SQL – Daten Abfragen

Einstieg

Die Structured Query Language (SQL) ist eine standardisierte Datenbanksprache zur Nutzung auf relationalen Datenbanksystemen wie MySQL oder MariaDB. Mit SQL können Abfragen, Manipulationen und Änderungen auf bzw. an den Tabellen der Datenbank vorgenommen werden.

Anmelden am Webinterface

Wir haben ein zentrales DBMS zur Benutzung durch die Schülerinnen und Schüler eingerichtet:

- info-bw Datenbank Server
- KMG Friedrichshafen

Aufgabe: Anmeldung an mysql

- Öffne mit deinem Browser das phpmyadmin Interface: https://dbms.kmg-fn.de
- Infos zu Benutzern und Passwort bekommst du im Unterricht.

Aufgabe: Anmeldung an mysql

- Öffne mit deinem Browser das phpmyadmin Interface: https://sql.info-bw.de
- Infos zu Benutzern und Passwort bekommst du im Unterricht/in der Fortbildung.

Datenbankimport

Aufgabe: Import der Datenbank

- Lade die Datei
  adressen.sql.zip
  auf deinen Computer herunter.
- Importiere die Datei in deine Datenbank. Du musst dabei darauf achten, dass du **in der linken Spalte zuerst deine Datenbank auswählst**, andernfalls scheitert der Import mit einem Fehler, da keine neuen Datenbanken angelegt werden können.

Du erhältst eine Tabelle: `adressen` in deiner Datenbank, in der ca. 20.000 Datensätze enthalten sind.

Klicke die Tabelle an und mache dich mit den gespeicherten Daten etwas vertraut.

Aufbau einer SQL Abfrage

Die Sprache SQL (Structured Query Language) wird weltweit gesprochen und (fast) jede (relationale) Datenbank kann damit abgefragt werden. SQL ähnelt etwas dem gesprochenen Englisch und wird in den Abfragen in der Regel mit Großbuchstaben geschrieben, damit Befehle besser von Argumenten unterscheiden kann. Es funktionieren allerdings auch kleingeschriebene Befehle. Jede SQL Abfrage muss mit einem Semikolon (aka Strichpunkt) abgeschlossen werden[1)].

Eine Klausel für SQL-Abfragen ist im Wesentlichen folgendermaßen aufgebaut:

```
SELECT [DISTINCT] { spalten | * }
FROM tabelle [alias] [,tabelle [alias]] ...
[WHERE {bedingung}]
[GROUP BY spalten [HAVING {bedingung}]]
[ORDER BY spalten [ASC | DESC]];
```

Argumente in eckigen Klammern sind dabei optional, die Schreibweise `{ spalten | * }` bedeutet eine der durch den | getrennte Möglichkeiten, also entweder eine Liste der zu wählenden Spalten oder das Sternchen für alle Spalten.

Ausführungsreihenfolge

```
SELECT (Spaltenauswahl bzw. Projektion)
FROM (Tabellenauswahl)
-> WHERE (Zeilenauswahl bzw. Selektion)
-> GROUP BY (Gruppierung)
-> HAVING (Gruppenauswahl)
-> ORDER BY (Sortierung)
```

Beispiele

Die SQL-Abfrage

```
SELECT * FROM adressen WHERE Hausnummer > 100
```

liefert eine Tabelle aller Datensätze, bei denen die Hausnummer größer als 100 ist. Die Wildcard nach dem SELECT Statement * steht für alle Tabellenspalten.

Die Abfrage

```
SELECT * FROM adressen WHERE Id = 10
```

Ermittelt den Datensatz mit der Id 10.

Aufgabe: Abfragen

Führe die beiden Beispielabfragen auf deiner Datenbank aus.

Fehlerprotokoll

Du wirst zu Beginn ziemlich sicher zahlreiche falsche Eingaben machen, die zu Fehlern führen, die nicht immer direkt verständlich sind. Um etwas Struktur in deine persönlichen Fehler zu bringen, solltest du ein Fehlerprotokoll führen, so dass du erkennen kannst, wenn du ähnliche Fehler immer wieder machst.

Du kannst die folgende Datei verwenden, um die Fehler direkt hinein zu kopieren und zu vermerken, wie du sie lösen kannst. Alternativ kannst du aber natürlich auch dein Notizprogramm verwenden.

fehlerprotokoll.odt

Fehlerprotokoll

Notieren Sie immer dann, wenn MySQL Ihnen eine Fehlermeldung liefert, die eingegebene Abfrage und die Fehlermeldung, die ausgegeben wurde. Versuchen Sie, herauszufinden, was der Grund für die Fehlermeldung war. Beschreiben Sie, wie man den Fehler in Zukunft vermeiden kann.

SQL-Abfrage	Fehlermeldung der Datenbank	Grund	Lösung (ggf. allgemeine Regel)
Beispiel: `SELET * FROM myTable`	`#1064 - You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'SELET * FROM myTable' at line 1`	Schlüsselwort `SELECT` wurde falsch geschrieben	Auf Schreibweise der Schlüsselwörter achten.

