7 Daten auswählen, ändern und löschen

In diesem Kapitel erfahren Sie, wie Sie die gewünschten Daten gezielt suchen bzw. finden und ändern bzw. mutieren oder bei Bedarf wieder löschen können.

Hinweis

▷ Auch hier gilt: Die dafür benötigten Anweisungen können lang und fehleranfällig sein. Aus diesem Grund wird im Folgenden **MySQL Workbench** eingesetzt. Damit können Sie Daten einfach und bequem selektieren, ändern und bei Bedarf wieder löschen.

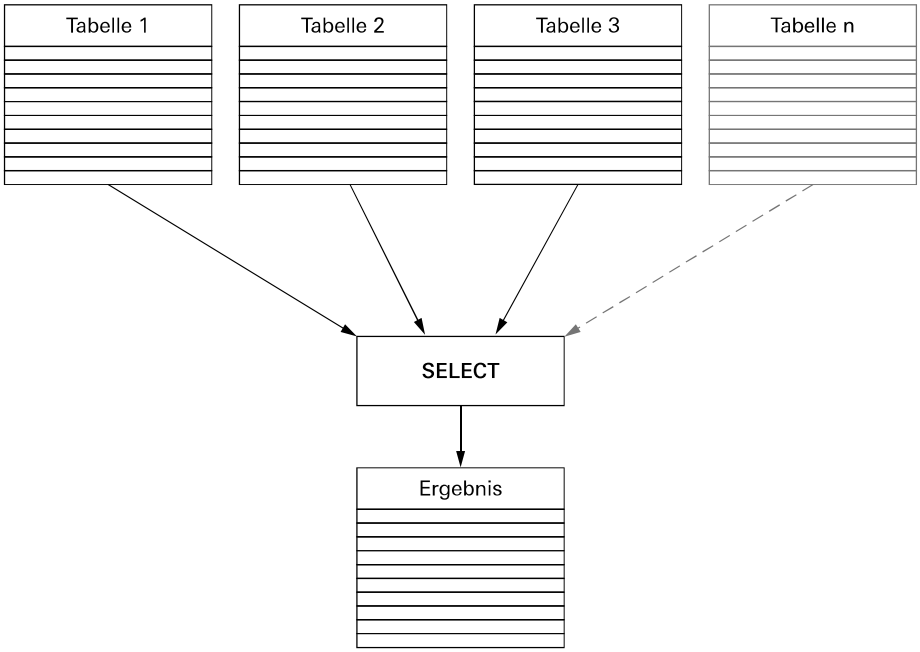
Für die Bearbeitung unseres Fallbeispiels empfehlen wir Ihnen, an dieser Stelle des Lehrmittels die vorbereitete Datenbank herunterzuladen. Gehen Sie dazu wie folgt vor:

1. Besuchen Sie die Website www.compendio.ch/Bildungsmedien/Downloads.asp?Titel=1610, laden Sie die Datei **Videoverwaltung_fertig.sql** herunter und speichern Sie diese im Verzeichnis **C:\MySQL** bzw. in dem Verzeichnis, das Sie für diesen Zweck angelegt haben (vgl. Kap. 6.3, S. 65).
2. Starten Sie MySQL und erzeugen Sie mit dem Befehl `mysql> CREATE DATABASE Videoverwaltung_fertig;` die neue Datenbank.
3. Lesen Sie die heruntergeladene Datei **Videoverwaltung_fertig.sql** mit dem Befehl `mysql> SOURCE C:\MySQL\Videoverwaltung_fertig.sql` in die neue, leere Datenbank «Videoverwaltung_fertig» ein.

