

Datenbanken & Webtechnologien

Marcel Remmy M. ENG.

PRAKTIKUM MEILENSTEIN 2

PAKET 2

SUPPORT: KW45

ABNAHME: KW47

PAKET 3

SUPPORT: KW46

ABNAHME: KW47

Meilenstein 2 | Paket 2

In diesem Paket lernen Sie, wie Sie ein Entity-Relationship-Diagramm in das Schema eines relationalen DBMS überführen. Sie lernen den SQL Dialekt für MySQL/MariaDB [1](#).

Öffnen Sie **HeidiSQL** oder einen alternativen SQL Client und verbinden Sie sich mit der MariaDB Instanz, die Sie zugeteilt bekommen (Support-Termin oder E-Mail).

Im Ordner **Praktikum/Meilenstein 2** in ILIAS befindet sich das ERD, das Sie für das Praktikum verwenden werden. Sie finden in der Übung 2 eine SQL-Datei mit ersten **CREATE TABLE** -Statements für dieses Praktikum.

Darüber hinaus sollten Sie – im eigenen Interesse – alle ihre SQL Anweisungen in eigene Textdateien (.txt oder .sql) speichern oder ihrem Praktikums-Dossier hinzufügen. Auch dieses Paket hat Fragen für ins Dossier.

Wichtig: Bearbeiten Sie die Aufgabe auf dem Ihnen zugeteilten MariaDB-Server und speichern Sie regelmäßig ein SQL-Export.

Tipps

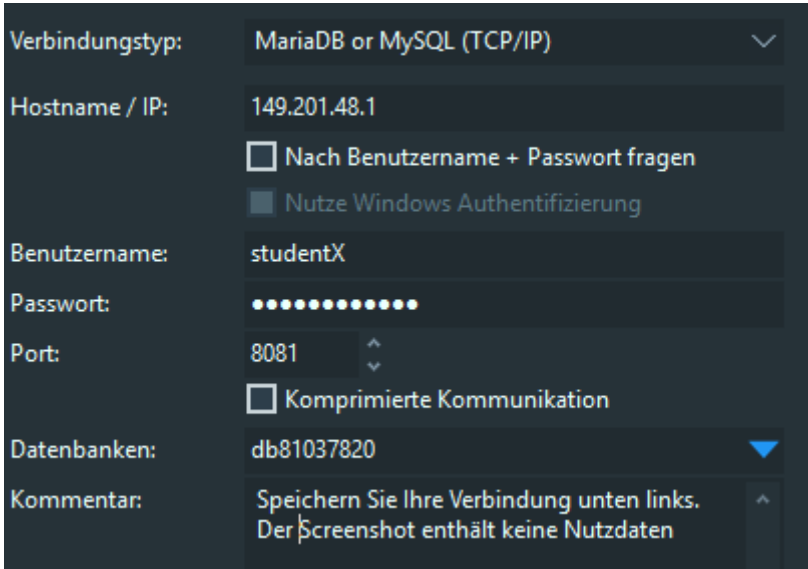
- Verwenden Sie SQL Kommentare, um den Überblick zu behalten (Diese werden eingeleitet mit **--**)
- Die MariaDB Knowledge Base [1](#) ist für dieses Paket die beste Quelle.

Aufgabe 2.0 : Verbindung mit der Datenbank

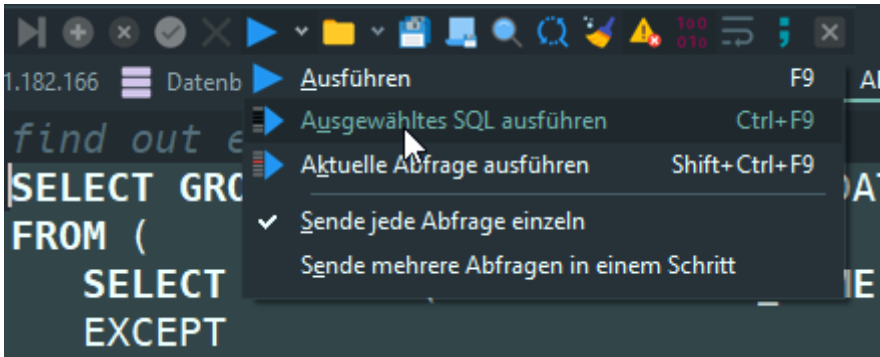
Sie erhalten im Praktikum Benutzernamen und -passwort für Ihre dedizierte Datenbank auf einem Server. Auf diesem lösen Sie die Aufgaben für dieses Paket.

