
O'REILLY®

3. Auflage

Einführung in SQL

Daten erzeugen, bearbeiten
und abfragen



Alan Beaulieu
Übersetzung
von Thomas Demmig

Inhalt

Cover

Titel

Impressum

Inhalt

Einleitung

1 Der Hintergrund

Einführung in Datenbanken

 Nicht-relationale Datenbanksysteme

 Das relationale Modell

 Ein wenig Fachjargon

Was ist SQL?

 SQL-Anweisungen

 SQL: eine nicht-prozedurale Sprache

 SQL-Beispiele

Was ist MySQL?

SQL unplugged

Weiteres Vorgehen

2 Datenbanken erstellen und mit Daten füllen

Eine MySQL-Datenbank anlegen

Das mysql-Kommandozeilentool

MySQL-Datentypen

 Zeichendaten

 Numerische Daten

 Temporale Daten

Tabellen anlegen

Schritt 1: Entwurf

Schritt 2: Verfeinerung

Schritt 3: Die SQL-Schemaanweisungen

Tabellen füllen und ändern

Daten einfügen

Daten ändern

Daten löschen

Wenn aus guten Anweisungen schlechte werden

Nicht-eindeutiger Primärschlüssel

Nicht-existenter Fremdschlüssel

Verstöße gegen Spaltenwerte

Ungültige Datumskonvertierung

Die Sakila-Datenbank

3 Datenbankabfragen

Die Mechanik von Abfragen

Abfrageklauseln

Die select-Klausel

Spaltenaliase

Duplikate entfernen

Die from-Klausel

Tabellen

Tabellenverknüpfungen

Tabellenaliase definieren

Die where-Klausel

Die Klauseln group by und having

Die order by-Klausel

Auf- und absteigende Sortierung

Sortieren nach numerischen Platzhaltern

Testen Sie Ihr Wissen

Übung 3-1

Übung 3-2

Übung 3-3

Übung 3-4

4 Filtern

Bedingungsauswertung

Verwendung von Klammern

Verwendung des Operators not

Aufbau einer Bedingung

Bedingungstypen

Gleichheitsbedingungen

Wertebereichsbedingungen

Mitgliedschaftsbedingungen

Bedingungen abgleichen

NULL: ein böses Wort

Testen Sie Ihr Wissen

Übung 4-1

Übung 4-2

Übung 4-3

Übung 4-4

5 Mehrere Tabellen abfragen

Was ist ein Join?

Kartesisches Produkt

Inner Joins

Die Join-Syntax von ANSI

Joins mit drei oder mehr Tabellen

Unterabfragen als Tabellen

Zweimal dieselbe Tabelle verwenden

Self Joins

Testen Sie Ihr Wissen

Übung 5-1

Übung 5-2

Übung 5-3

6 Umgang mit Mengen

Grundlagen der Mengenlehre

Mengenlehre in der Praxis

Mengenoperatoren

Der union-Operator

Der intersect-Operator

Der except-Operator

Regeln für Mengenoperationen

Ergebnisse zusammengesetzter Abfragen sortieren

Präzedenz von Mengenoperationen

Testen Sie Ihr Wissen

Übung 6-1

Übung 6-2

Übung 6-3

7 Daten erzeugen, bearbeiten und konvertieren

Der Umgang mit String-Daten

String-Daten erzeugen

String-Bearbeitung

Der Umgang mit numerischen Daten

Arithmetische Funktionen

Die Genauigkeit von Zahlen steuern

Vorzeichenbehaftete Daten

Der Umgang mit temporalen Daten

Zeitzone

Temporale Daten erzeugen

Temporale Daten bearbeiten

Konvertierungsfunktionen

Testen Sie Ihr Wissen

Übung 7-1

Übung 7-2

Übung 7-3

8 Gruppieren und Aggregieren von Daten

Gruppieren von Daten

Aggregatfunktionen

Implizite und explizite Gruppen

Unterschiedliche Werte zählen

Ausdrücke

Umgang mit null-Werten

Gruppen erzeugen

Gruppieren auf einer einzelnen Spalte

Gruppieren auf mehreren Spalten

Gruppieren mit Ausdrücken

Rollups erzeugen

Gruppen-Filterbedingungen

Testen Sie Ihr Wissen

Übung 8-1

Übung 8-2

Übung 8-3

9 Unterabfragen

Was ist eine Unterabfrage?

Typen von Unterabfragen

Nicht-korrelierte Unterabfragen

Unterabfragen, die eine Spalte und mehrere Zeilen liefern

Unterabfragen, die mehrere Spalten liefern

Korrelierte Unterabfragen

Der exists-Operator

Datenbearbeitung mit korrelierten Unterabfragen

Einsatz von Unterabfragen

Unterabfragen als Datenquellen

Unterabfragen zum Erzeugen von Ausdrücken

Zusammenfassung zu Unterabfragen

Testen Sie Ihr Wissen

Übung 9-1

Übung 9-2

Übung 9-3

10 Weitere Joins

Outer Joins

Left und Right Outer Joins

Outer Joins mit drei Tabellen

Cross Joins

Natural Joins

Testen Sie Ihr Wissen

Übung 10-1

Übung 10-2

Übung 10-3 (für Tüftler)

11 Bedingungslogik

Was ist Bedingungslogik?

Der Case-Ausdruck

Searched Case-Ausdrücke

Einfache Case-Ausdrücke

Beispiele für Case-Ausdrücke

Umwandlungen von Ergebnismengen

Prüfung auf Vorhandensein

Fehler bei einer Division durch null

Bedingte Updates

Der Umgang mit null-Werten

Testen Sie Ihr Wissen

Übung 11-1

Übung 11-2

12 Transaktionen

Mehrbenutzerdatenbanken

Sperrern

Granularität von Sperrern

Was ist eine Transaktion?

Transaktion starten

Transaktion beenden

Savepoints

Testen Sie Ihr Wissen

Übung 12-1

13 Indizes und Constraints

Indizes

Indexerstellung

Indextypen

Verwendung von Indizes

Der Nachteil von Indizes

Constraints

Constraints anlegen

Testen Sie Ihr Wissen

Übung 13-1

Übung 13-2

14 Views

Was sind Views?

Warum Views verwenden?

Datensicherheit

Datenaggregation

Komplexität verbergen

Partitionierte Daten verknüpfen

Aktualisierbare Views

Einfache Views aktualisieren

Komplexe Views aktualisieren

Testen Sie Ihr Wissen

Übung 14-1

Übung 14-2

15 Metadaten

Daten über Daten

information_schema

Mit Metadaten arbeiten

Skripte zur Schemagenerierung

Deployment-Überprüfung

Dynamisch SQL erzeugen

Testen Sie Ihr Wissen

Übung 15-1

Übung 15-2

16 Analytische Funktionen

Konzepte analytischer Funktionen

Datenfenster

Lokalisiertes Sortieren

Rangfolgen

Rangfolgefunktionen

Mehrere Rangfolgen erstellen

Reporting-Funktionen

Fenstergrenzen

lag und lead

Verketteten von Spaltenwerten

Testen Sie Ihr Wissen

Übung 16-1

Übung 16-2

Übung 16-3

17 Mit großen Datenbanken arbeiten

Partitionieren

Partitionierungskonzepte

Tabellen partitionieren

Indizes partitionieren

Partitionierungsmethoden

Vorteile des Partitionierens

Clustering

Sharding

Big Data

Hadoop

NoSQL und Dokumentendatenbanken

Cloud Computing

Zusammenfassung

18 SQL und Big Data

Einführung in Apache Drill

Dateien mit Drill abfragen

MySQL mit Drill abfragen

MongoDB mit Drill abfragen

Drill mit mehreren Datenquellen verwenden

Die Zukunft von SQL

A ER-Diagramm der Musterdatenbank

B Lösungen zu den Übungen

Index

Über den Autor

Über den Übersetzer

Kolophon

In manchen Fällen wird mit allen Zeilen einer Tabelle gearbeitet, nämlich:

- Wenn alle Daten aus einer Tabelle geleert werden sollen, um neue Data-Warehouse-Feeds zu präsentieren.
- Wenn alle Zeilen einer Tabelle modifiziert werden sollen, nachdem eine neue Spalte hinzugefügt wurde.
- Wenn alle Zeilen aus einer Message-Queue-Tabelle abgeholt werden sollen.

In Fällen wie diesen benötigt eine SQL-Anweisung keine `where`-Klausel, da keine Zeilen von der Betrachtung ausgeschlossen werden müssen. Meist ist jedoch nur ein Teil der Tabellenzeilen von Interesse. Daher haben alle SQL-Datenanweisungen (außer `insert`) eine optionale `where`-Klausel mit einer oder mehreren Filterbedingungen, die die Zahl der von der SQL-Anweisung betrachteten Zeilen eingrenzen. Zusätzlich enthält die `select`-Anweisung eine `having`-Klausel, um Filterbedingungen zu formulieren, die für gruppierte Daten gelten sollen. Dieses Kapitel untersucht die verschiedenen Filterbedingungen, die in den `where`-Klauseln der `select`-, `update`- und `delete`-Anweisungen stehen können; den Einsatz von Filterbedingungen in der `having`-Klausel einer `select`-Anweisung werden wir uns in Kapitel 8 ansehen.

Bedingungsauswertung

Eine `where`-Klausel kann eine oder mehrere durch die Operatoren `and` und `or` getrennte Bedingungen enthalten. Wenn mehrere Bedingungen lediglich durch den `and`-Operator getrennt werden, müssen alle diese Bedingungen `true` sein, damit die betreffende Zeile in die Ergebnismenge aufgenommen wird. Betrachten Sie die folgende `where`-Klausel:

```
WHERE first_name = 'STEVEN' AND create_date > '2006-01-01'
```

Diese beiden Bedingungen führen dazu, dass nur Zeilen berücksichtigt werden, bei denen der Vorname Steven ist und das Erstellungsdatum nach dem 1. Januar 2006 liegt. Dieses Beispiel verwendet zwar nur zwei Bedingungen, aber im Prinzip können Sie beliebig viele Bedingungen in die `where`-Klausel schreiben. Wenn alle durch den Operator `and` getrennt sind, müssen auch alle `true` sein, damit die Zeile in die Ergebnismenge kommt.

Sind sämtliche Bedingungen der `where`-Klausel durch den Operator `or` getrennt, muss hingegen nur eine von ihnen `true` sein, damit die betreffende Zeile in der Ergebnismenge Aufnahme findet. Betrachten Sie die folgenden beiden Bedingungen:

```
WHERE first_name = 'STEVEN' OR create_date > '2006-01-01'
```

Jetzt kann eine Zeile auf mehreren Wegen in die Ergebnismenge gelangen:

- Der Vorname ist Steven, und das Erstellungsdatum liegt nach dem 1. Januar 2006.
- Der Vorname ist Steven, und das Erstellungsdatum liegt vor dem 1. Januar 2006, oder es ist dieses Datum.
- Der Vorname ist nicht Steven, aber das Erstellungsdatum liegt nach dem 1. Januar 2006.

