

doch nur dann benutzen, wenn MySQL mit SSL-Unterstützung läuft (siehe auch Kapitel 5.1.6).

»DES« und die Erweiterung »3-DES« (Triple-DES) waren lange Zeit der Standard in der Datenverschlüsselung. Da sie nicht mehr dem aktuellen Standard entsprechen, möchten wir hier nicht weiter darauf eingehen.

Die »AES«-Funktionen (Advanced Encryption Standard) stehen seit Version 4.0.2 zur Verfügung und sind im Unterschied zu den »DES«-Funktionen nicht von anderen Dateien, wie z.B. der Schlüsseldatei, abhängig. »AES« (auch bekannt unter seinem ursprünglichen Namen »Rijndael«) wurde in einem mehrjährigen Prozess als Nachfolger von »DES« auserwählt. Sie haben damit die bestmögliche Verschlüsselung Ihrer Daten in MySQL – vorausgesetzt, Ihr Passwort ist gut gewählt!

```
mysql> INSERT INTO kunden
      SET kreditkarte=AES_ENCRYPT('123456789','passwort');
```

Die Entschlüsselung erreicht man mit:

```
mysql> SELECT AES_DECRYPT(kreditkarte,'passwort')
      FROM kunden;
```

»AES_DECRYPT« erkennt es, wenn Sie z.B. ein falsches Passwort verwenden oder versuchen Daten zu entschlüsseln, die nicht mit »AES_ENCRYPT« verschlüsselt worden sind, und liefert »NULL« zurück.

4.1.2 Daten abfragen – SELECT

In Kapitel 3 haben wir »SQL« in »DDL« (Data Definition Language/Datendefinitionssprache), »DML« (Data Manipulation Language/Datenabfragesprache) und »DCL« (Data Control Language/Datenkontrollsprache) unterteilt.

Mit den Befehlen aus der Gruppe der Data Manipulating Language ist es möglich, Abfragen aus Tabellen zu formulieren. Für die Formulierung von Abfragen steht uns die SQL-Anweisung »SELECT« zur Verfügung. In diesem Kapitel soll es zunächst um die Abfrage von Daten aus einer Tabelle gehen. In Kapitel 4.2.5 werden wir Abfragen über mehrere Tabellen behandeln. Grundlage für die folgenden Beispiele ist die Künstlerdatenbank »kuenstlerdaten«.

Die »SELECT«-Anweisung hat die Aufgabe, Daten aus einer oder mehreren einzelnen oder verbundenen Tabellen auszulesen. Sie hat eine sehr komplexe

Struktur. Es gibt eine Anzahl von Klauseln, mit der man eine Abfrage sehr differenziert einschränken kann:

Einschränkende
Anweisungen

- FROM-Klausel – gibt an welche Tabellen einbezogen werden.
- WHERE-Klausel – gibt an welche Bedingungen die Daten erfüllen müssen.
- GROUP BY-Klausel – gibt an, wie das Ergebnis gruppiert wird.
- HAVING-Klausel – gibt die sekundäre Bedingung an, die die Zeilen erfüllen müssen.
- ORDER BY-Klausel – gibt an, wie das Ergebnis sortiert werden soll.
- LIMIT – begrenzt das Ergebnis auf eine bestimmte Anzahl von Datensätzen

Diese Reihenfolge entspricht auch der Reihenfolge, in der die Klauseln abgearbeitet werden, sofern sie verwendet werden. Jede Klausel erzeugt eine temporäre Arbeitstabelle, auf der wiederum die als nächstes folgende Klausel operiert. Das Ergebnis einer »SELECT«-Anweisung ist also immer eine Ergebnistabelle, die die Einschränkungen der als letztes spezifizierten Klausel erfüllt (siehe auch Abbildung 4.2).

Die wichtigsten Operationen, die man auf eine Tabelle anwenden kann, sind:

- Projektion
- Selektion
- kartesisches Produkt (siehe Abfragen über mehrere Tabellen)
- Verbund (siehe Abfragen über mehrere Tabellen)

Projektion

Auswahl aller Spalten einer Tabelle

Unter einer Projektion versteht man die Auswahl von bestimmten oder allen Spalten (vollständige Projektion) einer Tabelle. Sollen alle Spalten einer Tabelle in der »SELECT«-Anweisung ausgewählt werden, wird dafür der Platzhalter »*« verwendet. Die Reihenfolge der angezeigten Tabellenspalten entspricht der in der »CREATE TABLE«-Anweisung definierten Reihenfolge. Die allgemeine Syntax lautet:

```
SELECT * FROM Tabellenname;
```

Sollen nur bestimmte Spalten ausgewählt werden, was man als eingeschränkte Projektion bezeichnet, werden die Spaltennamen explizit aufgeführt.

```
SELECT Spaltenname1, ..., Spaltenname n  
FROM Tabellenname;
```

