firstname='Heinrich' WHERE id='1'
```

Im Beispiel: MySQL

III. Anwendung – 1. SQL – a) Befehle

SELECT-Befehle können Joins enthalten:

```
SELECT author.lastname AS lastname,
       author.firstname AS firstname,
       book.title AS title,
       book.year AS year
  FROM book
  LEFT JOIN author
    ON book.author_id = author.id
  ORDER BY book.year,
           author.lastname,
           author.firstname
```

firstname	lastname	title	year
Bernhard	Windscheid	Pandekten, Band 1	1862
Bernhard	Windscheid	Pandekten, Band 2,1	1865
Bernhard	Windscheid	Pandekten, Band 2,2	1866
Bernhard	Windscheid	Pandekten, Band 3	1870
Rudolf	von Jhering	Scherz und Ernst i. d. Jurispr.	1884

III. Anwendung – 1. SQL – b) Zugriff mit R

- RMariaDB – R package for MariaDB and MySQL

  https://cran.r-project.org/web/packages/RMariaDB/index.html

- RSQLite – R package for SQLite

  https://cran.r-project.org/web/packages/RSQLite/index.html

  Übersicht zu Datenbankzugriff-Packages für R:
  https://cran.r-project.org/web/views/Databases.html

III. Anwendung – 1. SQL – b) Zugriff mit R

```
library(RMariaDB)

connection <- dbConnect(drv = RMariaDB::MariaDB(),
                        host = "127.0.0.1",
                        username = "root",
                        password = "",
                        dbname = "library")

result <- dbSendQuery(connection, "SELECT * FROM author")
table <- dbFetch(result)
dbClearResult(result)

dbDisconnect(connection)

# Shows queried table
table
```

III. Anwendung – 1. SQL – c) Zugriff mit Python

– Connector/Python – Python library for MySQL access

  https://dev.mysql.com/doc/connector-python/en/connector-python-introduction.html

– sqlite3 – Python library for SQLite access

  https://docs.python.org/3/library/sqlite3.html

III. Anwendung – 2. CSV

- Tabellen einer relationalen Datenbank können auch als CSV-Tabellen gespeichert werden
- Einzelne SQL-Funktionen sind auch unmittelbar in R oder Python implementiert, z.B. Joins:
  - R: merge(); join-Funktionen im dplyr-package

    https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/merge

    https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/join
  - Python: join() und merge() für Pandas DataFrames

    pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.join.html

III. Anwendung – 2. CSV

```
library(dplyr)

# Read from CSV
authors <- read.csv("authors.csv")
books <- read.csv("books.csv")

# Left join with dplyr::left_join()
left_join(books, authors, by = c("author_id" = "id"))

# Left join using infix operator (%>%)
# infix operator uses left-hand side as first argument for right-hand side
books %>% left_join(authors, by = c("author_id" = "id"))

# Left join with base::merge()
# Configuring merge() as left join: all.x = TRUE, all.y = FALSE
merge(books, authors, by.x = "author_id", by.y = "id", all.x = TRUE, all.y = FALSE)
```

IV. Anwendungsbeispiele

- Christoph Kling (Mannheim): Konkurshistorische Datenbank Deutsches Kaiserreich (1879–1914)

  https://digi.bib.uni-mannheim.de/konkursdatenbank/

  Verwendet MariaDB; zugehörge Dissertation im Erscheinen

  Zugleich beispielhaft für Formen der Veröffentlichung von SQL-Datenbanken

- Andreas Fleckner (Berlin): Anlegermitverschulden vor dem Bankensenat, FS-Hopt 2020, S. 253–277

  Online-Anhang (noch) nicht abrufbar (vgl. a.a.O, Fn. 8)

  Verwendet wohl SQLite; vgl. Coupette/Fleckner, JZ 2018, 379 (385)

V. Ausblick: Grenzen relationaler Datenmodelle

- An verschiedenen Stellen stoßen relationale Datenmodelle inzwischen an Grenzen
- NoSQL-Bewegung (auch als „not only SQL“ verstanden)
- „Postrelationale Datenbanken“ (Andreas Meier)
- Unterschiedliche Lösungsansätze
  - Beispiel: MongoDB als dokumentenorientiertes, flexibles NoSQL-DBMS

MongoDB: https://www.mongodb.com/

List of NoSQL-DBMS: https://hostingdata.co.uk/nosql-database/

V. Ausblick: Grenzen relationaler Datenmodelle