Tabelle 4-1 zeigt die möglichen Resultate einer `where`-Klausel mit zwei durch `or` getrennten Bedingungen.

Tabelle 4-1: Auswertung von zwei Bedingungen mit dem `or`-Operator

Zwischenergebnis	Endergebnis
WHERE true OR true	true
WHERE true OR false	true
WHERE false OR true	true
WHERE false OR false	false

Im obigen Beispiel ist eine Zeile nur in einem Fall aus der Ergebnismenge ausgeschlossen: wenn der Vorname nicht Steven ist und das Erstellungsdatum vor dem 1. Januar 2006 liegt oder dieses Datum ist.

Verwendung von Klammern

Wenn Ihre `where`-Klausel mindestens drei Bedingungen und sowohl `and` als auch `or` enthält, sollten Sie durch Klammern klarstellen, was Sie bezwecken, damit sowohl die Datenbank als auch nachfolgende Generationen Ihren Code noch lesen können. Die folgende `where`-Klausel erweitert das obige Beispiel um die Prüfung, ob der Vorname Steven oder der Nachname Young ist und das Erstellungsdatum nach dem 1. Januar 2006 liegt:

```
WHERE (first_name = 'STEVEN' OR last_name = 'YOUNG')
```

```
    AND create_date > '2006-01-01'
```

Jetzt gibt es drei Bedingungen: Damit eine Zeile in die Ergebnismenge gelangt, müssen zuerst die erste oder zweite Bedingung (oder beide) `true` und dann die dritte ebenfalls `true` sein. Tabelle 4-2 zeigt die möglichen Resultate dieser `where`-Klausel.

Tabelle 4-2: Auswertung von drei Bedingungen mit `and` und `or`

Zwischenergebnis	Endergebnis
<code>WHERE (true OR true) AND true</code>	<code>true</code>
<code>WHERE (true OR false) AND true</code>	<code>true</code>
<code>WHERE (false OR true) AND true</code>	<code>true</code>
<code>WHERE (false OR false) AND true</code>	<code>false</code>
<code>WHERE (true OR true) AND false</code>	<code>false</code>
<code>WHERE (true OR false) AND false</code>	<code>false</code>
<code>WHERE (false OR true) AND false</code>	<code>false</code>
<code>WHERE (false OR false) AND false</code>	<code>false</code>

Man sieht: Je mehr Bedingungen eine `where`-Klausel aufweist, umso mehr Kombinationen muss der Server auswerten. In diesem Fall ergeben nur drei der acht möglichen Kombinationen das Endergebnis `true`.

Verwendung des Operators not

Hoffentlich war das obige Beispiel mit den drei Bedingungen gut zu verstehen. Betrachten Sie gleichwohl folgende Abänderung:

```
WHERE NOT (first_name = 'STEVEN' OR last_name = 'YOUNG')  
  
    AND create_date > '2006-01-01'
```

Haben Sie sie erkannt? Ich habe vor dem ersten Satz an Bedingungen einen not-Operator eingefügt. Nun suche ich nicht mehr nach Personen, deren Vorname Steven oder deren Nachname Young lautet und deren Datensatz nach dem 1. Januar 2006 erstellt wurde, sondern ich erhalte nur die Zeilen, bei denen der Vorname nicht Steven oder der Nachname Young ist und deren Datensatz nach dem 1. Januar 2006 angelegt wurde. Tabelle 4-3 zeigt die möglichen Resultate für dieses Beispiel.

Tabelle 4-3: Auswertung von drei Bedingungen mit and, or und not

Zwischenergebnis	Endergebnis
WHERE NOT (true OR true) AND true	false
WHERE NOT (true OR false) AND true	false
WHERE NOT (false OR true) AND true	false
WHERE NOT (false OR false) AND true	true
WHERE NOT (true OR true) AND false	false
WHERE NOT (true OR false) AND false	false
WHERE NOT (false OR true) AND false	false
WHERE NOT (false OR false) AND false	false

Der not-Operator ist für einen Datenbankserver kein Problem, aber ein Mensch kann eine where-Klausel mit diesem Operator nicht mehr so leicht durchschauen. Daher trifft man ihn nur selten an. In diesem Fall könnte man die where-Klausel auch ohne not-Operator schreiben:

```
WHERE first_name <> 'STEVEN' AND last_name <> 'YOUNG'
```

```
AND create_date > '2006-01-01'
```

Dem Server ist es sicherlich egal, aber Sie als Leser können diese Version der *where*-Klausel bestimmt besser verstehen.

Aufbau einer Bedingung

Da Sie nun gesehen haben, wie mehrere Bedingungen vom Server ausgewertet werden, gehen wir wieder einen Schritt zurück und betrachten, was eigentlich eine einzelne Bedingung ausmacht. Eine Bedingung besteht aus einem oder mehreren *Ausdrücken*, die durch einen oder mehrere *Operatoren* verbunden sind. Ein Ausdruck kann Folgendes sein:

- eine Zahl
- eine Spalte einer Tabelle oder View
- ein String-Literal wie beispielsweise 'Maple Street'
- eine eingebaute Funktion wie beispielsweise `concat('Learning', ' ', 'SQL')`
- eine Unterabfrage
- eine Liste von Ausdrücken wie beispielsweise `('Boston', 'New York', 'Chicago')`

Folgende Operatoren sind in Bedingungen zulässig:

- Vergleichsoperatoren wie beispielsweise `=`, `!=`, `<`, `>`, `<>`, `LIKE`, `IN` und `BETWEEN`
- arithmetische Operatoren wie beispielsweise `+`, `-`, `*` und `/`

Der folgende Abschnitt zeigt, wie diese Ausdrücke und Operatoren kombiniert werden können, um verschiedene Arten von Bedingungen zu konstruieren.

Bedingungstypen

Es gibt viele verschiedene Wege, unerwünschte Daten herauszufiltern. Sie können bestimmte Werte, Wertemengen oder Wertebereiche ein- oder ausschließen oder mit diversen Mustererkennungstechniken Teilübereinstimmungen in String-Daten ausfindig machen. Im Folgenden werden diese Bedingungstypen genauer erklärt.

Gleichheitsbedingungen

Ein großer Prozentsatz der Filterbedingungen, die Sie schreiben oder über die Sie stolpern werden, hat die Form '*spalte = ausdruck*':

```
title = 'RIVER OUTLAW'
```

```
fed_id = '111-11-1111'
```

```
amount = 375.25
```

```
film_id = (SELECT film_id FROM film WHERE title = 'RIVER  
OUTLAW')
```

Solche Bedingungen bezeichnet man als *Gleichheitsbedingungen*, da sie einen Ausdruck mit einem anderen gleichsetzen. Die ersten drei Beispiele setzen eine Spalte mit einem Literal gleich (zweimal ein String und einmal eine Zahl), und das vierte Beispiel setzt eine Spalte mit dem Rückgabewert einer Unterabfrage gleich. Die folgende Abfrage verwendet zwei Gleichheitsbedingungen: eine in der on-Klausel (eine Join-Bedingung) und die andere in der where-Klausel (eine Filterbedingung):

```
mysql> SELECT c.email
```

```
-> FROM customer c
```

```
-> INNER JOIN rental r
```

```
-> ON c.customer_id = r.customer_id
```

```
-> WHERE date(r.rental_date) = '2005-06-14';
```

```
+-----+
```

```
| email
```

```
|
```

+-----+

| CATHERINE.CAMPBELL@sakilacustomer.org |
| JOYCE.EDWARDS@sakilacustomer.org |
| AMBER.DIXON@sakilacustomer.org |
| JEANETTE.GREENE@sakilacustomer.org |
| MINNIE.ROMERO@sakilacustomer.org |
| GWENDOLYN.MAY@sakilacustomer.org |
| SONIA.GREGORY@sakilacustomer.org |
| MIRIAM.MCKINNEY@sakilacustomer.org |
| CHARLES.KOWALSKI@sakilacustomer.org |
| DANIEL.CABRAL@sakilacustomer.org |
| MATTHEW.MAHAN@sakilacustomer.org |
| JEFFERY.PINSON@sakilacustomer.org |
| HERMAN.DEVORE@sakilacustomer.org |
| ELMER.NOE@sakilacustomer.org |
| TERRANCE.ROUSH@sakilacustomer.org |
| TERRENCE.GUNDERSON@sakilacustomer.org |

```
+-----+
```

```
16 rows in set (0.03 sec)
```

Diese Abfrage zeigt die E-Mail-Adressen aller Kunden, die am 14. Juni 2005 einen Film ausgeliehen haben.

Ungleichheitsbedingungen

Ebenfalls ziemlich gebräuchlich ist die *Ungleichheitsbedingung*, die fordert, dass zwei Ausdrücke *nicht* gleich sind. Hier habe ich die Filterbedingung aus der *where*-Klausel der obigen Abfrage in eine Ungleichheitsbedingung umgewandelt:

```
mysql> SELECT c.email
      -> FROM customer c
      ->   INNER JOIN rental r
      ->   ON c.customer_id = r.customer_id
      -> WHERE date(r.rental_date) <> '2005-06-14';
```

```
+-----+
```

```
| email |
```

```
+-----+
```

```
| MARY.SMITH@sakilacustomer.org |
```

```
| MARY.SMITH@sakilacustomer.org |
```

```
| MARY.SMITH@sakilacustomer.org |
```

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

| MARY.SMITH@sakilacustomer.org |

...

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

| AUSTIN.CINTRON@sakilacustomer.org |

+-----+

16028 rows in set (0.03 sec)

Diese Abfrage zeigt alle E-Mail-Adressen für Filmausleihvorgänge, die an einem Datum stattfanden, das nicht der 14. Juni 2005 war. Bei der Konstruktion von Ungleichheitsbedingungen können Sie entweder `!=` oder `<>` als Operator verwenden.

Daten mit Gleichheitsbedingungen modifizieren

Gleichheits- und Ungleichheitsbedingungen werden gern auch zur Modifikation von Daten eingesetzt. Angenommen, die Videothek wirft einmal jährlich alte Kontendatensätze über Bord. Ihre Aufgabe ist es, aus der `rental`-Tabelle die Zeilen zu entfernen, bei denen der Ausleihvorgang im Jahr 2004 lag. Hier sehen Sie eine Möglichkeit, dies zu tun:

```
DELETE FROM rental  
  
WHERE year(rental_date) = 2004;
```

Diese Anweisung enthält eine Gleichheitsbedingung, das folgende Beispiel nutzt dagegen zwei Ungleichheitsbedingungen, um alle Zeilen zu entfernen, bei denen das Ausleihdatum nicht in den Jahren 2005 oder 2006 lag:

```
DELETE FROM rental  
  
WHERE year(rental_date) <> 2005 AND year(rental_date) <> 2006;
```



Bei der Formulierung von `delete`- und `update`-Anweisungen werde ich versuchen, jede Anweisung so zu schreiben, dass keine Zeilen modifiziert werden. So bleiben die Daten bei der Ausführung der Anweisungen unverändert, und die Ausgaben Ihrer `select`-Anweisungen entsprechen immer den in diesem Buch gezeigten.