Wenn wir uns auf unsere Beispieltabelle »kuenstlerdaten« beziehen, bedeutet das z.B. für die Auswahl aller Tabellenspalten:

```
mysql> SELECT *
        FROM kuenstlerdaten;
```

Für die eingeschränkte Projektion lautet die Anweisung:

```
mysql> SELECT name, vorname
        FROM kuenstlerdaten;
```

Für die Anzeige ist es oft sinnvoll (z.B. aus Gründen der Lesbarkeit), die Spaltennamen für die Anzeige anders zu bezeichnen. Zu diesem Zweck kann man einen so genannten Aliasnamen (lt. *alias*, dt. *auch genannt*) definieren. Dieser Name gibt einer Spalte einen neuen Namen, der auf die ursprüngliche Bezeichnung verweist. Dafür verwendet man die »AS«-Klausel:

```
SELECT Spaltenname1 AS Aliasname1, ...,
       Spaltenname n AS Aliasname n
FROM Tabellename;
```

```
mysql> SELECT name AS Kuenstlername, vorname AS Kuenstlervorname
        FROM kuenstlerdaten;
```

Anmerkung: Wenn Sie einen Spaltenbezeichner für eine Spalte definieren, sollte man in der weiteren Verwendung, z.B. in der »ORDER BY«-Klausel, mit dem ursprünglichen Spaltennamen operieren, sonst erhalten Sie unerwartete Ergebnisse. Ebenfalls relevant ist das, wenn Sie eine Abfrage über mehrere Tabellen durchführen. Werden SQL-Schlüsselwörter als Spaltenbezeichner verwendet, wie z.B. *Alter*, so müssen Sie diesen Bezeichner in doppelte oder einfache Hochkommata setzen (*AS "Alter"* bzw. *AS 'Alter'*).

Spaltenbezeichner
in Hochkommata

In machen Fällen ist es wünschenswert, mehrere Spalten einer Tabelle in der Ausgabe zusammenzuführen. Hierfür steht uns die SQL-Anweisung »CONCAT« zur Verfügung. In unserem Beispiel wollen wir die Spalten »Name und »Vorname« zusammenführen und diese durch ein Komma trennen.

```
mysql> SELECT CONCAT(vorname, ', ', name)
        FROM kuenstlerdaten;
```

```

+-----+
| CONCAT(vorname, ' ', name) |
+-----+
| Loreena, McKennit          |
| Serge, Gainsbourg         |
| Joe, Cocker                |
| Kim, Smith                 |
| William Michael, Broad    |
| Gordon, Sumner             |
| Adam, Green                |
| Martin L., Gore           |
| Karl, Bartos               |
| Anni-Fried, Lyngstad      |
| Franz Xaver, Ohnesorg     |
+-----+

```

Hier haben wir noch einen kleinen Schönheitsfehler. Als Spaltenüberschrift erhalten wir in der Ausgabe »CONCAT(vorname, ' ', name)«, diese können wir verändern, indem wir ein Alias mit der Bezeichnung »Vorname, Name« definieren:

```
mysql> SELECT CONCAT(vorname, ' ', name) AS "Vorname, Name"
      FROM kuenstlerdaten;
```

Selektion

Im Gegensatz zur Projektion werden bei der Selektion nur bestimmte Datensätze einer Tabelle ausgewählt. Diese Datensätze müssen eine definierte Bedingung erfüllen. Die einfachste Möglichkeit, eine Auswahl ohne doppelt vorkommende Werte durchzuführen, bietet die »DISTINCT«-Klausel (dt. *verschieden*). Mit dieser Klausel ist es möglich alle mehrfach vorkommenden Werte einer Spalte nur einfach anzuzeigen. Diese Möglichkeit setzt man ein, wenn man wissen möchte, welche verschiedenen Werte sich in einer Tabellenspalte befinden.

```
SELECT DISTINCT Spaltenname
      FROM Tabellenname;
```

Aus unserer Beispieltabelle möchten wir die Namen der verschiedenen Länder auflisten, aus denen die Künstler kommen. Hierbei werden doppelt vorkommende Ländernamen eliminiert:

```
mysql> SELECT DISTINCT land
      FROM kuenstlerdaten;
```