7.1 Daten selektieren

Für das Suchen und Auffinden von Daten spielt die Anweisung SELECT eine zentrale Rolle. Mithilfe dieser Anweisung wird eine **Abfrage an die Datenbank** gerichtet. In der einfachsten Form betrifft diese Datenabfrage eine einzelne Tabelle. Oft werden aber Daten aus mehreren Tabellen gewünscht (z. B. die Titel aller Filme, die ein bestimmter Kunde bezogen hat). Die Anfrage richtet sich dann gleichzeitig an mehrere Tabellen. In solchen Fällen wirkt die Anweisung SELECT wie ein Filter. Dieses Prinzip lässt sich wie folgt darstellen:

[7-1] Funktionsprinzip der Anweisung SELECT



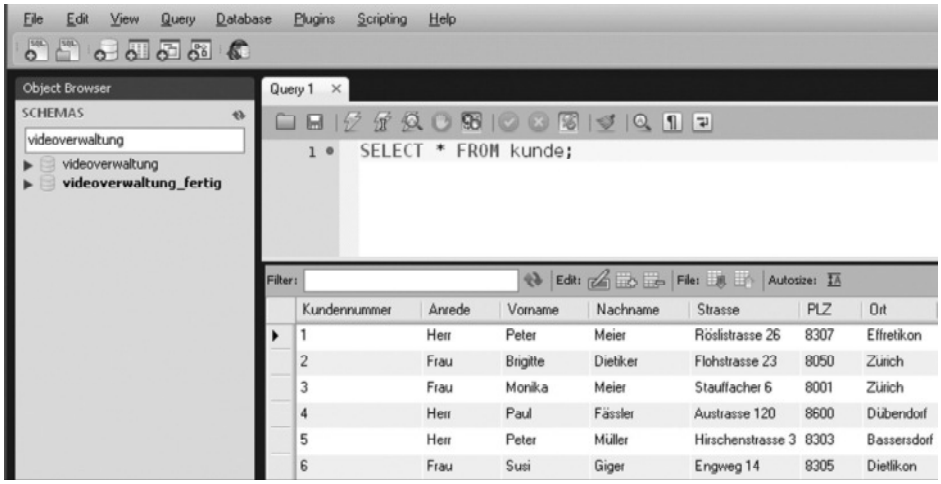
7.2 Spalten auswählen

Der Grundaufbau der Anweisung SELECT besteht aus den Namen der Spalten, die aufgelistet werden sollen, und dem Namen der Tabelle, die durchsucht werden soll. Sollen alle Spalten einer Tabelle aufgelistet werden, wird anstelle der Spaltennamen ein * verwendet:

```
SELECT * FROM Kunde;
```

Um diese Anweisung in der Benutzeroberfläche **MySQL Workbench** einzugeben, müssen Sie zuerst die entsprechende Datenbank mit einem Doppelklick auswählen (hier: Videoverwaltung_fertig) und danach das obere Teilfenster anklicken. Hier können Sie obige Anweisung eingeben und danach mit dem Befehl «Execute» oder mithilfe der Tastenkombination [Ctrl] + [Enter] ausführen. Nach der Ausführung sehen Sie im unteren Teilfenster der Benutzeroberfläche MySQL Workbench das Resultat der Abfrage. Im folgenden Screenshot sehen Sie beispielhaft alle Daten der Tabelle KUNDE:

[7-2] Benutzeroberfläche MySQL Workbench



Im Folgenden erfahren Sie, wie Sie bestimmte Daten aus einzelnen oder mehreren Tabellen auswählen und gruppieren, sortieren oder auswerten können.

7.2.1 Einzelne Spalten auflisten

Sollen nur bestimmte Spalten einer Tabelle aufgelistet werden, so müssen Sie diese in der Anweisung SELECT explizit aufführen. Beispiel:

```
SELECT Vorname, Nachname FROM Kunde;
```

7.2.2 Einzelne Zeilen auflisten

Mit der sog. Klausel WHERE können Sie gezielt Zeilen auswählen, die bestimmte Kriterien erfüllen. Die folgenden Beispiele zeigen auf, wie vielfältig die Klausel WHERE eingesetzt werden kann. Sie möchten die Tabelle KUNDE nach allen Kunden aus Bassersdorf abfragen:

```
SELECT * FROM Kunde WHERE Ort = 'Bassersdorf';
```

Sie möchten die Tabelle KUNDE nach den Vornamen und Nachnamen der Kunden aus Kloten abfragen:

```
SELECT Vorname, Nachname FROM Kunde WHERE Ort = 'Kloten';
```

Sie möchten die Tabelle FILM nach den Titeln abfragen, die mehr als CHF 30.– gekostet haben:

```
SELECT Titel, EPreis FROM Film WHERE Epreis > 30;
```