```
{
    "author": 2,
    "title": "Pandekten",
    "volumes": [
        {
            "desc": "Band 1",
            "year": 1862
        },
        {
            "volumes": [
                {
                    "desc": "Band 2,1",
                    "year": 1865
                },
                {
                    "desc": "Band 2,2",
                    "year": 1866
                }
            ]
        },
        {
            "desc": "Band 3",
            "year": 1870
        }
    ]
}
```

V. Ausblick: Grenzen relationaler Datenmodelle

- Manchmal ist es notwendig, Informationen zu jedem Datensatz *ohne einheitliche Struktur* zu speichern
- Behelf (ohne SQL zu verlassen): Ein Feld, in dem etwa ein flexibles JSON-Objekt gespeichert wird
- MySQL enthält inzwischen einen JSON-Datentyp und Funktionen zur Interaktion mit JSON-Objekten

  MySQL ≥ 5.7.8 https://dev.mysql.com/doc/refman/8.0/en/json.html

  MariaDB ≥ 10.2.7 https://mariadb.com/kb/en/json-data-type/

VI. Literaturauswahl

- *Harrington:* Relational Database Design and Implementation, 4th ed. 2016 [Primo FU]
- *Meier/Kaufmann:* SQL- & NoSQL-Datenbanken, 8. Aufl. 2016 [Primo TU]; Vorauflage *Meier:* Relationale und postrelationale Datenbanken, 7. Aufl. 2010 [Primo FU]

Zur theoretischen Grundlegung:

- *Edgar F. Codd:* A Relational Model of Data for Large Shared Data Banks, Comm. ACM 13 (1970), 377–387; zuvor internes IBM-Papier
- *Edgar F. Codd:* The Relational Model for Database Management: Version 2, 1990

B. Vorstellung der Software

I. Entstehung

- FUELS-Projekt *Germany, Inc.:*
  Aufbau einer Datenbank zu deutschen börsen-gehandelten Aktiengesellschaften 1960–2020
- Fragestellung: Wie kann die händische Datenerfassung umgesetzt werden?
- Entwicklung einer Lösung für alle Vorhaben dieser Art an Stelle eines nur für dieses Vorhaben verwendbaren Programmes

II. Anwendungsfeld

- **Datenerfassung von Hand**
- Relationale Struktur möglich, aber nicht zwingend
  - Exportformate: CSV, SQL
- Gleichzeitiges Arbeiten mit mehreren Personen
  - Zugang über Webbrowser:
    Kein Installationsaufwand für einzelne Geräte

III. Funktionen – 1. Grundlegendes

- Alle Beteiligten haben eigene Zugänge
- „Projekt“: Einzelne Datenbank,
  mehrere Projekte auf derselben Installation
  mit unterschiedlichen Beteiligten möglich
- Rollenzuweisung innerhalb eines Projektes
- „Wiki“: Bereitstellen detaillierter Leitfäden

III. Funktionen – 2. Strukturentwicklung

- Relationsfelder: Auswahl eines Datensatzes aus einer bestimmten Tabelle
- Wertfelder: Eingabe eines Wertes – Datentypen: Text, Auswahl, Datum, Zahl, boolsches Feld (u.a.)
- Validationskriterien: Automatische Eingabeprüfung und Bestimmung des passenden SQL-Datentyps
- Erläuterung: Leitfäden für die Datenerfassung

III. Funktionen – 2. Strukturentwicklung

- Reguläre Ausdrücke als Validationskriterien für Wertfelder ermöglichen flexible Eingabeprüfung
- Beispiel: Aktenzeichen von Verfassungsbeschwerden

```
^[1|2] BvR \d+\/\d{2}$
```

Erläuterung:

`^`	Anfang der Eingabe
`[1|2]`	„1“ oder „2“
`\d`	Arabische Ziffer
`+`	Vorheriges Zeichen wenigstens ein Mal
`{2}`	Vorheriges Zeichen genau zwei Mal
`$`	Ende der Eingabe

- Können getestet werden etwa auf regex101.com

Einführung: https://www.oreilly.com/content/an-introduction-to-regular-expressions/

III. Funktionen – 3. Eingabe

- Einfaches Eingabeformular für jede Tabelle vorhanden
- Zusätzlich Konfiguration von Eingabeformularen für komplexere Vorgänge möglich („Eingaberoutinen“)
- Bei Gelegenheit der Eingabe:
  - Kommentare: Festhalten von Unklarheiten
  - Kennzeichnung zur Nachverfolgung
  - Zuweisung zu einem anderen Projektbeteiligten

III. Funktionen – 4. Erfasste Daten

- Tabellarische Vorschau
- Korrekturmöglichkeit
- Kennzeichnung, Zuweisung
- Änderungshistorie
- Kommunikation mit Kommentaren

III. Funktionen – 5. Export

- Formate: CSV, SQL, weitere Formate geplant
- Export nur von Datensätzen, die den Validitätskriterien entsprechen
- Exportdateien können heruntergeladen werden

IV. Installation

- Technische Anforderungen:
  - Apache HTTPD Webserver mit PHP ≥ 7.0
  - MySQL/MariaDB-Datenbankserver
- Installation auch auf „normalem“ Computer möglich

  Für Windows: XAMPP – https://www.apachefriends.org/download.html
- Installationshinweise auf GitHub

IV. Installation

- Installation für *Germany, Inc.* auf ZEDAT-Userpage
- ZEDAT bietet mit „Userpage“ jedem Benutzer einen Webspace – URL: https://userpage.fu-berlin.de/<Benutzername>
  - Apache und PHP (PHP-Version 7.0.33)
  - Zugang zum Webspace mit SFTP (oder WebDav)
  - Außerdem MariaDB-Server (10.1.48), phpMyAdmin

https://www.zedat.fu-berlin.de/Userpage

https://www.zedat.fu-berlin.de/Datenbanken

V. GitHub und Demo-Installation

- GitHub-Repository mit Quellcode und Download: https://github.com/SyntaxCacao/DBConstructor/
- Demo-Installation für alle offen: https://userpage.fu-berlin.de/germanyinc/dbc-demo/
  Benutzername: admin
  Passwort: admin