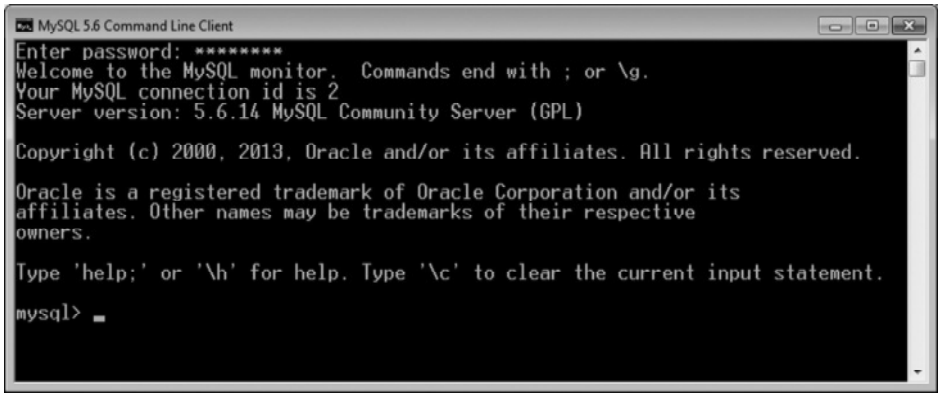
Nach dem Start von Windows wird der **MySQL-Server** automatisch gestartet. Um nun mit MySQL arbeiten zu können (Datenbank anlegen, öffnen, Tabellen erstellen, Abfragen formulieren usw.), müssen Sie noch ein Clientprogramm starten.

Für die ersten Schritte arbeiten wir mit dem mitgelieferten **Kommandozeileneditor**. Je nach installierter Version bzw. Edition von MySQL wird der Editor unterschiedlich gestartet. Häufig finden Sie – in derselben MySQL-Gruppe wie die grafische Benutzeroberfläche namens «MySQL Workbench» – ein Programm mit der Bezeichnung «MySQL Command Line Client». Starten Sie das Programm «MySQL Command Line Client» und geben Sie das (während der Installation vergebene) Passwort bei der entsprechenden Aufforderung ein.

Damit haben Sie sich unter dem Benutzernamen **root** beim MySQL-Server angemeldet. Root ist hier standardmässig der sog. Datenbankadministrator. In einer produktiven Umgebung werden weitere Benutzer mit eingeschränkten Zugriffsrechten definiert. In der Folge erscheint der Kommandozeileneditor von MySQL:

[6-2] Kommandozeileneditor von MySQL



Hinweis

▷ Sie haben bereits gesehen, dass für verschiedene Bestandteile einer Datenbank unterschiedliche Bezeichnungen bestehen (s. Kap. 5, S. 42). Solange im Rahmen dieses Lehrmittels SQL eingesetzt wird, arbeiten wir mit den Begriffen **Datenbank, Tabellen, Zeilen** und **Spalten.**

6.2 Datenbank erzeugen und aktivieren

Bevor Sie die einzelnen Tabellen der Datenbank und deren Spalten definieren können, müssen Sie dafür einen «Behälter» erstellen. Dieser Behälter stellt die künftige Datenbank dar. Geben Sie dazu im Kommandozeileneditor die Anweisung für die **Erstellung der Datenbank** mit der zugehörigen Bezeichnung an. Für unser Beispiel lautet diese Anweisung wie folgt:

```
mysql> CREATE DATABASE Videoverwaltung;
```

Wie der Name sagt, werden im Kommandozeileneditor von MySQL Kommandozeilen[1] erfasst. Dabei müssen Sie die vorgegebenen **Anweisungen bzw. Befehle der Daten-**

[1] Der Begriff «Kommando» stammt aus dem Lateinischen und bedeutet «Befehl».

banksprache SQL verwenden (hier: CREATE DATABASE). Beachten Sie dabei folgende Punkte:

- Es spielt keine Rolle, ob Sie SQL-Anweisungen gross- oder kleinschreiben. Zur besseren Unterscheidung werden die Befehle im vorliegenden Lehrmittel durchgängig grossgeschrieben.
- Jede Anweisung wird durch einen Strichpunkt abgeschlossen.
- Um den Befehl auszuführen, müssen Sie nach jeder Anweisung die [Enter]-Taste drücken.