Sie möchten die Tabelle FILM nach den Titeln der Kategorie Krimi abfragen, die mehr als CHF 30.– gekostet haben:

```
SELECT Titel, Kategorie, EPreis FROM Film
WHERE Epreis > 30 AND Kategorie = 'KRI';
```

Sie möchten die Tabelle FILM nach den Titeln der Kategorien Krimi und Actionfilm abfragen:

```
SELECT Titel, Kategorie FROM Film
WHERE Kategorie = 'KRI' OR Kategorie = 'ACT';
```

Sie möchten die Tabelle KUNDE nach allen Kunden abfragen, deren Nachname mit der Buchstabenkombination «Me» beginnt:

```
SELECT Vorname, Nachname FROM Kunde
WHERE Nachname LIKE 'Me%';
```

Hinweis

▷ Im obigen Beispiel wurde das Zeichen % verwendet. Dieses «Jokerzeichen» steht für beliebig viele Zeichen oder keines.

Sie möchten die Tabelle KUNDE nach allen Kunden abfragen, die den Nachnamen «Meier» oder «Meyer» tragen:

```
SELECT Vorname, Nachname FROM Kunde
WHERE Nachname LIKE 'Me_er';
```

Hinweis

▷ Ist in einer Zeichenkette ein einzelnes Zeichen unbekannt, so kann als Jokerzeichen das Zeichen _ verwendet werden.

7.2.3 Zeilen sortieren

In der Praxis werden Listen oft alphabetisch, numerisch oder chronologisch sortiert. Die folgenden Beispiele sollen aufzeigen, wie das entsprechende Prädikat ORDER BY funktioniert.

Sie möchten die Kunden nach Nachnamen aufsteigend sortieren:

```
SELECT Vorname, Nachname, Ort FROM Kunde
ORDER BY Nachname;
```

Sie möchten die Kunden nach Nachnamen absteigend (englisch: descending) sortieren:

```
SELECT Vorname, Nachname, Ort FROM Kunde
ORDER BY Nachname DESC;
```

Sie möchten die Kunden nach Ortschaft und anschliessend nach Nachnamen aufsteigend sortieren:

```
SELECT Vorname, Nachname, Ort FROM Kunde
ORDER BY Ort, Nachname;
```

Sie möchten die Kunden aus Kloten nach Nachnamen sortieren:

```
SELECT Vorname, Nachname FROM Kunde
WHERE Ort = 'Kloten'
ORDER BY Nachname;
```

7.2.4 Zeilen gruppieren und auswerten

Oft müssen Daten nach einem bestimmten Merkmal gruppiert und gesamthaft ausgewertet werden. Im Rahmen unserer Beispiel-Datenbank möchten wir herausfinden, wie viele Filme jede Filmkategorie enthält. Dies wird mit dem Prädikat GROUP BY und der Funktion COUNT() erreicht. Die entsprechende Abfrage lautet wie folgt:

```
SELECT Kategorie, COUNT(Videonummer) FROM Film
GROUP BY Kategorie;
```

Sie möchten die Anzahl der DVDs pro Kategorie ermitteln:

```
SELECT Kategorie, SUM(Lagerbestand) FROM Film
GROUP BY Kategorie;
```

In der folgenden Tabelle finden Sie die wichtigsten Funktionen, die Sie mit gruppierten Daten verwenden können:

Funktion	Beschreibung
COUNT()	Zählt die Anzahl Zeilen in einer Gruppe.
SUM()	Addiert die Werte einer Spalte.
AVG()	Ermittelt den Mittelwert einer Spalte.
MIN()	Ermittelt den kleinsten Wert einer Spalte.
MAX()	Ermittelt den grössten Wert einer Spalte.