Da MySQL-Sessions standardmäßig im Auto-Commit-Modus ablaufen (siehe Kapitel 12), könnten Sie, falls meine Anweisungen die Beispieldaten ändern würden, diese Änderungen nicht wieder zurückrollen (ungeschehen machen). Sie selbst können natürlich mit Ihren Beispieldaten machen, was Sie wollen; meinetwegen können Sie

sie auch löschen und die Skripte erneut ausführen, aber ich für meinen Teil werde darauf achten, dass die Daten intakt bleiben.

Wertebereichsbedingungen

Sie können in einer Bedingung nicht nur prüfen, ob ein Ausdruck gleich (oder ungleich) einem anderen Ausdruck ist, sondern auch, ob ein Ausdruck in einem bestimmten Wertebereich liegt. Diese Art von Bedingung findet man häufig im Umgang mit numerischen oder temporalen Daten. Betrachten Sie folgende Abfrage:

```
mysql> SELECT customer_id, rental_date  
  
-> FROM rental  
  
-> WHERE rental_date < '2005-05-25';
```

```
+-----+-----+  
  
| customer_id | rental_date          |  
  
+-----+-----+  
  
|          130 | 2005-05-24 22:53:30 |  
  
|          459 | 2005-05-24 22:54:33 |  
  
|          408 | 2005-05-24 23:03:39 |  
  
|          333 | 2005-05-24 23:04:41 |  
  
|          222 | 2005-05-24 23:05:21 |  
  
|          549 | 2005-05-24 23:08:07 |  
  
|          269 | 2005-05-24 23:11:53 |
```

```
|          239 | 2005-05-24 23:31:46 |
```

```
+-----+-----+
```

```
8 rows in set (0.00 sec)
```

Diese Abfrage findet alle Ausleihvorgänge, die vor dem 25. Mai 2005 stattfanden. Doch eventuell möchten Sie nicht nur eine Obergrenze für das Ausleihdatum angeben, sondern auch eine Untergrenze:

```
mysql> SELECT customer_id, rental_date
```

```
-> FROM rental
```

```
-> WHERE rental_date <= '2005-06-16'
```

```
-> AND rental_date >= '2005-06-14';
```

```
+-----+-----+
```

```
| customer_id | rental_date          |
```

```
+-----+-----+
```

```
|          416 | 2005-06-14 22:53:33 |
```

```
|          516 | 2005-06-14 22:55:13 |
```

```
|          239 | 2005-06-14 23:00:34 |
```

```
|          285 | 2005-06-14 23:07:08 |
```

```
|          310 | 2005-06-14 23:09:38 |
```

```

|          592 | 2005-06-14 23:12:46 |
...
|          148 | 2005-06-15 23:20:26 |
|          237 | 2005-06-15 23:36:37 |
|          155 | 2005-06-15 23:55:27 |
|          341 | 2005-06-15 23:57:20 |
|          149 | 2005-06-15 23:58:53 |

+-----+-----+

```

364 rows in set (0.00 sec)

Diese Version der Abfrage findet alle Filme, die am 14. oder 15. Juni 2005 ausgeliehen wurden.

Der between-Operator

Wenn Sie *sowohl* eine obere *als auch* eine untere Grenze für Ihren Wertebereich haben, können Sie statt zwei separaten Bedingungen auch eine einzige Bedingung mit dem Operator `between` formulieren:

```
mysql> SELECT customer_id, rental_date
```

```
    -> FROM rental
```

```
    -> WHERE rental_date BETWEEN '2005-06-14' AND '2005-06-16';
```

```
+-----+-----+
```

```
| customer_id | rental_date |
```

```
+-----+-----+
```

```
|          416 | 2005-06-14 22:53:33 |
```

```
|          516 | 2005-06-14 22:55:13 |
```

```
|          239 | 2005-06-14 23:00:34 |
```

```
|          285 | 2005-06-14 23:07:08 |
```

```
|          310 | 2005-06-14 23:09:38 |
```

```
|          592 | 2005-06-14 23:12:46 |
```

```
...
```

```
|          148 | 2005-06-15 23:20:26 |
```

```
|          237 | 2005-06-15 23:36:37 |
```

```
|          155 | 2005-06-15 23:55:27 |
```

```
|          341 | 2005-06-15 23:57:20 |
```

```
|          149 | 2005-06-15 23:58:53 |
```

```
+-----+-----+
```

```
364 rows in set (0.00 sec)
```

Bei der Verwendung des `between`-Operators sind einige Dinge zu beachten: Geben Sie immer zuerst (hinter `between`) die Untergrenze und als Zweites (hinter `and`) die Obergrenze des Wertebereichs an. Hier sehen Sie, was passiert, wenn Sie versehentlich zuerst die Obergrenze angeben:

```
mysql> SELECT customer_id, rental_date  
  
-> FROM rental  
  
-> WHERE rental_date BETWEEN '2005-06-16' AND '2005-06-14';
```

Empty set (0.00 sec)

Es werden keine Daten zurückgegeben, da der Server in Wirklichkeit aus Ihrer einen Bedingung zwei Bedingungen (mit den Operatoren `<=` und `>=`) macht, wie in:

```
SELECT customer_id, rental_date  
  
-> FROM rental  
  
-> WHERE rental_date >= '2005-06-16'  
  
-> AND rental_date <= '2005-06-14'
```

Empty set (0.00 sec)

Da es unmöglich ein Datum geben kann, das sowohl größer als der 16. Juni 2005 als auch kleiner als der 14. Juni 2005 ist, gibt die Abfrage die leere Menge zurück. Und das bringt mich zur zweiten Falle von `between`: Denken Sie stets daran, dass Ober- und Untergrenze *inklusiv* sind, dass ihre Werte also im Wertebereich enthalten sind. In diesem Fall will ich alle Ausleihvorgänge vom 14. oder 15. Juni 2005 haben, also gebe ich `2005-06-14` als Unter- und `2005-06-16` als Obergrenze an. Da ich die Stunden-, Minuten- und Sekundenangaben des Datums werts nicht festlege, wird die Zeit auf Mitternacht gelegt, und der

effektive Bereich ist 2005-06-14 00:00:00 bis 2005-06-16 00:00:00. Damit sind alle Ausleihvorgänge vom 14. oder 15. Juni eingeschlossen.

Neben Datumsintervallen können Bedingungen auch Zahlenintervalle angeben. Diese sind leicht verständlich, wie das folgende Beispiel zeigt:

```
mysql> SELECT customer_id, payment_date, amount
```

```
    -> FROM payment
```

```
    -> WHERE amount BETWEEN 10.0 AND 11.99;
```

```
+-----+-----+-----+
| customer_id | payment_date          | amount |
+-----+-----+-----+
|           2 | 2005-07-30 13:47:43 | 10.99 |
|           3 | 2005-07-27 20:23:12 | 10.99 |
|          12 | 2005-08-01 06:50:26 | 10.99 |
|          13 | 2005-07-29 22:37:41 | 11.99 |
|          21 | 2005-06-21 01:04:35 | 10.99 |
|          29 | 2005-07-09 21:55:19 | 10.99 |
...
|         571 | 2005-06-20 08:15:27 | 10.99 |
|         572 | 2005-06-17 04:05:12 | 10.99 |
```

```

|          573 | 2005-07-31 12:14:19 | 10.99 |
|          591 | 2005-07-07 20:45:51 | 11.99 |
|          592 | 2005-07-06 22:58:31 | 11.99 |
|          595 | 2005-07-31 11:51:46 | 10.99 |

```

```
+-----+-----+-----+
```

114 rows in set (0.01 sec)

Alle Bezahlvorgänge zwischen \$10,00 und \$11,99 werden gemeldet. Denken Sie auch hier daran, den kleineren Betrag als Erstes anzugeben.

String-Bereiche

Während Datums- und Zahlenintervalle leicht zu verstehen sind, lassen sich Bedingungen, die nach String-Intervallen suchen, schon etwas schwerer veranschaulichen. Angenommen, Sie suchten nach Kunden, deren Nachname in einen bestimmten Bereich fällt. Diese Abfrage liefert alle Kunden, bei denen die Anfangsbuchstaben der Nachnamen in dem Bereich zwischen FA und FR liegen:

```
mysql> SELECT last_name, first_name
```

```
    -> FROM customer
```

```
    -> WHERE last_name BETWEEN 'FA' AND 'FR';
```

```
+-----+-----+
```

```
| last_name | first_name |
```

```
+-----+-----+
```

	FARNSWORTH		JOHN	
	FENNELL		ALEXANDER	
	FERGUSON		BERTHA	
	FERNANDEZ		MELINDA	
	FIELDS		VICKI	
	FISHER		CINDY	
	FLEMING		MYRTLE	
	FLETCHER		MAE	
	FLORES		JULIA	
	FORD		CRYSTAL	
	FORMAN		MICHEAL	
	FORSYTHE		ENRIQUE	
	FORTIER		RAUL	
	FORTNER		HOWARD	
	FOSTER		PHYLLIS	
	FOUST		JACK	
	FOWLER		JO	

```
| FOX          | HOLLY          |
```

```
+-----+-----+
```

```
18 rows in set (0.00 sec)
```

Es gibt zwar fünf Kunden, deren Nachname mit FR beginnt, aber sie sind nicht in der Ergebnismenge enthalten, da ein Name wie FRANKLIN außerhalb des Bereichs liegt. Wir könnten aber vier der fünf Kunden mit aufnehmen, indem wir die rechte Bereichsgrenze auf FRB erweitern:

```
mysql> SELECT last_name, first_name
```

```
    -> FROM customer
```

```
    -> WHERE last_name BETWEEN 'FA' AND 'FRB';
```

```
+-----+-----+
```

```
| last_name  | first_name  |
```

```
+-----+-----+
```

```
| FARNSWORTH | JOHN        |
```

```
| FENNELL    | ALEXANDER  |
```

```
| FERGUSON   | BERTHA     |
```

```
| FERNANDEZ  | MELINDA    |
```

```
| FIELDS     | VICKI      |
```

```
| FISHER     | CINDY      |
```

FLEMING	MYRTLE	
FLETCHER	MAE	
FLORES	JULIA	
FORD	CRYSTAL	
FORMAN	MICHEAL	
FORSYTHE	ENRIQUE	
FORTIER	RAUL	
FORTNER	HOWARD	
FOSTER	PHYLLIS	
FOUST	JACK	
FOWLER	JO	
FOX	HOLLY	
FRALEY	JUAN	
FRANCISCO	JOEL	
FRANKLIN	BETH	
FRAZIER	GLENDA	

+-----+-----+

22 rows in set (0.00 sec)

Um mit String-Bereichen arbeiten zu können, müssen Sie die Reihenfolge der Zeichen Ihres Zeichensatzes kennen (die Sortierreihenfolge der Zeichen eines Zeichensatzes nennt man auch *Kollation*).