(A1)

Arbeite mit der Befehlsübersicht A in der Tabelle `adressen`. Filtere mit einer geeigneten SQL-Abfrage die folgenden Informationen aus der Tabelle heraus.

1. Alle Name, Vorname und Wohnort aller Datensätze.
2. Die Vor- und Nachnamen aller Einträge.
3. Alle Postleitzahlen und Orte, deren PLZ größer oder gleich 80000 ist.
4. Vor- und Nachname aller Personen, die im Postleitzahlbereich 4xxxx bis 6xxxx wohnen.
5. Alle Datensätze, mit den Nachnamen Maier, Mayer, Meier, Meyer oder Müller.
6. Alle Personen, die mit "Herr" angesprochen werden und älter als 80 Jahre sind.
7. Alle Einträge die neuer als 10 Jahre sind.
8. Alle Personen, die in einer Stadt wohnen, die ein "x" enthält.
9. Alle Einträge mit Vornamen, die mit "Am" beginnen.
10. Alle Personen mit einem Faxgerät und einem Mobiltelefon, deren Nachnamen ein "y" enthält.
11. Alle Personen, die jünger als 45 Jahre sind, in einem Ort wohnen der mit "A" beginnt und mehr

als 3333 Bonuspunkte gesammelt haben.

Lösungen

(A2)

Denke dir mindestens 2 weitere Kriterien für die Filterung von Adressen aus, notiere diese und erstelle eine passende SQL Abfrage.

(A3)

Arbeite jetzt mit Befehlsübersicht B in der Tabelle `adressen`. Man kann mit den Werten, die von einer SELECT-Abfrage zurückgegeben werden auch rechnen (lassen):

```
SELECT BBestand FROM buecher
```

BBestand
5
17
100
0
20
...

```
SELECT 2*BBestand FROM buecher
```

2 * BBestand
10
34
200
0
40
...

Filtere mit einer geeigneten SQL-Abfrage die folgenden Informationen aus der Tabelle heraus.

1. Alle Personen, die mehr als 15 offene Rechnungen haben.
2. Die Anzahl der Personen, die mehr als 10 offene Rechnungen haben.
3. Die Anzahl der Personen, die mehr als 19 offene Rechnungen haben und kein Guthaben auf ihrem Konto.
4. Alle Vornamen, Nachnamen, Wohnorte und Kontostände bei denen der Kontostand mehr als 50EUR beträgt, aufsteigend Sortiert nach dem Kontostand und nach dem Nachnamen
5. Alle Postleitzahlen in der Datenbank
6. Alle Geburtsdaten, die in der Datenbank vorkommen, ohne die Datensätze, bei deinen kein Geburtsdatum angegeben ist

7. Den Datensatz mit dem höchsten Kontostand
8. Die durchschnittliche Zahl der Bonuspunkte aller Personen aus Aachen die am Bonusprogramm teilnehmen.
9. Die durchschnittliche Zahl der offenen Rechnungen aller Personen, die jünger als 35 Jahre sind und nicht am Bonusprogramm teilnehmen.

(A4)

Überlege dir 5 weitere verschachtelte Abfragen mit mehreren Kriterien, welche die Befehle aus Blatt B verwenden.

(Bonus)

Möglicherweise ist hier eine kleine Recherche von nöten.

1. Eine Liste aller Mailprovider
2. Eine Liste aller Vorwahlen

Lösungshinweis I

Man muss den Inhalt einer Tabellenzelle aufteilen. Das geht zum Beispiel mit SELECT SUBSTRING_INDEX, Infos dazu z.B. dort:
https://www.w3schools.com/sql/func_mysql_substring_index.asp

> **Hinweis**: Wir werden später sehen, dass ein solches Vorgehen eigentlich vermieden werden sollte - man möchte für gewöhnlich die Informationen so in der Tabelle ablegen, dass man die Inhalte nicht manipulieren muss, um an die Informationen zu gelangen.

Lösungshinweis II

Für die Maildomains sieht de Lösung so aus:

```
SELECT DISTINCT SUBSTRING_INDEX(EMail, "@", -1) FROM adressen;
```

Die Lösung für die Vorwahlen kannst du sicher selbst finden.

Material

02-sql-select.odp	1.5 MiB	05.11.2020 11:32
02-sql-select.pdf	744.2 KiB	05.11.2020 11:32
2024-11-22_12-02.png	61.1 KiB	22.11.2024 11:03
adressen.sql.zip	1.5 MiB	05.11.2020 11:44
adressen.sqlite.db.zip	1.6 MiB	24.11.2024 16:38
fehlerprotokoll.odt	13.7 KiB	22.11.2024 11:04
lsg_i_a1.zip	998.0 B	16.02.2022 14:07
sql_befehle_a.pdf	223.0 KiB	14.11.2019 11:57
sql_befehle_b.pdf	112.1 KiB	21.10.2022 09:03
sql_rechnen.png	22.9 KiB	14.11.2019 11:58
webshop.sql.zip	2.2 KiB	14.11.2019 11:57

1)

Im SQL Befehlsfenster in phpmyadmin kann auf den Strichpunkt verzichtet werden, nicht aber auf der mysql-Kommandozeile oder bei der Abfrage aus PHP o.ä.