7.2.5 Daten aus mehreren Tabellen auswählen

Wir haben bereits gesehen, dass die Daten einer relationalen Datenbank in verschiedenen Tabellen gespeichert werden. Dabei werden die Daten in den Zeilen jeder Tabelle durch den Primärschlüssel eindeutig identifiziert. Diesen Sachverhalt wollen wir uns vergegenwärtigen, wenn wir Daten selektieren möchten, die in mehreren Tabellen abgespeichert sind. Sie möchten z. B. eine Liste mit allen ausgeliehenen Filmen und folgenden Angaben: dem Ausleihdatum, dem Vornamen und dem Nachnamen des jeweiligen Kunden. Für diesen Fall wäre folgende Anweisung naheliegend:

```
SELECT Vorname, Nachname, Ausleihe FROM Kunde, Ausleihe;
```

Leider liefert diese Anweisung nicht das gewünschte Resultat, sondern eine Unmenge von Daten. Sie erhalten nämlich eine Liste mit allen Zeilen aus der Tabelle KUNDE sowie der Tabelle AUSLEIHE. Der Grund: MySQL «weiss» nicht, wie die Zeilen der Kundentabelle mit den Zeilen der Ausleihtabelle in Beziehung stehen. Richtig lautet die gewünschte Abfrage folgendermassen:

```
SELECT Vorname, Nachname, Ausleihe FROM Kunde, Ausleihe
WHERE Kunde.Kundennummer = Ausleihe.Kundennummer;
```

In der obigen Anweisung wird klar festgehalten, dass nur die Zeilen aufgelistet werden sollen, bei denen die Kundennummern der beiden Tabellen übereinstimmen. Wenn Sie zusätzlich wissen wollen, welche Titel die Kunden jeweils bezogen haben, müssen Sie die Tabelle FILM in die Abfrage integrieren und deren Beziehung zur Tabelle AUSLEIHE definieren. Die entsprechende Anweisung sieht wie folgt aus:

```
SELECT Vorname, Nachname, Ausleihe, Titel FROM Kunde, Ausleihe, Film
WHERE Kunde.Kundennummer = Ausleihe.Kundennummer
AND Film.Videonummer = Ausleihe.Videonummer;
```

Die Frage «Welche Filme hat Otto Meier bis jetzt bezogen?» lässt sich wie folgt beantworten:

```
SELECT Vorname, Nachname, Ausleihe, Titel FROM Kunde, Ausleihe, Film
WHERE Kunde.Kundennummer = Ausleihe.Kundennummer
AND Film.Videonummer = Ausleihe.Videonummer
AND Vorname = 'Otto'
AND Nachname = 'Meier';
```

7.2.6 Abkürzungen verwenden

Ein **«Alias»**[1] ist eine Abkürzung, die eingesetzt werden kann, um Zeit zu sparen und eine Anweisung kürzer und übersichtlicher zu halten. Solche Abkürzungen können etwa den Namen einer Tabelle oder Spalte ersetzen. Folgende Anweisung kennen Sie bereits:

```
SELECT Vorname, Nachname, Ausleihe, Titel FROM Kunde, Ausleihe, Film
WHERE Kunde.Kundennummer = Ausleihe.Kundennummer
AND Film.Videonummer = Ausleihe.Videonummer
AND Vorname = 'Otto'
AND Nachname = 'Meier';
```

Diese Anweisung können Sie nun z. B. abkürzen, indem Sie die Tabellen KUNDE, AUSLEIHE und FILM durch die «Aliases» K, A und F ersetzen. Die entsprechende Anweisung lautet nun wie folgt:

```
SELECT Vorname, Nachname, Ausleihe, Titel
FROM Kunde AS K, Ausleihe AS A, Film AS F
WHERE K.Kundennummer = A.Kundennummer
AND F.Videonummer = A.Videonummer
AND Vorname = 'Otto'
AND Nachname = 'Meier';
```

Hinweis

▷ Das Prädikat AS kann bei der Festlegung der Aliases auch weggelassen werden. Möglicherweise wird dadurch aber die Lesbarkeit der Anweisung erschwert.