Mitgliedschaftsbedingungen

Es gibt Fälle, in denen man einen Ausdruck nicht auf einen Einzelwert oder Wertebereich eingrenzen möchte, sondern auf eine endliche Wertemenge. Wenn Sie beispielsweise alle Filme finden möchten, deren Klassifizierung nach Altersempfehlungen entweder 'G' oder 'PG' ist, tun Sie Folgendes¹:

```
mysql> SELECT title, rating
```

```
    -> FROM film
```

```
    -> WHERE rating = 'G' OR rating = 'PG';
```

```
+-----+-----+
| title                | rating |
+-----+-----+
| ACADEMY DINOSAUR    | PG     |
| ACE GOLDFINGER      | G      |
| AFFAIR PREJUDICE   | G      |
| AFRICAN EGG        | G      |
| AGENT TRUMAN       | PG     |
```

ALAMO VIDEOTAPE	G	
ALASKA PHANTOM	PG	
ALI FOREVER	PG	
AMADEUS HOLY	PG	
...		
WEDDING APOLLO	PG	
WEREWOLF LOLA	G	
WEST LION	G	
WIZARD COLDBLOODED	PG	
WON DARES	PG	
WONDERLAND CHRISTMAS	PG	
WORDS HUNTER	PG	
WORST BANGER	PG	
YOUNG LANGUAGE	G	

+-----+----+

372 rows in set (0.00 sec)

Diese where-Klausel (zwei durch or verknüpfte Bedingungen) war zwar nicht schwer zu konstruieren, aber stellen Sie sich eine Menge von 10 oder 20

Ausdrücken vor! Für solche Situationen können Sie stattdessen den Operator `in` nutzen:

```
SELECT title, rating

FROM film

WHERE rating IN ('G','PG');
```

Mit dem `in`-Operator können Sie eine Menge von beliebig vielen Ausdrücken in einer einzigen Bedingung unterbringen.

Unterabfragen

Sie können nicht nur selbst eine Menge von Ausdrücken wie beispielsweise ('G', 'PG') schreiben, sondern auch eine solche Menge direkt von einer Unterabfrage erzeugen lassen. Vielleicht gehen Sie ja davon aus, dass jeder Film, der den String 'PET' enthält, ein Kinderfilm ist. Dann könnten Sie eine Unterabfrage auf der `film`-Tabelle ausführen, um alle Bewertungen zu diesen Filmen abzurufen und damit dann alle Filme mit diesen Bewertungen auszugeben:

```
mysql> SELECT title, rating

-> FROM film

-> WHERE rating IN (SELECT rating FROM film WHERE title
LIKE '%PET%');
```

```
+-----+----+
| title                | rating |
+-----+----+
| ACADEMY DINOSAUR    | PG     |
```

ACE GOLDFINGER	G	
AFFAIR PREJUDICE	G	
AFRICAN EGG	G	
AGENT TRUMAN	PG	
ALAMO VIDEOTAPE	G	
ALASKA PHANTOM	PG	
ALI FOREVER	PG	
AMADEUS HOLY	PG	
...		
WEDDING APOLLO	PG	
WEREWOLF LOLA	G	
WEST LION	G	
WIZARD COLDBLOODED	PG	
WON DARES	PG	
WONDERLAND CHRISTMAS	PG	
WORDS HUNTER	PG	
WORST BANGER	PG	

```
| YOUNG LANGUAGE          | G          |
```

```
+-----+-----+
```

```
372 rows in set (0.00 sec)
```

Die Unterabfrage gibt eine Menge mit 'G' und 'PG' zurück, und die Hauptabfrage schaut nach, ob die Spalte `rating` in dieser Rückgabemenge zu finden ist.

Verwendung von `not in`

Manchmal möchte man wissen, ob ein bestimmter Ausdruck in einer Menge von Ausdrücken vorhanden ist, manchmal aber auch, ob er darin *nicht* vorhanden ist. In solchen Situationen leistet der Operator `not in` gute Dienste:

```
SELECT title, rating
```

```
FROM film
```

```
WHERE rating NOT IN ('PG-13','R', 'NC-17');
```

Diese Abfrage findet alle Konten, die *nicht* mit 'PG-13', 'R' oder 'NC-17' bewertet wurden, und liefert damit die gleichen 372 Zeilen wie die vorige Abfrage.

Bedingungen abgleichen

Bisher haben Sie Bedingungen zu sehen bekommen, die einen genauen String, einen String-Bereich oder eine String-Menge identifizieren. Es gibt jedoch noch einen letzten Bedingungstyp, der teilweise Übereinstimmungen von Strings erkennen kann. Sie können zum Beispiel alle Kunden ausfindig machen, deren Nachname mit »Q« beginnt. Mit einer eingebauten Funktion können Sie den ersten Buchstaben der Spalte `last_name` herausziehen, wie in folgender Anweisung:

```
mysql> SELECT last_name, first_name
```

```
-> FROM customer
```

```
-> WHERE left(last_name, 1) = 'Q';
```

```
+-----+-----+
| last_name   | first_name |
+-----+-----+
| QUALLS      | STEPHEN    |
| QUINTANILLA | ROGER      |
| QUIGLEY     | TROY       |
+-----+-----+
```

```
3 rows in set (0.00 sec)
```

Die eingebaute Funktion `left()` kann dies zwar tun, gibt Ihnen aber wenig Flexibilität. Stattdessen können Sie auch Suchausdrücke mit Wildcard-Zeichen formulieren, wie es im nächsten Abschnitt beschrieben wird.

Verwendung von Wildcards

Wenn Sie sich für teilweise Übereinstimmungen von Strings interessieren, könnte Sie Folgendes interessieren:

- Strings mit einem bestimmten Anfangs-/Endbuchstaben
- Strings, die mit einem bestimmten Teilstring anfangen oder enden
- Strings, in denen irgendwo ein bestimmtes Zeichen vorkommt
- Strings, in denen irgendwo ein bestimmter Teilstring vorkommt
- Strings, die unabhängig von den enthaltenen Zeichen ein bestimmtes Format aufweisen

Mit bestimmten Suchausdrücken können Sie diese und andere Teilübereinstimmungen von Strings mithilfe der Wildcard-Zeichen aus Tabelle 4-4 ausfindig machen.

Tabelle 4-4: Wildcard-Zeichen

Wildcard-Zeichen	Passt auf
_	genau ein Zeichen
%	beliebig viele Zeichen (einschließlich null)

Der Unterstrich ist Platzhalter für ein einzelnes und das Prozentzeichen für beliebig viele Zeichen. Wenn Sie Bedingungen mit solchen Suchausdrücken formulieren, verwenden Sie den `LIKE`-Operator:

```
mysql> SELECT last_name, first_name
```

```
    -> FROM customer
```

```
    -> WHERE last_name LIKE '_A_T%S';
```

```
+-----+-----+
```

```
| last_name | first_name |
```

```
+-----+-----+
```

```
| MATTHEWS | ERICA      |
```

```
| WALTERS  | CASSANDRA |
```

```
| WATTS    | SHELLY    |
```

```
+-----+-----+
```

```
3 rows in set (0.00 sec)
```

Der Suchausdruck im obigen Beispiel spezifiziert Strings, die an zweiter Stelle ein »A« und an vierter Stelle ein »T« aufweisen, worauf beliebig viele Zeichen folgen – aber am Ende muss ein »S« stehen. Tabelle 4-5 zeigt einige weitere Suchausdrücke zusammen mit ihren Interpretationen.

Tabelle 4-5: Beispiele für Suchausdrücke

Suchausdruck	Interpretation
F%	Strings, die mit »F« anfangen
%t	Strings, die mit »t« aufhören
%bas%	Strings, die den Teilstring »bas« enthalten
__t_	vier Zeichen lange Strings mit einem »t« an dritter Position
---_---_---	elf Zeichen lange Strings mit Bindestrichen an der vierten und siebten Position

Die Wildcard-Zeichen funktionieren gut für einfache Suchausdrücke, aber wenn es etwas komplizierter wird, können Sie auch mehrere Suchausdrücke miteinander verbinden:

```
mysql> SELECT last_name, first_name
```

```
    -> FROM customer
```

```
    -> WHERE last_name LIKE 'Q%' OR last_name LIKE 'Y%';
```

```
+-----+-----+
| last_name | first_name |
+-----+-----+
| QUALLS   | STEPHEN   |
| QUIGLEY  | TROY      |
```

```

| QUINTANILLA | ROGER      |
| YANEZ       | LUIS       |
| YEE         | MARVIN     |
| YOUNG       | CYNTHIA    |

```

```
+-----+-----+
```

```
6 rows in set (0.00 sec)
```

Diese Abfrage findet alle Kunden, deren Nachnamen mit »Q« oder »Y« anfangen.

Verwendung von regulären Ausdrücken

Wenn Ihnen Wildcards nicht flexibel genug sind, können Sie Suchausdrücke mit regulären Ausdrücken konstruieren. Ein regulärer Ausdruck ist im Wesentlichen ein Suchausdruck auf Steroiden. Wenn Sie SQL-Neuling sind, aber bereits mit Sprachen wie Perl programmiert haben, kennen Sie sich mit regulären Ausdrücken bestens aus. Sollten Sie jedoch noch nie mit ihnen zu tun gehabt haben, lesen Sie am besten *Reguläre Ausdrücke* von Jeffrey Friedl (O'Reilly), da dieses Thema viel zu umfangreich ist, um es in diesem Buch behandeln zu können.

Die obige Abfrage (alle Kunden zu finden, deren Nachnamen mit »Q« oder »Y« anfangen) würde in der MySQL-Implementierung von regulären Ausdrücken folgendermaßen aussehen:

```

mysql> SELECT last_name, first_name
      -> FROM customer
      -> WHERE last_name REGEXP '^[QY]';

```

```
+-----+-----+
```

```
| last_name | first_name |
```

```

+-----+-----+
| YOUNG          | CYNTHIA      |
|
| QUALLS         | STEPHEN      |
|
| QUINTANILLA   | ROGER        |
|
| YANEZ          | LUIS         |
|
| YEE            | MARVIN       |
|
| QUIGLEY        | TROY         |
|
+-----+-----+

```

6 rows in set (0.16 sec)

Der Operator `regexp` nimmt einen regulären Ausdruck (hier `'^[QY]'`) und wendet ihn auf den Ausdruck auf der linken Seite der Bedingung an (die Spalte `last_name`). Jetzt enthält die Abfrage eine einzelne Bedingung mit einem regulären Ausdruck anstelle zweier mit Wildcards.

Oracle Database und Microsoft SQL Server unterstützen ebenfalls reguläre Ausdrücke. Bei Oracle Database verwendet man die Funktion `regexp_like` anstelle des Operators `regexp` und bei SQL Server den Operator `like`.

NULL: ein böses Wort

Ich habe mich darum gedrückt, solange ich konnte, aber jetzt lässt es sich nicht mehr umgehen, dieses Thema, das Unsicherheit, Angst und Entsetzen hervorruft: der `null`-Wert. `null` ist die Abwesenheit eines Werts. Bevor ein Angestellter gefeuert wird, wird seine `end_date`-Spalte in der `employee`-Tabelle auf `null` gesetzt. Es gibt einfach keinen Wert, den man der Spalte `end_date` in dieser Situation vernünftigerweise zuweisen könnte. `null` ist jedoch eine wackelige Angelegenheit, da es mehrere Varianten davon gibt:

Nicht zutreffend

Zum Beispiel die Employee-ID-Spalte für ein am Geldautomaten getätigtes Bankgeschäft.