Die erfolgreiche Ausführung eines Befehls wird mit «Query OK» bestätigt. Darunter wird das **MySQL-Prompt** angezeigt:

```
Query OK, 1 row affected (0.00 sec)
mysql>
```

Damit Sie nun mit der erzeugten Datenbank arbeiten können und z. B. Tabellen erstellen oder Attribute erfassen können, müssen Sie diese aktivieren. Die **Aktivierung einer Datenbank** erfolgt mittels Anweisung USE. In unserem Beispiel lautet der Befehl wie folgt:

```
mysql> USE Videoverwaltung;
Database changed
mysql>
```

6.3 Tabellen erstellen, ändern und löschen

Die Datenbank ist erzeugt. Sie können nun die Tabellen erstellen, die zu dieser Datenbank gehören. Dies kann beispielsweise über den Kommandozeileneditor wie folgt geschehen:

```
mysql>     CREATE TABLE Kunde(
    ->     Kundennummer INTEGER NOT NULL AUTO_INCREMENT,
    ->     Anrede ENUM('Herr', 'Frau'),
    ->     Vorname VARCHAR(20),
    ->     Nachname VARCHAR(20),
    ->     Strasse VARCHAR(25),
    ->     PLZ INTEGER,
    ->     Ort VARCHAR(20),
    ->     Telefon VARCHAR(13),
    ->     Geburtsdatum DATE,
    ->     PRIMARY KEY (Kundennummer));
```

Diese Methode ist aber unpraktisch: Sollten die Anweisungen einen Fehler enthalten und Sie merken dies erst, nachdem Sie die [Enter]-Taste gedrückt haben, müssen Sie die Anweisung noch einmal schreiben. Wir empfehlen Ihnen daher, für die Erstellung von Tabellen einen gängigen Texteditor (z. B. Editor™ oder Open-Source-Texteditor Notepad++[1]) zu verwenden. Auf diese Weise können Sie die entsprechenden Textdateien jederzeit sichern und nachträglich wieder ändern. Die erfassten Textdateien können Sie beispielsweise im neu zu erstellenden Verzeichnis **C:\MySQL** abspeichern.

Die Hauptarbeit bei der Erstellung einer Tabelle besteht darin, deren Spalten (Felder) zu definieren. Dies geschieht, indem Sie jeder Spalte einen Namen geben und deren Eigenschaften bestimmen. So sieht beispielsweise die Anweisung für die Erstellung der Tabelle KUNDE aus:

[1] Dieser Editor erkennt die Syntax aller gängigen Programmiersprachen.

```
CREATE TABLE Kunde
(
Kundennummer INTEGER NOT NULL AUTO_INCREMENT,
Anrede ENUM('Herr', 'Frau'),
Vorname VARCHAR(20),
Nachname VARCHAR(20),
Strasse VARCHAR(25),
PLZ INTEGER,
Ort VARCHAR(20),
Telefon VARCHAR(13),
Geburtsdatum DATE,
PRIMARY KEY (Kundennummer)
);
```

Erläuterungen zur obigen Tabelle:

- Jeder Spalte (z. B. Kundennummer) wurde ein bestimmter Spaltentyp (Datentyp) zugewiesen (z. B. INTEGER). Vergleichen Sie zu den verschiedenen Datentypen die Erläuterungen in Kapitel 6.3.1, S. 67.
- Die Spalte mit der Bezeichnung «Kundennummer» hat die Eigenschaft «NOT NULL» erhalten. Damit werden Null-Werte (keine Eingabe, leeres Feld) in dieser Spalte unterbunden.
- Mit der Eigenschaft AUTO_INCREMENT wird veranlasst, dass die Kundennummer bei jedem neuen Kunden automatisch um eins erhöht wird. Begonnen wird mit der Nummer 1 für den ersten Kunden.
- Mit PRIMARY KEY wird der Primärschlüssel der Tabelle festgelegt (hier: Kundennummer).
- Es spielt keine Rolle, wie lang die einzelnen Zeilen sind. Wichtig ist, dass das Ende der Anweisung durch ein Semikolon (;) gekennzeichnet wird.