Tipps zu HeidiSQL

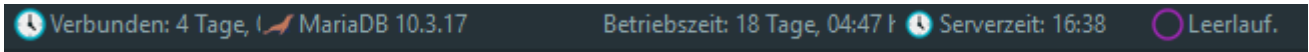
- Verbinden Sie sich im **Verbindungsmanager** (Datei ➤ Verbindungsmanager) mit der Datenbank, die Ihnen zugeteilt wurde und speichern Sie die Verbindung, falls gewünscht. Verwenden Sie die Daten, die Sie per E-Mail (Alumni-Adresse) oder im Support-Termin erhalten haben.



- Mit **STRG+T** können Sie Abfragefenster/Tabs öffnen, in denen Sie SQL schreiben können
- Sie können die Ausführung der ausgewählten SQL-Befehle mit **STRG+F9** anfordern. Shortcut **F9** führt alle SQL-Befehle im Abfragefenster aus.



- Prüfen Sie bei der Arbeit mit HeidiSQL, ob Sie mit dem Server verbunden sind. In der Fußleiste des Programms erhalten Sie wichtige Infos.



Aufgabe 2.1 : Entitäten in Tabellen überführen

Die in der Übung begonnene SQL-Datei ist unvollständig und muss von Ihnen ergänzt bzw. geändert werden. Die Datei ist so angelegt, dass beim Ausführen die definierten Tabellen gelöscht werden, sofern Sie vorhanden sind (**IF EXIST** Anweisungen). Dies ermöglicht Ihnen ein einfacheres Testen ihrer Tabellendefinitionen.

Beachten Sie in allen Fällen die [Business Rules](#).

- Erzeugen Sie diese Anweisungen für neu angelegte Tabellen selbst und beachten Sie die Reihenfolge.
- In der Abgabe muss Ihr komplettes SQL-Script mehrfach hintereinander ohne Fehler ausführbar sein!
- **DROP DATABASE** ist nicht gültig, nutzen Sie lediglich **DROP TABLE** in Ihrem SQL-Script zum Löschen und lassen Sie die DB bestehen.
- Lassen Sie die Prüfung auf referentielle Integrität zu jeder Zeit intakt (das ist Standard).
- Modifizieren Sie zunächst die **CREATE TABLE** Anweisungen der Entitäten wie folgt ab und achten Sie darauf, die Attribute exakt so zu benennen, wie es das ER-Diagramm vorgibt.
- Wählen Sie immer passende Datentypen (z.B. für "Bild.Binärdaten" bietet sich ein Binary Large Object an).
- Beachten Sie bei der Umsetzung in SQL auch optionale Attribute und Schlüssel. Nicht-optionale Attribute dürfen keine **NULL** -Werte erlauben. Verwenden Sie wenn möglich sinnvolle **DEFAULT** Werte.
- Definieren Sie abhängig von der Domäne des Attributs die reservierte Speichergröße und setzen Sie sich mit den Datentypen auseinander. Wählen Sie den bestmöglichen Datentyp für jedes Attribut! (siehe [2](#) und [3](#)). Wählen Sie vor allem für den Primärschlüssel optimale Datentypen und lassen Sie ggfs. die Surrogate automatisch generieren (mit **AUTO_INCREMENT**).
- 'Flags' sind binär und werden mit dem Datentyp bool abgebildet.
- Temporale Daten müssen den korrekten Datentyp enthalten, definieren Sie, ob Sie nur das Datum (**DATE**)oder auch den Zeitpunkt (**DATETIME**) speichern müssen.