Wenn Sie mit Funktionen arbeiten, können Aliases ebenfalls nützlich sein. In der folgenden Anweisung wird z. B. die automatisch berechnete Spalte mit der Summe des Lagerbestands «TotalLB» genannt. Bei der Sortierung kann auf diesen Namen Bezug genommen werden:

```
SELECT Kategorie, SUM(Lagerbestand) AS TotalLB FROM Film
GROUP BY Kategorie
ORDER BY TotalLB DESC;
```

7.3 Daten mutieren

Für die Mutation bzw. Änderung von Daten spielt die Anweisung UPDATE eine zentrale Rolle. Mit UPDATE wird eine **Mutationsanweisung** an die Datenbank gerichtet. In unserem Fall stellt sich beispielsweise heraus, dass eine als «Frau Lea Zumstein» erfasste Kundin in Wirklichkeit «Leanna Zumstein» heisst. Die entsprechende Änderungsanweisung lautet wie folgt:

```
UPDATE Kunde SET Vorname = 'Leanna'
WHERE Kundennummer = 7;
```

Um zu prüfen, ob die Daten korrekt mutiert wurden, können Sie sich mittels SELECT-Befehl die geänderten Daten anzeigen lassen.

[1] Dieser Begriff stammt aus dem Lateinischen und bedeutet wörtlich «sonst». Damit ist in unserem Zusammenhang eine verkürzte, gleichbedeutende Anweisung gemeint.

7.4 Daten löschen

Für das Löschen von Daten spielt die Anweisung DELETE eine zentrale Rolle. Mit DELETE wird eine **Löschanweisung** an die Datenbank gerichtet. Im Gegensatz zur Anweisung DROP TABLE löscht DELETE nur bestimmte Zeilen und nicht die gesamte Tabelle. In unserem Fall stellt sich beispielsweise heraus, dass die erste Ausleihe falsch erfasst wurde und wieder gelöscht werden muss. Bevor Sie dies tun, lassen Sie sich mit folgendem Befehl die älteste Ausleihe (MIN) mit vorhandener Rückgabe (IS NOT NULL) anzeigen:

```
SELECT Ausleihnr, Kundennummer, Videonummer, MIN(Ausleihe), Rueckgabe
FROM Ausleihe WHERE Rueckgabe IS NOT NULL;
```

Diese Anweisung bestätigt die Vermutung bezüglich der zu löschenden Ausleihe (Annahme: die 1. Ausleihe wurde nur testhalber erfasst). In der Folge löschen Sie die Ausleihe mit der Nummer 1 durch diesen Befehl:

```
DELETE FROM Ausleihe
WHERE Ausleihnr = 1;
```

Wenn Sie nun den Befehl SELECT MIN erneut anwenden, wird die Ausleihnummer nicht mehr angezeigt.

Mit der SQL-Anweisung SELECT wird eine **Datenabfrage** an die Datenbank gerichtet (z. B. die Titel aller Filme, die ein bestimmter Kunde bezogen hat). Eine solche Anfrage richtet sich an einzelne oder mehrere Tabellen und wirkt wie ein Filter.

Die **Select-Anweisung** besteht aus den Bezeichnungen (Namen) der Tabellenspalten, die aufgelistet werden sollen, und dem Namen der Tabelle, die durchsucht werden soll. Sollen alle Spalten einer Tabelle aufgelistet werden, wird anstelle der Spaltenbezeichnungen der Platzhalter * verwendet. Mithilfe der Select-Anweisung können zudem bestimmte Daten bzw. Tabellenzeilen sortiert, gruppiert und bei Bedarf ausgewertet werden. Daten aus mehreren Tabellen lassen sich zusammenziehen, indem in der Select-Anweisung die Beziehung zwischen den betroffenen Tabellen definiert wird. Weiter ist es möglich, längere Select-Anweisungen mithilfe von **Aliases** zu vereinfachen und übersichtlich zu gestalten.

Mit der **Update-Anweisung** lassen sich bestimmte Datensätze einfach und rasch ändern bzw. mutieren. Mit der **Delete-Anweisung** können bestimmte Datensätze endgültig gelöscht werden.

Unter der Benutzeroberfläche **MySQL Workbench** wird zuerst die Datenbank ausgewählt und danach das obere Teilfenster angeklickt. Nun kann die gewünschte Anweisung eingegeben und mit dem Befehl «Execute» oder mithilfe der Tastenkombination [Ctrl] + [Enter] ausgeführt werden. Nach der Ausführung wird im unteren Teilfenster das Resultat der Datenabfrage angezeigt.