Wert noch nicht bekannt

Wenn zum Beispiel zu dem Zeitpunkt, da die Daten eines Kunden angelegt werden, dessen Federal ID noch unbekannt ist.

Wert nicht definiert

Wenn zum Beispiel ein Konto für ein Produkt angelegt wird, das noch nicht in die Datenbank eingetragen wurde.



Manche Theoretiker sind der Ansicht, es müsse für alle diese (und andere) Situationen jeweils unterschiedliche Ausdrücke geben, aber die meisten Praktiker sind sich einig, dass das viel zu verwirrend wäre.

Bei der Arbeit mit `null` müssen Sie an Folgendes denken:

- Ein Ausdruck kann *null sein*, aber nicht *gleich null*.
- `null` und `null` sind nie gleich.

Ob ein Ausdruck `null` ist, kann man mit dem Operator `is null` testen:

```
mysql> SELECT rental_id, customer_id
```

```
-> FROM rental
```

```
-> WHERE return_date IS NULL;
```

```
+-----+-----+
```

```
| rental_id | customer_id |
```

```
+-----+-----+
```

```
|      11496 |          155 |
```

11541	335
11563	83
11577	219
11593	99
...	
15867	505
15875	41
15894	168
15966	374

+-----+-----+

183 rows in set (0.01 sec)

Diese Abfrage liefert alle Ausleihvorgänge, bei denen Filme nie zurückgegeben wurden. Hier sehen Sie die gleiche Abfrage mit = null anstelle von is null:

```
mysql> SELECT rental_id, customer_id
      -> FROM rental
      -> WHERE return_date = NULL;
```

Empty set (0.01 sec)

Die Abfrage wird korrekt geparkt und ausgeführt, gibt aber keine Zeilen zurück. Unerfahrenen SQL-Programmierern unterläuft häufig dieser Ausrutscher, und der Datenbankserver meldet dann keinen Fehler. Seien Sie also vorsichtig bei der Formulierung von Bedingungen, die null testen sollen.

Wenn Sie sehen möchten, ob einer Spalte ein Wert zugewiesen wurde, verwenden Sie den Operator `is not null`:

```
mysql> SELECT rental_id, customer_id, return_date
```

```
    -> FROM rental
```

```
    -> WHERE return_date IS NOT NULL;
```

```
+-----+-----+-----+
```

```
| rental_id | customer_id | return_date          |
```

```
+-----+-----+-----+
```

```
|          1 |          130 | 2005-05-26 22:04:30 |
```

```
|          2 |          459 | 2005-05-28 19:40:33 |
```

```
|          3 |          408 | 2005-06-01 22:12:39 |
```

```
|          4 |          333 | 2005-06-03 01:43:41 |
```

```
|          5 |          222 | 2005-06-02 04:33:21 |
```

```
|          6 |          549 | 2005-05-27 01:32:07 |
```

```
|          7 |          269 | 2005-05-29 20:34:53 |
```

```
...
```

16043	526	2005-08-31 03:09:03
16044	468	2005-08-25 04:08:39
16045	14	2005-08-25 23:54:26
16046	74	2005-08-27 18:02:47
16047	114	2005-08-25 02:48:48
16048	103	2005-08-31 21:33:07
16049	393	2005-08-30 01:01:12

+-----+-----+-----+

15861 rows in set (0.02 sec)

Diese Version der Abfrage liefert alle Ausleihvorgänge, bei denen die Filme zurückgegeben wurden, was den Großteil der Zeilen in der Tabelle betrifft (15.861 von 16.044).

Bevor wir das Thema null wieder verlassen, möchte ich eine weitere mögliche Falle aufzeigen. Angenommen, Sie müssten alle Ausleihvorgänge finden, die nicht zwischen Mai und August 2005 zurückgegeben wurden. Instinktiv tun Sie zunächst Folgendes:

```
mysql> SELECT rental_id, customer_id, return_date
```

```
    -> FROM rental
```

```
    -> WHERE return_date NOT BETWEEN '2005-05-01' AND '2005-09-01';
```

+-----+-----+-----+

```

| rental_id | customer_id | return_date          |
+-----+-----+-----+
|      15365 |           327 | 2005-09-01 03:14:17 |
|      15388 |            50 | 2005-09-01 03:50:23 |
|      15392 |           410 | 2005-09-01 01:14:15 |
|      15401 |           103 | 2005-09-01 03:44:10 |
|      15415 |           204 | 2005-09-01 02:05:56 |
...
|      15977 |           550 | 2005-09-01 22:12:10 |
|      15982 |           370 | 2005-09-01 21:51:31 |
|      16005 |           466 | 2005-09-02 02:35:22 |
|      16020 |           311 | 2005-09-01 18:17:33 |
|      16033 |           226 | 2005-09-01 02:36:15 |
|      16037 |            45 | 2005-09-01 02:48:04 |
|      16040 |           195 | 2005-09-02 02:19:33 |
+-----+-----+-----+

```

62 rows in set (0.01 sec)

Es stimmt zwar, dass diese 62 Ausleihvorgänge außerhalb unseres Zeitfensters von Mai bis August beendet wurden, aber wenn Sie die Daten genau anschauen, erkennen Sie, dass alle Zeilen im `return_date` nicht `null` sind. Was ist aber mit den 183 Ausleihvorgängen, deren Filme nie zurückgegeben wurden? Man kann durchaus sagen, dass diese 183 Zeilen ebenfalls in der Ergebnismenge enthalten sein sollten, weil die Filme nicht zwischen Mai und August zurückkamen. Um also die korrekte Antwort zu finden, müssen Sie auch die Möglichkeit einbeziehen, dass manche Zeilen in der Spalte `return_date` den Wert `null` aufweisen:

```
mysql> SELECT rental_id, customer_id, return_date
```

```
-> FROM rental
```

```
-> WHERE return_date IS NULL
```

```
-> OR return_date NOT BETWEEN '2005-05-01' AND '2005-09-01';
```

```
+-----+-----+-----+
```

```
| rental_id | customer_id | return_date |
```

```
+-----+-----+-----+
```

```
| 11496 | 155 | NULL |
```

```
| 11541 | 335 | NULL |
```

```
| 11563 | 83 | NULL |
```

```
| 11577 | 219 | NULL |
```

```
| 11593 | 99 | NULL |
```

...

	15939		382		2005-09-01 17:25:21	
	15942		210		2005-09-01 18:39:40	
	15966		374		NULL	
	15971		187		2005-09-02 01:28:33	
	15973		343		2005-09-01 20:08:41	
	15977		550		2005-09-01 22:12:10	
	15982		370		2005-09-01 21:51:31	
	16005		466		2005-09-02 02:35:22	
	16020		311		2005-09-01 18:17:33	
	16033		226		2005-09-01 02:36:15	
	16037		45		2005-09-01 02:48:04	
	16040		195		2005-09-02 02:19:33	

+-----+-----+-----+

245 rows in set (0.01 sec)

Nun enthält die Ergebnismenge die 62 Ausleihvorgänge, bei denen die Filme außerhalb des Fensters von Mai bis August zurückgegeben wurden, und die 183 Fälle, bei denen die Filme nie zurückkamen, was zusammen 245 Zeilen ergibt.

Wenn Sie mit unbekanntenen Datenbanken arbeiten, sollten Sie immer zuerst herausfinden, welche Spalten einer Tabelle null-Werte erlauben. So können Sie in Ihren Filterbedingungen Vorkehrungen treffen, damit Ihnen keine Daten durchrutschen.

Testen Sie Ihr Wissen

Die folgenden Übungen prüfen, ob Sie Filterbedingungen verstanden haben. Die Lösungen finden Sie in Anhang B.

Für die ersten beiden Übungen wird die folgende Untermenge an Zeilen aus der payment-Tabelle verwendet:

```
+----+-----+----+-----+
| payment_id | customer_id | amount | date(payment_date) |
+----+-----+----+-----+
|          101 |           4 |    8.99 | 2005-08-18         |
|          102 |           4 |    1.99 | 2005-08-19         |
|          103 |           4 |    2.99 | 2005-08-20         |
|          104 |           4 |    6.99 | 2005-08-20         |
|          105 |           4 |    4.99 | 2005-08-21         |
|          106 |           4 |    2.99 | 2005-08-22         |
|          107 |           4 |    1.99 | 2005-08-23         |
|          108 |           5 |    0.99 | 2005-05-29         |
|          109 |           5 |    6.99 | 2005-05-31         |
```

	110		5		1.99		2005-05-31	
	111		5		3.99		2005-06-15	
	112		5		2.99		2005-06-16	
	113		5		4.99		2005-06-17	
	114		5		2.99		2005-06-19	
	115		5		4.99		2005-06-20	
	116		5		4.99		2005-07-06	
	117		5		2.99		2005-07-08	
	118		5		4.99		2005-07-09	
	119		5		5.99		2005-07-09	
	120		5		1.99		2005-07-09	

+-----+-----+-----+-----+

Übung 4-1

Welche Payment-IDs würden durch folgende Filterbedingungen zurückgegeben?

```
customer_id <> 5 AND (amount > 8 OR date(payment_date) =
'2005-08-23')
```

Übung 4-2

Welche Payment-IDs würden durch diese Filterbedingungen zurückgegeben?

```
customer_id = 5 AND NOT (amount > 6 OR date(payment_date) =  
'2005-06-19')
```

Übung 4-3

Erstellen Sie eine Abfrage, die alle Zeilen aus der payment-Tabelle liefert, bei denen der Betrag entweder 1,98, 7,98 oder 9,98 ist.

Übung 4-4

Erstellen Sie eine Abfrage, die alle Kunden findet, deren Nachname an zweiter Stelle ein »A« und dann irgendwo nach dem »A« ein »W« enthält.