Angenommen, Sie haben obige Tabelle unter dem Namen **Kunde.sql** abgespeichert. Um die erfasste Anweisung auszuführen, müssen Sie nun folgenden Befehl ausführen:

```
mysql> SOURCE C:\MySQL\Kunde.sql
```

Die erfolgreiche Ausführung dieses Befehls wird von MySQL wie folgt bestätigt:

```
Query OK, 0 rows affected (0.06 sec)
mysql>
```

Wenn Sie MySQL nicht wie vorgeschlagen installiert haben, werden die neuen Tabellen nicht automatisch als InnoDB-Tabellen erstellt. In diesem Fall müssen Sie die Tabellen wie folgt definieren:

```
CREATE TABLE Kunde
(
...
...
) ENGINE=INNODB;
```

Hinweis

▷ Wenn Sie nicht sicher sind, ob eine Tabelle vom Typ InnoDB ist, geben Sie folgenden Befehl ein: **SHOW CREATE TABLE Kunde;**

Der Typ wird angezeigt, egal, ob Sie diesen bei der Erstellung selber bestimmt haben oder ob MySQL den Typ automatisch und stillschweigend bestimmt hat.

6.3.1 Datentypen zuweisen

Bei der Erzeugung der Kundentabelle haben Sie gesehen, dass jeder Spalte ein bestimmter Datentyp zugewiesen wird (z. B. INTEGER). Damit wird von vorneherein festgelegt, welche Sorte von Daten in dieser Spalte eingegeben werden darf. In der folgenden Tabelle sind die **wichtigsten Datentypen** aufgelistet, die MySQL unterstützt:

Datentyp	Erläuterungen
Numerisch	
DECIMAL oder NUMERIC	Speichert Zahlen unter Beibehaltung der exakten Genauigkeit (z. B. Finanzbuchhaltung). So erlaubt z. B. «Betrag DECIMAL(65,30)» als Betrag, eine Zahl mit insgesamt 65 Stellen, wovon 30 Stellen nach dem Komma, zu speichern, und zwar mit positivem oder negativem Vorzeichen.
INTEGER oder INT	Speichert Zahlen von –2 147 483 648 bis 2 147 483 647. Es sind keine Bruchzahlen erlaubt.
Datum und Zeit	
DATE	Speichert Daten von 1000-01-01 bis 9999-12-31. Bei MySQL wird das Datum immer im Format Jahr–Monat–Tag gespeichert und bearbeitet.
DATETIME	Speichert Daten und Zeiten von 1000-01-01 00:00:00 bis 9999-12-31 23:59:59.
TIME	Speichert Zeiten von –838:59:59 bis 838:59:59.
YEAR	Speichert eine Zahl zwischen 1 901 und 2 155.
TIMESTAMP	Speichert den Zeitstempel. Der Bereich liegt zwischen '1970-01-01 00:00:00' und einem Zeitpunkt irgendwann im Jahr 2037. Wird oft verwendet, um Zeitstempel der Datenbankoperation (z. B. Einfügen) festzuhalten.
Text	
CHAR(Anzahl Zeichen)	Speichert alphanumerische Zeichen. Die maximale Anzahl Zeichen beträgt 255 Zeichen. Die zur Zeit der Tabellenerstellung definierte Länge ist fix.
ENUM('Herr', 'Frau')	Speichert alphanumerische Zeichen. Der Benutzer kann nur Begriffe eingeben, die im ENUM aufgelistet sind.
VARCHAR(Anzahl Zeichen)	Speichert alphanumerische Zeichen. Die maximale Anzahl Zeichen beträgt 65 535 Zeichen. Die Länge ist variabel und ist vom Inhalt abhängig.

Der Anweisung für die Erstellung der Tabelle KUNDE (vgl. Kap. 6.3, S. 65) können Sie Folgendes entnehmen:

- Es wird nicht möglich sein, einen Nachnamen und einen Vornamen einzugeben, der länger als 20 Zeichen lang ist.
- Beim Geburtsdatum kann der Benutzer nicht versehentlich 1988-13-30 eintippen. Diese Eingabe würde als 0000-00-00 gespeichert.