Business Rules

1. Die Entität **Nutzer** weist 'Auth'-Attribute auf, deren Zweck erst in einer späteren Vorlesung behandelt werden, hier jedoch Angaben der zu speichernden Daten:
 - 'Hash' ist immer eine 60 Zeichen lange Zeichenfolge
2. Die Attribute 'E-Mail' und 'Nutzername' von **Nutzern** müssen jeweils eindeutige Werte aufweisen. 'LetzterLogin' speichert einen Zeitpunkt und ist standardmäßig **NULL** . In 'Geburtsdatum' wird optional das Datum gespeichert und daraus kann das 'Alter' errechnet werden.
3. Die **Matrikelnummer** bei **Studenten** muss eindeutig und eine 8- oder 9-stellige Zahl sein. Der 'Studiengang' wird nur die folgenden Werte erlauben: ET, INF, ISE, MCD, WI. Überlegen Sie, welcher Datentyp sich dafür am besten eignen. ✍️ Behandeln Sie im Dossier sich ergebende Vor- und Nachteile.
4. Für **Gäste** muss ein 'Ablaufdatum' angegeben werden. Standardmäßig ist das eine Woche in der Zukunft. Der 'Grund' wird bis zu 254 Zeichen lang.
5. 'Vorrat' gibt bei **Mahlzeiten** an, wie oft die Speise heute noch verfügbar ist. (eine Ganzzahl und standardmäßig 0, nicht **NULL**). 'Verfügbar' ist ein Flag, das angibt ob die Speise bestellbar ist.
6. **Deklarationen** enthalten ein 'Zeichen', das aus ein oder zwei Buchstaben bestehen kann und eine 'Beschriftung', die maximal 32 Zeichen lang sein darf.
7. Die 'ID' bei **Zutaten** ist immer eine fünfstellige Zahl und soll nicht automatisch hochgezählt werden.
8. Die Preisangaben in **Preise** werden den Wert 99,99 nicht übersteigen und sind immer positiv. Der 'Gastpreis' muss immer angegeben sein, denn er gilt letztlich, wenn einem Kunden keine Vergünstigung zusteht. Der 'Studentpreis' muss immer günstiger als der 'MA-Preis' sein. Preise für eine Mahlzeit gelten für ein 'Jahr'.
9. Der 'Endpreis' in **Bestellungen** kann natürlich größer werden. 'Bestellzeitpunkt' ist im Standard jetzt und der 'Abholzeitpunkt' muss, wenn er angegeben wird, später als der Bestellzeitpunkt sein.
10. Die Attribute 'Nutzer'. 'Aktiv', 'Zutaten'. 'Vegetarisch', 'Zutaten'. 'Vegan', 'Zutaten'. 'Glutenfrei' und 'Zutaten'. 'Bio' sind Flags, erlauben Sie also nur 1 oder 0.

Constraints

- Nutzen Sie die Möglichkeit *benannter Constraints*, wie im Beispiel für "KfzEindeutig" gezeigt:

```
CREATE TABLE Kfz
(
    kennzeichen varchar(10) NOT NULL,
    CONSTRAINT KfzEindeutig UNIQUE (kennzeichen)
);
```

So ist es leichter, Constraints nachträglich zu löschen oder zu ändern.

Aufgabe 2.2 : Relationen umsetzen

Prinzipiell kann jede Relation in eine eigene Tabelle überführt werden, mit je einem Fremdschlüssel für die verknüpften Entitätstypen. Überführen Sie die Relationen in die Datenbank.

Welche Relationen lassen sich ohne eigene Tabelle abbilden bzw. welche nicht? Welche Vorteile bietet jene Vorgehensweise?

Erweitern Sie die passenden `CREATE TABLE` Statements der Tabellen um `FOREIGN KEY` Constraints⁴.

Spezialisierung

Eine Spezialisierung (z.B. für Nutzer {FH-angehörige, Gäste}) ist auch mit `FOREIGN KEY` umzusetzen. Was müssen Sie ändern, um diese besondere Beziehung abzudecken? Welche Vor-/Nachteile hat die von Ihnen ausgewählte Abbildung? Gibt es Anweisungen im ER-Diagramm, die Sie nicht vollständig abbilden können?

Legen Sie die Integritätsbedingungen für die nötigen Kreuztabellen (N:M) selbst fest, wenn Sie nicht aus den [Business Rules](#) oder dem ER-Diagramm hervorgehen.