Symbole

- ' (einfache Anführungszeichen) 118
- () (runde Klammern)
 - Bedingungsauswertung 68
 - Priorisieren arithmetischer Operationen 128
 - Reihenfolge von Unterabfragen 113
 - Unterabfrage umschließend 53
- * (Asterisk)
 - in Aggregatsfunktion 147, 149
 - in select-Klausel 48
- \ (Backslash)
 - als Escape-Zeichen 118
- % (Prozentzeichen)
 - als Modulo-Operator 130
- + (Verkettungs-)Operator 127
- || (vertikale Balken)
 - als Verkettungsoperator 120

A

- Abfragen *siehe auch* Unterabfragen
 - ausführen 45, 47
 - Bedingungslogik 199
 - Case-Ausdruck 200, 209
 - gruppieren 145, 148
 - Indizes 221, 231
 - Joins 87, 93
 - Cross Joins 190

- drei und mehr Tabellen verbinden 93
- Natural Joins 196, 197
- Self Joins 98
- Klauseln 47, 65
 - from 53, 57
 - group by 60
 - having 60
 - ohne 19
 - order by 61, 65
 - where 58, 60
- Mehrbenutzerdatenbanken 211
- Versionierung 212
- Views 55
 - zusammengesetzte 105
 - Mengenoperationsregeln 111, 113
- abgleichen
 - Bedingungen 79
 - Wildcards 79
- abschneiden, Strings 116
- absteigende Sortierreihenfolge 63
- Aggregatfunktionen 146, 152
 - count() 147
 - count(*) 148
- Aggregation
 - Views zur Zusammenstellung von Daten einsetzen 241
- Aliase
 - für Tabellen 98
 - Spalten-, einfügen 50
 - Tabellen-, definieren 57
- all-Operator 164
- Amazon Web Services (AWS) 299
- American National Standards Institute (*siehe auch* ANSI) 91
- analytische Funktionen 265
 - Datenfenster 266, 273, 276, 277
 - Rangfolgefunktionen 268

- Reporting-Funktionen 274
- Spaltenwerte verketteten 281
- Übungen zu 282
- Anführungszeichen, eingebettete 118
- ANSI (American National Standards Institute) 7
 - Join-Syntax 91
- ANSI-Modus 117
- Anweisungen
 - create table 31
 - dynamische SQL-Ausführung in MySQL 259
 - für aktualisierbare Views 243
 - Gültigkeitsbereich 159
 - insert 34
 - Werte 36
 - Klassen 8
 - Problembhebung 39
 - Schemata, erstellen 29
 - select, Abfrageklauseln 47, 52
 - übergeordnete 159
 - Unterabfragen 159
 - anwenden 173, 182
 - korrelierte 169, 173
 - nicht-korrelierte 161, 168
 - Typen 160
- any-Operator 166
- Anzeigen von Indizes 223
- Apache Drill 14, 301
 - Dateien abfragen 302
 - dfs-(Distributed-File-System-)Plug-in 302
 - MySQL abfragen 304
- Archive-Speichermodul 218
- Argumente
 - einargumentige numerische Funktionen 129
 - größer als null 127
 - truncate()-Funktion 132

arithmetische Operationen in Filterbedingungen 70

arithmetische Operatoren 128

as-Schlüsselwort

 einsetzen mit Spaltenaliasen 51

 einsetzen mit Tabellenaliasen 57

atomare Sperren 134

aufsteigende Sortierreihenfolge 63

Ausdrücke

 Aggregatfunktionen 151

 Bedingungen, erstellen 70

 case 200, 209

 durchsuchen 80

 gruppieren 154

 reguläre, anwenden 81

 select-Anweisungen 49

 Unterabfragen als Generatoren 180

auswerten, Bedingungen 67, 70

Autocommit-Modus 215

Auto-Increment-Feature, starten 34

avg-Funktion 274

AWS (Amazon Web Services) 299

B

B-Baum-Indizes 227

BDB-Speichermodul 218

bearbeiten

 Daten

 Gruppen erzeugen 153

 mit korrelierten Unterabfragen 172

 Strings 121

 temporale Daten 139, 143

Bedingungen

 auswerten 67, 70

 erstellen 70

 Filter 59

 Gleichheits- 70

- Gruppen filtern 157
- Typen 70, 81
- Updates 207
- Bedingungslogik 199
 - Case-Ausdruck 200, 209
- Befehle
 - commit 213, 216
 - rollback 213, 216
 - show index 223
- Beispiele, Sakila-Datenbank XVI, 17, 18, 41
- Benutzer
 - Mehrbenutzerdatenbanken 211
 - Transaktionen 213, 220
- Berechtigungen 302
- Bereiche
 - Bedingungen 73
 - Strings 75
- Bereichspartitionierung 287
- between-Operator 74
- Beziehungen, ER-Diagramm 317
- Big Data 297
 - Apache Drill 301
 - Cloud Computing 299
 - Hadoop 14, 298, 301
 - NoSQL und Dokumentendatenbanken 14, 298, 301
 - Zukunft 315
- Bitmap-Indizes 228
- Blattknoten 227

C

- C# (Programmiersprache), Toolkits zur SQL-Integration 10
- Case-Ausdruck 200, 209
- cast()-Funktion 137, 143
- ceil()-Funktion 131
- char()-Funktion 120
- CHAR-Datentyp 115

- Check Constraints 30, 232
- CLOB-Datentyp 116
- Cloud Computing 299
- Clustering 296
- Codd, Dr. E. F. 4, 7
- columns-View 252
- commit-Befehl 213, 216
- Common Table Expressions (CTEs) 179
- Composite-Partitionierung 293
- concat()-Funktion 120, 125
- Constraints 232
 - Abfrage, die die Anzahl an Primärschlüssel-Constraints liefert 258
 - Check Constraints 30
 - erstellen 232
 - Fremdschlüssel 33
 - Informationen zu information_schema abrufen 253
 - Informationen zu Primärschlüssel-Constraints abrufen 256
 - Primärschlüssel 30
 - Übungen 235
- Coordinated Universal Time (UTC) 134
- count()-Funktion 147, 187, 274
- count(*)-Funktion 148
- create index-Befehl 223
- create table-Anweisungen 31
 - Abfragen erzeugen mithilfe von information_schema 256
 - Wohlgeformtheit prüfen 258
- create view-Anweisungen 237
- Cross Joins 190, 196
- CTEs (Common Table Expressions) 179

D

- Data Dictionaries 9, 250
- Data Feed generieren 177
- date_add()-Funktion 142, 193
- datediff()-Funktion 142
- Dateien

- mit Apache Drill abfragen 302
- Datenbanken 1, 7
 - Big Data 297, 301
 - Clustering 296
 - Constraints 232
 - ER-Diagramm 317
 - in MySQL auflisten 18
 - Indizes 221, 231
 - Mehrbenutzer- 211
 - Sperren 212
 - MySQL 13
 - Natural Joins 196, 197
 - neue Technologien 14
 - nicht-relationale Datenbanksysteme 2
 - relationale Modelle 4
 - Sharding 296
 - Terminologie 7
 - Transaktionen 213, 220
 - Zukunft 315
- Datenfenster 266, 273, 276, 277
- Datenquellen, Unterabfragen als 173
- Datentypen 20, 27
 - numerische Daten 23, 25
 - temporale Daten 25, 27
 - Zeichen 20
 - anwenden in Strings 115, 128
 - CHAR 115
 - CLOB 116
 - Text 116
 - varchar 115
- Datum/Uhrzeit-Spalten, befüllen 136
- Datumsangaben *siehe auch* temporale Datentypen
 - Bestandteile 136
 - erzeugen 138
 - formatieren 26

- String-in-Datum-Konvertierungen 137
 - unzulässige Konvertierungen 40
 - zurückgeben 139
- Deadlocks 216
- deallocate-Anweisungen 259
- definieren
 - Abfrage-Joins 87, 93
 - null 32
 - Tabellenaliase 57
 - Zeichenspalten 20
- dense_rank-Funktion 269
- Deployment-Prüfung für Schemaobjekte 258
- describe-Befehl (desc) 31
 - Views betrachten 238
- Dezimalpunkte (numerische Datentypen) 25
- Diagramm (ER) 317
- Dictionaries 9
- distinct-Schlüsselwort 51
- Division durch null, Fehler 206
- Dokumentendatenbanken 298
- Drei-Wege-Outer-Joins 189
- Duplikate
 - Zeilen, löschen 51
- Durability 215
- durchsuchen *siehe auch* Indizes
 - Ausdrücke 80
 - Case-Ausdrücke 200
 - Wildcards 79
- dynamische SQL-Ausführung 259

E

- Ebenen von Sperren 212
- einargumentige numerische Funktionen 129
- einfache Anführungszeichen (') 118
- einfache Case-Ausdrücke 202
- einfügen

- Daten in Tabellen 33
- Indizes 222
- Intervalle 141
- Schlüsselwörter 51
- Spaltenalias 50
- eingebaute Funktionen *siehe auch* Funktionen
 - numerische 129
 - select-Anweisungen 49
 - Strings 121
- eingebettete Anführungszeichen 118
- einzelne Spalte
 - gruppieren 153
 - Unterabfragen 162
- Elemente, zählen 150
- Englisch-Zeichensätze 21
- Entitäten 7
- Entity-Relationship-(ER-)Diagramm 317
- Entwurf, Tabellen 27
- ER-(Entity-Relationship-)Diagramm 317
- Ergebnismengen 7
 - Case-Ausdrücke 203
 - zusammengesetzte Abfragen, sortieren 111
- erstellen
 - Bedingungen 70
 - Mehrspaltenindizes 226
 - Schemaanweisungen 29
 - Strings, zeichenweise 120
- erzeugen, Tabellen 175, 191
- Escape-Zeichen für Strings 118
- except-Operationen 102
- except-Operator 109
- execute-Anweisungen 259
- exists-Operator 170
- explizite Gruppen 149
- extract()-Funktion 141

F

fehlender Fremdschlüssel 39

Fehler *siehe auch* Problembhebung

 Division durch null 206

Fenster 273, 276, 277

Filterbedingungen in where-Klauseln 67

filtern

 Bedingungen 59

 auswerten 67, 70

 erstellen 70

 Typen 70, 81

 Gruppen 147, 157

 null 82, 85

Fließkommamatypen 24

Fließkommazahl, Genauigkeit 131

floor()-Funktion 131

formatieren

 Constraints 232

 Datumsangaben 26, 136

 Indizes 222

 Mehrspaltenindizes 226

 Sonderzeichen 119

 Tabellen 27, 33

 Views 55

Fremdschlüssel 4, 6, 7

 Constraints 33, 232

 ER-Diagramm 317

 fehlender 39

 selbstreferenzierender 98

Friedl, Jeffrey 81

from-Klausel 53, 57

 für aktualisierbare Views 243

 in select-Anweisungen 57

 Join-Reihenfolge und 95

 on-Subklausel 89

ANSI-Join-Syntax in 92

using-Subklausel 91

Funktionen

Aggregat- 146, 152

count() 147

count(*) 148

cast() 137, 143

ceil() 131

char() 120

concat() 120, 125

date_add() 142, 193

datediff() 142

Datumsangaben

erzeugen 138

zurückgeben 139

einargumentige numerische 129

extract() 141

floor() 131

getutcdate() 134

Konvertierung 143

last_day() 141

length() 121

locate() 122

mod() 129

now() 19

position() 122

pow() 130

quote() 119

replace() 127

round() 131

sign() 133

strcmp() 123

Strings, zurückgeben 121, 141

str_to_date() 139

stuff() 128

truncate() 132

Zahlen, zurückgeben 142

G

Ganzzahl, Integer 24

Genauigkeit

Fließkommatypen 25

Zahlen, steuern 131

geringe Kardinalität, Daten 228

getutcdate()-Funktion 134

Gleichheitsbedingungen 70

globaler Index 287

GMT (Greenwich Mean Time) 134

Go, Toolkits zur SQL-Integration 11

Granularitäten von Sperren 212

Greenwich Mean Time (GMT) 134

Größe ändern bei Texttypen 23

Größenordnung (Fließkommatypen) 25

group by-Klausel 60

group_concat-Funktion 281

gruppieren 145, 148

Aggregatfunktionen 148, 152

Ausdrücke 154

einzelne Spalte 153

Elemente, zählen 150

erzeugen 153

filtern 157

implizite oder explizite Gruppen 149

mehrere Spalten 154

Rollups 155

Unterabfragen 177

Gültigkeitsbereich, Anweisungen 159

H

Hadoop 14, 298, 301

Hash-Funktion 292

Hash-Partitionierung 292
having-Klausel 60, 166
hierarchische Datenbanksysteme 2
Hive 301
hohe Kardinalität, Daten 228
horizontale Partitionierung 287