6.3.2 Tabellen mit Fremdschlüssel erstellen

In der Tabelle AUSLEIHE ist u. a. die Kundennummer ein Fremdschlüssel. Die Definition eines Fremdschlüssels wird mit FOREIGN KEY gefolgt von dem Namen der betreffenden Spalte eingeleitet. Es folgt der Verweis (REFERENCES) auf die entsprechende Spalte in der anderen Tabelle:

```
CREATE TABLE Ausleihe
(
Ausleihnr INTEGER NOT NULL AUTO_INCREMENT,
Kundennummer INTEGER NOT NULL,
Videonummer INTEGER NOT NULL,
Ausleihe DATE,
Rueckgabe DATE,
PRIMARY KEY (Ausleihnr),
FOREIGN KEY (Kundennummer) REFERENCES Kunde(Kundennummer)
);
```

Hinweis

▷ Beachten Sie, dass der Datentyp der Spalte (Kundennummer) in beiden Tabellen gleich sein muss.

Wenn Sie nun eine neue Zeile in die Tabelle AUSLEIHE einfügen, müssen Sie eine Kundennummer eingeben, die in der Tabelle KUNDE schon existiert. Ansonsten wird folgende Fehlermeldung eingeblendet:

```
ERROR 1216 (23000): Cannot add or update a child row: a foreign key constraint
fails
mysql>
```

6.3.3 Tabellen einer Datenbank auflisten

Die Tabellen einer Datenbank lassen sich mit folgender Anweisung auflisten:

```
mysql> SHOW TABLES;
```

Nachdem Sie alle Tabellen der Beispiel-Datenbank «Videoverwaltung» angelegt haben, sollte in unserem Fall folgende Liste erscheinen:

```
+---------------------------+
| Tables_in_Videoverwaltung |
+---------------------------+
| ausleihe                  |
| film                      |
| kunde                     |
+---------------------------+
3 rows in set (0.06 sec)
```

6.3.4 Tabellenstruktur anzeigen

Um sich die Definition einer Tabelle anzeigen zu lassen, müssen Sie eine der folgenden Anweisungen verwenden:

```
mysql> SHOW COLUMNS FROM Tabellenname;
```

oder

```
mysql> SHOW FIELDS FROM Tabellenname;
```

Im Rahmen unserer Beispiel-Datenbank «Videoverwaltung» erscheint für die Tabelle KUNDE beispielsweise folgende Tabellenstruktur:

```
+--------------+---------------------+------+-----+---------+----------------
+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------
+
| Kundennummer | int(11)             | NO   | PRI | NULL    | auto_increment |
| Anrede       | enum('Herr','Frau') | YES  |     | NULL    |                |
| Vorname      | varchar(20)         | YES  |     | NULL    |                |
| Nachname     | varchar(20)         | YES  |     | NULL    |                |
| Strasse      | varchar(25)         | YES  |     | NULL    |                |
| PLZ          | int(11)             | YES  |     | NULL    |                |
| Ort          | varchar(20)         | YES  |     | NULL    |                |
| Telefon      | varchar(13)         | YES  |     | NULL    |                |
| Geburtsdatum | date                | YES  |     | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------
+
9 rows in set (0.17 sec)
```

Wie Sie der obigen Tabellenstruktur entnehmen können, werden nicht nur die Bezeichnungen der einzelnen Spalten der Kundentabelle angezeigt, sondern auch der jeweils zulässige Datentyp und die Angabe, ob Null-Werte erlaubt sind, etc.

6.3.5 Tabellenstruktur bearbeiten

Die Struktur einer Tabelle lässt sich nachträglich verändern. Wie Sie dabei vorgehen müssen, wird im Folgenden beschrieben.

Spalte hinzufügen

Angenommen, Sie stellen nach der Erstellung der Tabelle fest, dass manche Kunden einen akademischen Titel wie z. B. Dr. oder Dipl. Ing. ETH. führen. Wenn Sie einen solchen Titel erfassen und weiter verwenden möchten (z. B. für die Anschrift bei Mailings oder für Auswertungen), können Sie mittels Befehl ALTER TABLE … ADD COLUMN eine zusätzliche Spalte «Titel» in die bestehende Tabelle KUNDE einfügen:

```
mysql> ALTER TABLE Kunde ADD COLUMN Titel VARCHAR(10);
```

Spalte löschen

Wenn Sie eine Spalte entfernen möchten, die Sie nicht (mehr) brauchen, können Sie den Befehl ALTER TABLE … DROP COLUMN verwenden. Am Beispiel der zuvor eingefügten Spalte «Titel» würde die entsprechende Anweisung wie folgt lauten:

```
mysql> ALTER TABLE Kunde DROP COLUMN Titel;
```

Hinweis

▷ Beim Löschen einer Spalte besteht die Gefahr, dass Daten mitgelöscht werden, die im Rahmen der Datenbankbenutzung weiterhin gebraucht werden. In solchen Fällen können Inkonsistenzen auftreten. Vergleichen Sie dazu auch das Kapitel 5.3, S. 53.