Aufgabe 2.3 : Kaskaden

- Legen Sie per `INSERT` Befehle vier Benutzer an. Spezialisieren Sie einen Nutzer zu Mitarbeiter und zwei zu Student.
- Setzen Sie für spezialisierte Nutzer das kaskadierende Löschen⁴ um. Es muss also möglich sein, aus der Tabelle **Nutzer** Einträge zu löschen, obwohl sie zuvor spezialisiert wurden.
- Wie müssen die spezialisierten Tabellen abgeändert werden, um kaskadierendes Löschen abzubilden? Setzen Sie dies um.
- Testen Sie Ihre Änderung, indem Sie einen FH-Angehörigen löschen. Z.B. so:

```
-- Löschen des Nutzers mit der Nummer 4
DELETE FROM `Nutzer` WHERE Nummer=4;
```

Wenn das dazu führt, dass auch die spezialisierten Studenten und Mitarbeiter löschar sind, haben Sie die Löschkaskade korrekt umgesetzt.

Weitere Business Rules

Bilden Sie die folgenden Business Rules in Ihrem SQL Script ab. Diese sollen Sie per `ALTER` -Anweisungen vornehmen. Dazu müssen Sie eventuell Constraints vorher löschen.

1. Wird eine Mahlzeit gelöscht sollen alle dazu gespeicherten Preise auch gelöscht werden. Die zugehörigen Zutaten und Kommentare werden nicht gelöscht, sondern verlieren lediglich die Zuordnung.
2. Wenn eine Kategorie gelöscht wird, sollen eventuell betroffene Kindkategorien nicht betroffen werden, diese verlieren einfach die Oberkategorie.
3. Wenn ein Bild gelöscht wird, sollen eventuell betroffene Kategorien nicht gelöscht werden, diese verlieren einfach ihr Bild.







Optional als Vorbereitung für Paket 3

Fügen Sie bereits ein paar Mahlzeiten ein. Sie können die Bild-Daten zunächst ignorieren.

Im Ordner **Praktikum/Meilenstein 2** befinden sich mit der Zeit Beispieldaten.

In´s Dossier

Dokumentieren Sie im Dossier allgemein, ...

-  was das Semikolon am Ende einer Anweisung bewirkt
-  wie Sie die binären Relationstypen (1:1, 1:N, N:M) abgebildet haben
-  den Unterschied zwischen Tabellen- und Spalten-Constraints und wann welche Art sinnvoll ist
-  wie Sie den Aufzählungsdatentyp ENUM, den MariaDB unterstützt, per CHECK Constraint auch in anderen DBMS nachbilden könnten
-  welche Constraints in MariaDB welchem Zweck dienen
-  wieso Sie die Datenbank `information_schema` sehen

Backups

Erneut: Bitte machen Sie regelmäßige Backups Ihres SQL-Skripts, da HeidiSQL gerne auch mal einfriert. Speichern Sie mit **STRG+S** ;) Mit diesem Skript sollten Sie jederzeit die gesamte Datenbank erneut anlegen können.

Lernziele

Nach Bearbeitung des Pakets haben Sie...




- einen SQL Client (z.B. HeidiSQL) kennengelernt
- ein gegebenes ER-Diagramm in ein relationales Schema überführt und können abschätzen, inwiefern dessen Vorgaben im DBMS umsetzbar sind
- Tabellen mit SQL DDL Befehlen erstellt und können erklären...
 - ... wie generalisiert bzw. spezialisiert wird
 - ... wie optionale Attribute, Relationen mit Attributen und PK und FK definiert werden können
 - ... welchen Zweck Constraints haben und wie sie eingesetzt werden
- einen Einblick in die Datentypen von MariaDB erhalten und Informationen aus `information_schema` entnommen

FH Aachen Fachbereich Elektrotechnik und Informationstechnik

Marcel Remmy
Büro H215
Eupener Straße 70
52064 Aachen

✉ | remmy@fh-aachen.de
🌐 | <http://fh-aachen.de/remmy>
🕒 | Sprechstunde nach Vereinbarung

Referenzen

1. MariaDB KB <https://mariadb.com/kb/en/library/sql-statements-structure/> 
2. Datentypen in MariaDB <https://mariadb.com/kb/en/the-mariadb-library/data-types/> 
3. Datentypen SQL https://www.w3schools.com/sql/sql_datatypes.asp 
4. FK Constraint <https://mariadb.com/kb/en/library/foreign-keys/> 