I

implizite Gruppen 149
Indizes 221, 231
 Abfrage, die die Anzahl an Indizes liefert 258
 anwenden 229
 anzeigen 223
 B-Baum 227
 Bitmap- 228
 erstellen 222
 Informationen zu information_schema abrufen 253
 Knoten 227
 löschen 225
 mehrere Spalten 226
 modifizieren 230
 partitionieren 287
 show-Befehl 223
 Text 228
 Typen 227
 Übungen 235
 unique- 225
 Volltext 229
information_schema-Metadaten
 Apache Drill 302
information_schema-Objekte 250
 columns-View 261
 Information zu Constraints 253
 Information zu Tabellenindizes 253
 verfügbare Views in MySQL 8.0 254
 Views in 251

- Inner Joins 89
- InnoDB-Speichermodul 218
- in-Operator 162
- insert-Anweisung 34
 - Daten über Views einfügen 246
 - Werte 36
- Integer
 - Ganzzahl 24
 - runden 131
- Integration von Toolkits 10
- intersect-Operatoren 108
- Intervalle
 - einfügen 141
 - Typen 140

J

- Java
 - SQL-Anweisungen und 259
 - Toolkits zur SQL-Integration 10
- JDBC-Treiber für MySQL 304
- Joins 87, 93
 - ANSI-Syntax für 91
 - Cross Joins 190, 196
 - drei oder mehr Tabellen verbinden 93
 - Join-Abfolge angeben 95
 - gleiche Tabelle mehrfach nutzen 97
 - Natural Joins 196, 197
 - Outer Joins 185, 190
 - Bedingungslogik 199
 - Drei-Wege- 189
 - Left/Right 188
 - Partition-Wise Joins 295
 - Self Joins 98
 - Views mit anderen Tabellen oder Views verknüpfen 239
- JSON
 - in Dokumentendatenbanken 298

K

kartesische Produkte 88

Klassen, Anweisungen 8

Klauseln

- Abfragen 47, 65

 - from 53, 57

 - group by 60

 - having 60

 - ohne 19

 - order by 36, 61, 65

 - where 58, 60

- select (Aggregatfunktionen) 146

- where (gruppierende Filterbedingungen) 157

Knoten, Indizes 227

Kollation 77

kombinieren, mehrere Tabellen 105

Kommandozeilentool mysql 11, 19

konfigurieren *siehe auch* formatieren

- Standardzeichensätze 22

Konventionen im Buch XV

Konvertierungen

- Funktionen 143

- String-in-Datum 137

- unzulässige Datumsangaben 40

- Werte 143

korrelierte Unterabfragen 161, 169, 173

L

lag-Funktion 279

last_day()-Funktion 141

lead-Funktion 279

Left Outer Joins 186, 188

length()-Funktion 121

Lesesperren 212

Listenpartitionierung 290

Literale 49

locate()-Funktion 122

lokaler Index 287

löschen

 Duplikate 51

 Indizes 225

 Tabellendaten 38

Lösungen zu den Übungsaufgaben 319

M

max-Funktion 274

Mehrbenutzerdatenbanken 211

mehrere Bedingungen, runde Klammern () 68

mehrere Spalten

 gruppieren 154

 Indizes 226

 Unterabfragen 167

mehrere Tabellen

 Abfragen

 Joins 87, 93

 Joins mit drei und mehr Tabellen 93

 Self Joins 98

 union-Operatoren 105

Mehrzeilenunterabfragen 162

MEMORY-Speichermodul 218

Mengen *siehe auch* Ergebnismengen

 Grundlagen der Mengenlehre 101, 105

 Mengenoperatoren 105, 110

 nicht überschneidende 108

 Regeln für Mengenoperationen 111, 113

Mengenlehre 101, 105

Merge-Speichermodul 218

Metadaten 9, 249

 Apache Drill 302

 Informationen in 249

 information_schema-Objekte 250

- nutzen in Skripten zur Schemagenerierung 255
- nutzen zur Deployment-Überprüfung 258
- nutzen, um dynamisch SQL zu generieren 259
- Partitionen von Tabellen 288
- Übungen zu 263
- veröffentlicht von Datenbankservern 250
- min-Funktion 274
- Mitgliedschaftsbedingungen 77
- mod()-Funktion 129
- modifizieren
 - Abfragen, Inner Joins 89
 - Daten, mit Gleichheitsbedingungen 72
 - Indizes 230
 - Tabellen 33, 39
- Modus prüfen und ändern für MySQL 117
- MongoDB 299
- Multibyte-Zeichensätze 21
- Multiparent-Hierarchien 4
- MyISAM-Speichermodul 218
- MySQL
 - aktualisierbare Views 243
 - Datenbanken auflisten 18
 - definieren 13
 - dynamisch SQL ausführen 259
 - Indizes, einfügen 222
 - information_schema-Datenbank 250
 - installieren 17
 - JDBC-Treiber 304
 - Join-Abfolge festlegen 95
 - mit Apache Drill abfragen 304
- mysql-Kommandozeilentool 11
 - anwenden 19
 - create table-Anweisung 31
 - xml-Option 37

N

- Natural Joins 196, 197
- negative Werte, vorzeichenbehaftete Daten 133
- Netzwerkdatenbanksysteme 3
- NewSQL 14
- nicht überschneidende Mengen 108
- nicht vorzeichenbehaftete Daten 24
- nicht-eindeutiger Primärschlüssel 39
- nicht-korrelierte Unterabfragen 161, 168
- nicht-prozedurale Sprachen 10
- nicht-relationale Datenbanksysteme 2
- Normalisierung 6, 28
- NoSQL 14, 298, 301
- not in-Operator 79
- not-Operator, anwenden 69
- now()-Funktion 19
- Nullen 24
 - Argumente größer als 127
 - Division durch null, Fehler 206
- Nullwerte 32
 - Aggregatfunktionen 151
 - Case-Ausdrücke 208
 - definieren 32
 - filtern 82, 85
 - Outer Joins 188
- numerische Daten, erzeugen 128, 133
- numerische Datentypen 23, 25
- numerische eingebaute Funktionen 129
- numerische Platzhalter, sortieren mit 64
- numerische Schlüsseldata, erzeugen 34

O

- Onlineressourcen
 - MySQL 17
 - Sakila-Datenbank XVI, 17, 18

- on-Subklausel der from-Klausel
 - ANSI-Join-Syntax in 92
- Operationen, Mengen 101, 105, 111, 113
- Operatoren 127
 - all 164
 - any 166
 - arithmetische 128
 - Bedingungen, erstellen 70
 - between 74
 - except 109
 - exists 170
 - in 162
 - intersect 108
 - Mengen- 105, 110
 - not 69
 - not in 79
 - regexp 124
 - union 105
 - Verkettungs- (+) 127
- Optimierung von Tabellen 28
- Optimierungshinweise (Optimizer Hints) 10
- Oracle Database
 - dynamische SQL-Ausführung 259
 - Einfügen und Aktualisieren von Daten über Views 247
 - Join-Abfolge festlegen 95
 - Metadaten 250
- Oracle Text 229
- Oracle-Datenbank
 - Größe von char, varchar2 und clob 23
 - temporäre Tabellen 55
- Oracle-Exadata-Plattform 296
- Oracle-PL/SQL-Sprache 259
- order by-Klausel 36, 61, 65
 - Unterabfragen 181
- Outer Joins 90, 185, 190

- Bedingungslogik 199
- Drei-Wege- 189
- Left Outer Joins 186
- Left/Right 188
- over-Klausel 267, 273, 274

P

- Packet-Capture-Dateien (PCAP) 302
- partition by hash-Schlüsselwort 292
- partition by list-Schlüsselwort 290
- partition by range-Schlüsselwort 287
- partition by-Klausel 267, 268, 273, 276
- Partition Pruning 295
- Partitionieren von Datenbanken
 - Bereichspartitionierung 287
 - Composite-Partitionierung 293
 - Gründe 285
 - Hash-Partitionierung 292
 - Indizes partitionieren 287
 - Listenpartitionierung 290
 - Metadaten 288
 - Partitionen reorganisieren 288
 - Tabellen partitionieren 287
 - Vorteile 295
- Partitionierungsfunktion 287
- Partitionierungsschlüssel 287
- Partition-Wise Joins 295
- PL/SQL-Sprache 10, 259
- Platzhalter, numerische sortieren 64
- position()-Funktion 122
- positive Werte, vorzeichenbehaftete Daten 133
- pow()-Funktion 130
- Präzedenz
 - numerische Daten, erzeugen 128
 - von Mengenoperationen 112
- prepare-Anweisungen 259

- Presto 301
- Primärschlüssel 4
 - Constraints 30, 232
 - Anzahl abfragen 258
 - Informationen abrufen zu 256
 - nicht-eindeutiger 39
- Problembhebung
 - Anweisungen 39
 - Division durch null, Fehler 206
 - Spaltenwerte 40
- prozedurale Sprachen 10
- Prüfen auf Vorhandensein 204
- Python, Toolkit zur SQL-Integration 11

Q

- quote()-Funktion 119

R

- range between-Unterklausele 278
- Rangfolgefunktionen 268
- rank-Funktion 269
- referenzieren, übergeordnete Abfrage 54
- Regeln, Mengenoperationen 111, 113
- regexp-Operatoren 124
- reguläre Ausdrücke, anwenden 81
- relationale Datenbanken 4, 13
 - Zukunft 315
- reorganize partition-Schlüsselwort 288
- replace()-Funktion 127
- Reporting
 - Mehrbenutzerdatenbanken 211
 - Task-orientierte Unterabfragen 177
- Reporting-Funktionen 274
- Richtlinien für Mengenoperationen 104
- Right Outer Joins 188
- rollback-Befehl 213, 216

- Rollups, gruppieren 155
- round()-Funktion 131
- row_number-Funktion 269
- rows between-Unterklauseel 278
- rows unbounded preceding-Unterklauseel 277
- Ruby, Toolkit zur SQL-Integration 10
- runde Klammern (), Bedingungsauwertung 68
- runden
 - Integer 131
 - Zahlen 131