6.3.6 Tabellen löschen

Das Löschen einer Tabelle geht sehr einfach, birgt allerdings grosse Gefahren, da die darin enthaltenen Daten beim Löschvorgang vollständig und unwiderruflich verloren gehen.

Wenn Sie sicher sind, dass alle Daten einer bestimmten Tabelle nicht mehr benötigt werden, können Sie diese mit dem Befehl DROP wie folgt aus der Datenbank entfernen (Beispiel):

```
mysql> DROP TABLE Kunde;
```

Hinweis

▷ Beim Löschen einer Tabelle besteht die Gefahr, dass Daten mitgelöscht werden, die im Rahmen der Datenbankbenutzung weiterhin gebraucht werden. In solchen Fällen können Inkonsistenzen auftreten. Vergleichen Sie dazu auch das Kapitel 5.3, S. 53.

6.4 Daten erfassen

Die Erfassung von Daten erfolgt über die Anweisung INSERT INTO … VALUES. Im folgenden Beispiel werden zwei neue Kunden in die Tabelle KUNDE eingefügt:

```
INSERT INTO Kunde(Kundennummer,Anrede,Vorname,Nachname,Strasse,PLZ,Ort,Telefon,Geburtsdatum)
VALUES(NULL,'Herr','Roger', 'Meili', 'Leuengasse 61',8302,'Kloten','01 814 17 07','1961-01-27');
INSERT INTO Kunde(Kundennummer,Anrede,Vorname,Nachname,Strasse,PLZ,Ort,Telefon,Geburtsdatum)
VALUES(NULL,'Frau','Monika','Hafner','Rosenrain 25',8303,'Bassersdorf','01 837 27 63','1967-05-
04');
```

Mit NULL wird signalisiert, dass wir keinen Wert für die Kundennummer eingeben. MySQL wird infolgedessen die Kundennummer selber ermitteln und abspeichern. Vergleichen Sie dazu die Erläuterungen zur Eigenschaft AUTO_INCREMENT im Rahmen der CREATE-Anweisung (vgl. Kap. 6.3, S. 65).

Wenn Sie vorhaben, Werte für alle Spalten einzugeben (was in der Praxis sehr oft der Fall ist), können Sie die Aufzählung der Spaltennamen auslassen und nur die Daten (VALUES) eingeben. Achten Sie in diesem Fall darauf, dass Sie die Daten in der gleichen Reihenfolge einfügen wie die Spalten in der Tabelle. Dazu folgendes Beispiel:

```
INSERT INTO Kunde VALUES(NULL,'Herr','Roger','Meili','Leuengasse
61',8302,'Kloten','01 814 17 07','1961-01-27');
```

6.5 Datenbank löschen

Eine komplette Datenbank lässt sich schnell und bequem mittels Befehl DROP DATABASE löschen. Wenn Sie z. B. die angelegte Datenbank «Videoverwaltung» wieder löschen möchten, lautet die entsprechende Anweisung wie folgt:

```
mysql> DROP DATABASE Videoverwaltung;
```

Beachten Sie, dass mit dieser Anweisung das entsprechende Verzeichnis (hier: Videoverwaltung) und die darin enthaltenen Dateien (hier: Tabellen KUNDE, FILM, AUSLEIHE) ohne Sicherheitsabfrage gelöscht werden. Dabei werden unwiderruflich sämtliche Daten entfernt, die in der betreffenden Datenbank erfasst wurden.

Die Umsetzung des bereinigten Datenmodells setzt ein geeignetes **Datenbankprogramm** voraus. In unserem Beispiel verwenden wir dafür das Datenbankmanagementsystem **MySQL,** das kostenlos vom Internet heruntergeladen und einfach installiert werden kann.

Die **Realisierung einer relationalen Datenbank mit MySQL** umfasst folgende Schritte:

1. MySQL installieren und starten
2. Datenbank erzeugen und aktivieren
3. Tabellen der Datenbank erstellen
4. Daten der Datenbank erfassen