S

- Sakila-Datenbank 41
 - herunterladen XVI, 17, 18
- Savepoints (Transaktionen) 218
- Schaltjahre 141
- Schemata
 - Deployment überprüfen 258
 - erstellen 29
 - information_schema 250
 - Skripte zur Generierung 255
- Schlüssel
 - Fremd- 4
 - Constraints 33, 232
 - ER-Diagramm 317
 - fehlender 39
 - selbstreferenzierender 98
 - Primär- 4
 - Constraints 30, 232
 - nicht-eindeutiger 39
 - zusammengesetzte 4
- Schlüsselwörter, einfügen 51
- Schnittmengen 102
- Schreibsperrren 212
- Seitensperrren 212
- selbstreferenzierender Fremdschlüssel 98

- select-Anweisungen
 - Abfragen 47, 52
 - aktualisierbare Views 243
 - Views abfragen 239
- select-Klausel, Aggregatfunktionen 146
- select-Schlüsselwort 51
- Self Joins 98
- Sequences 34
- Server
 - Aggregatfunktionen 148, 152
 - Indizes, anzeigen 223
 - Natural Joins 196, 197
 - sperrern 212
 - SQL92-Join-Syntax 91
 - Transaktionen
 - beenden 216
 - Savepoints 218
 - starten 215
- SET-Befehl 117
- Sharding 296
- Shared-Disk/Shared-Cache-Cluster 296
- show databases-Anweisung 18
- show-Befehl, Indizes 223
- Sicherheit
 - Berechtigungen 302
 - Datensicherheit mit Views 240
- sign()-Funktion 133
- Single-Parent-Hierarchien 3
- skalare Unterabfragen
 - in order by-Klausel 181
- Skalar-Unterabfragen 161
- Sommerzeit 134
- sortieren
 - aufsteigende/absteigende Sortierreihenfolge 63
 - Kollation 77

- numerische Platzhalter 64
- zusammengesetzte Abfragen, Ergebnisse 111
- Spalten 4, 7
 - Abfrage, die die Anzahl an Spalten zurückliefert 258
 - Aliase, einfügen 50
 - befüllen 116
 - Check Constraints 30
 - Constraints 232
 - Datum/Uhrzeit, befüllen 136
 - einzelne Spalte
 - gruppieren 153
 - Unterabfragen 162
 - Indizes, einfügen 222
 - mehrere Spalten
 - gruppieren 154
 - Indizes 226
 - Unterabfragen 167
 - Natural Joins 196, 197
 - null 32
 - temporale Datentypen 26
 - vorzeichenbehaftete Daten 133
 - Werte verketteten 281
 - Werte, Problembehebung 40
 - Zeichen, definieren 20
- Spark 14, 301
- Spark SQL 301
- Speichermodule 212
 - auswählen 218
 - Sperrungen und 213
- Sperrungen 211
 - Deadlocks 216
 - Ebenen 212
- Sprachen
 - nicht-prozedurale 10
 - PL/SQL 10

- TransactSQL 10
- sprechende Schlüssel 6
- SQL (Structured Query Language)
 - definieren 7, 13
 - dynamisch erzeugen 259
 - neue Technologien 14
 - Verbreitung XIII
 - Zukunft 315
- SQL Server
 - Daten über Views einfügen und aktualisieren 247
 - dynamische SQL-Ausführung 259
 - Größe von char und varchar 23
 - Join-Abfolge festlegen 95
 - Metadaten und information_schema 250
 - XML-Ausgaben für Abfragen generieren 38
- SQL92-Join-Syntax 91
- Standardzeichensätze, definieren 22
- starten
 - Auto-Increment-Feature 34
 - Transaktionen 215
- Steuern der Genauigkeit von Zahlen 131
- STRAIGHT_JOIN-Schlüsselwort 95
- strcmp()-Funktionen 123
- strict-Modus 117
- Strings
 - abschneiden 116
 - anwenden 115, 128
 - bearbeiten 121
 - Bereiche 75
 - eingebaute Funktionen 121
 - erzeugen 116
 - Escape-Zeichen 118
 - Funktionen
 - Rückgabe von Strings 125
 - Rückgabe von Zahlen 121

- zurückgeben 125, 141
- konvertieren 143
- SQL-Anweisungen an Datenbankserver übergeben als 259
- String-in-Datum-Konvertierungen 137
- temporale Daten, erzeugen 136
- umwandeln in datetimes in MySQL 40
- Wildcards, anwenden 79
- str_to_date()-Funktion 139
- stuff()-Funktion 128
- subpartition by-Schlüsselwort 293
- sum-Funktion 274
- Surrogatschlüssel 6
- Syntax
 - ANSI-Join 91
 - Case-Ausdrücke 200
- Systemkatalog 250

T

- Tabelle *siehe auch* Datenbanken 4, 7, 211
 - abgeleitet 53
 - Aliase 98
 - Aliase, definieren 57
 - ändern 38
 - anlegen 27, 33
 - befüllen 33, 39
 - Constraints 232
 - Cross Joins 190, 196
 - Daten erzeugen 175
 - Entwurf 27
 - ER-Diagramm 317
 - from-Klausel 53, 57
 - group by-Klausel 60
 - having-Klausel 60
 - Indizes 221, 231
 - löschen 38
 - mehrere

- Joins 87, 93
 - Joins mit drei und mehr Tabelle *siehe auch* Datenbanken 93
 - Self Joins 98
 - union-Operatoren 105
- Mengenoperation, Richtlinien 104
- modifizieren 33, 39
- Natural Joins 196, 197
- optimieren 28
- order by-Klausel 61, 65
- Outer Joins 185, 190
 - Bedingungslogik 199
 - Drei-Wege- 189
 - Left/Right 188
- partitionieren 287
- Sperren 212
- temporär 54
- Unterabfragen 53, 159
 - anwenden 95, 173, 182
 - korrelierte 169, 173
 - nicht-korrelierte 161, 168
 - Typen 160
- Verknüpfungen 56
- where-Klausel 58, 60
- zusammengestellte Daten 241

Tabellenaliase 57

Task-orientierte Unterabfragen 177

Tastaturen, Sonderzeichen 119

temporale Daten

- anwenden 133, 135
- bearbeiten 139, 143
- erzeugen 135, 139

temporale Datentypen 25, 27

temporary-Schlüsselwort 54

Terminologie für Datenbanken 7

Text

- Größe ändern 23
- Indizes 228
- Typen 23
- Textdatentyp 116
- Toad Data Point 301
- Toolkits, Integration 10
- Tools
 - mysql-Kommandozeilentool 11
 - anwenden 19
 - create table-Anweisung 31
 - Oracle Text 229
- Transact-SQL-Sprache 10, 259
- Transaktionen 213, 220
 - beenden 216
 - Mehrbenutzerdatenbanken 211
 - Savepoints 218
 - starten 215
 - Übungen zu 220
- Transformationen, Ergebnismengen 203
- treibende Tabelle 95
- Trends, gruppieren 145, 148
- truncate()-Funktion 132
- Typen
 - Bedingungen 70, 81
 - Constraints 232
 - Indizes 227
 - Intervalle 140
 - Text 23
 - Unterabfragen 160
 - korrelierte 169, 173
 - nicht-korrelierte 161, 168

U

- übergeordnete Abfrage referenzieren 54
- übergeordnete Anweisungen 159
- überlappende Daten 105, 108

- Übungsaufgaben, Lösungen 319
- Uhrzeiten *siehe auch* temporale Datentypen 25
 - Konfiguration von Zeitzonen 134
- Ungleichheitsbedingungen 71
- union all-Operator 105, 108
 - partitionierte Daten in Views bereitstellen 243
- union-Operator 105, 108
- unique-Constraints 232
- unique-Indizes 225
- Universal Time 134
- Unterabfragen
 - als Ausdrucksgeneratoren 180
 - als Datenquellen 173
 - anwenden 78, 173, 182
 - benannte in with-Klausel 179
 - einzelne Spalte 162
 - gruppieren 177
 - in Anweisungen für aktualisierbare Views 243
 - in having-Klausel 166
 - in order by-Klausel 181
 - mehrere Spalten 167
 - mehrere Zeilen 162
 - skalare 161
- Tabellen, anwenden als 95
- Task-orientierte 177
- Typen 160
 - korrelierte 169, 173
 - nicht-korrelierte 161, 168
- Übersicht 159
- unterabfragegenerierte Tabellen 53
- unzulässige Datumskonvertierungen 40
- Updates
 - Bedingungen 207
 - Tabellendaten 38
 - Views 243

- einfache Views aktualisieren 244
- komplexe Views aktualisieren 245
- UTC (Coordinated Universal Time) 134

V

- varchar-Datentyp 115
- Vereinigungsmengen 101
- Vergleichsoperatoren
 - in Filterbedingungen 70
- Verkettungsoperator (+) 127
- Verknüpfungen, Tabellen 56
- verschiedene Werte, zählen 150
- Versionierung 212
- vertikale Partitionierung 287
- verwalten von Datenbanken 2
- Views 55, 237
 - abfragen 238
 - aktualisierbare 243
 - Definition 237
 - erstellen mit create view-Anweisung 237
 - Gründe für die Verwendung 240
 - Datensicherheit 240
 - Datenzusammenstellung 241
 - Komplexität verbergen 242
 - partitionierte Daten verbinden 243
 - in information_schema 250
 - Informationen abrufen 252
 - information_schema-Views in MySQL 8.0 abrufen 254
 - Übungen zu 247
 - untersuchen mit describe-Befehl 238
- Volltextindizes 229
- Vorhandensein, prüfen 204
- vorzeichenbehaftete Daten 133

W

- Werte

- insert-Anweisungen 36
- konvertieren 143
- null
 - Aggregatfunktionen 151
 - Case-Ausdrücke 208
 - filtern 82, 85
- Spalten, Problembehebung 40
- verschiedene, zählen 150
- vorzeichenbehaftete Daten 133
- where-Klausel 58, 60
 - ANSI-Join-Syntax in 92
 - Bedingungen, auswerten 67, 70
 - Filterbedingungen in 67
 - gruppierende Filterbedingungen 157
- Wildcards, anwenden 79
- with-Klausel 179

X

- XML
 - Ausgabe generieren für Abfragen 37
 - in Dokumentendatenbanken 298

Y

- YARN 298

Z

- Zahlen
 - Aggregatfunktionen 152
 - Funktionen, zurückgeben 142
 - Genauigkeit, steuern 131
 - runden 131
 - String-Funktionen, Rückgabe von 121
- zählen verschiedener Werte 150
- Zeichen 20
 - Datentypen
 - anwenden in Strings 115, 128

- CHAR 115
- CLOB 116
- Text 116
- varchar 115
- Sonderzeichen, formatieren 119
- Wildcards 79
- Zeilen 4, 7
 - Aggregatfunktionen 148, 152
 - Duplikate, löschen 51
 - Outer Joins 185, 190
 - Sperrungen 212
 - Unterabfragen, mehrere Zeilen 162
- Zeitzone 134
- Zugriff, Transaktionen 213, 220
- zurückgeben
 - Datumsangaben 139
 - Strings 141
 - Zahlen 142
- zusammengesetzte Abfragen 105
 - Mengenoperationsregeln 111, 113
- zusammengesetzte Schlüssel 4
- Zweignoten 227