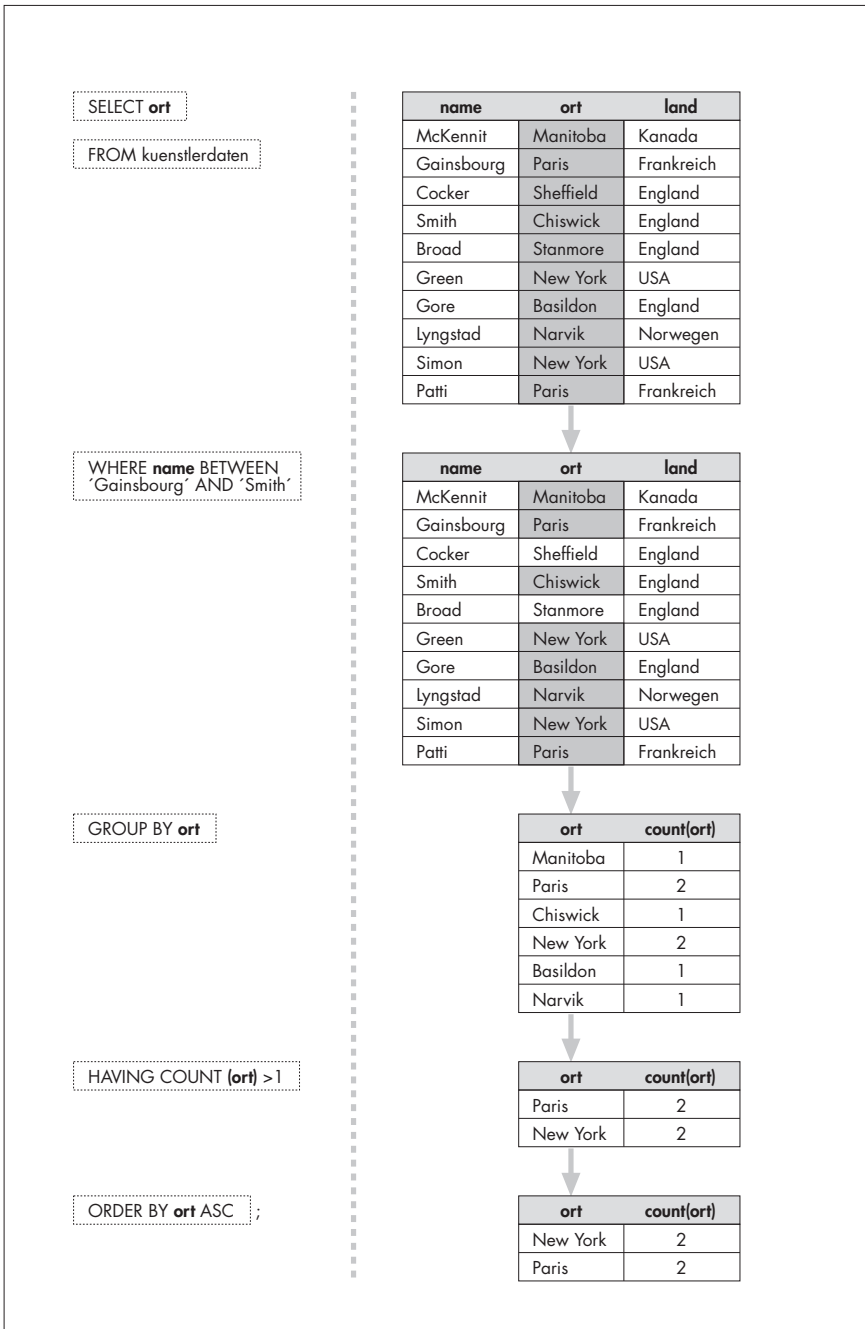


Abb. 4.2 Wirkungsweise von Klauseln in der »SELECT«-Anweisung auf die Ergebnistabelle

Eine andere Möglichkeit der Selektion ist der Einsatz der »WHERE«-Klausel, die wir im Kapitel 4.2.1 besprechen werden. Hiermit können sehr differenzierte Einschränkungen definiert werden.

4.1.2.1 COUNT()-Funktion

Die »COUNT()« Funktion bezeichnet man auch als Aggregat-Funktion. Wenn Sie der »COUNT(*)«-Funktion den »Asterisk« – als Asterisk bezeichnet man das »*«-Zeichen – übergeben, zählt sie alle Datensätze in der Ergebnistabelle, sowohl doppelt vorhandene als auch NULL-Werte. In der Tabelle »kuenstlerdaten« soll die Anzahl der eingetragenen Künstler ermittelt werden. Hierzu schreibt man:

```
mysql> SELECT COUNT(*) AS Anzahl
        FROM kuenstlerdaten;
```

```
Ergebnis : 11
```

Das oben gezeigte Beispiel hat alle Datensätze gezählt. In der Tabelle »alben« haben wir in der Spalte »kuenstler« doppelte Einträge. Wir möchten nun die Anzahl der Künstler zählen, dafür müssen wir die doppelten Einträge ignorieren. Dieses erreichen wir mit der »DISTINCT«-Klausel.

```
mysql> SELECT COUNT(DISTINCT kuenstler)
        FROM alben;
```

Wir erhalten als Ergebnis 11. Im Gegensatz dazu zählt die Klausel »ALL« alle Datensätze. Diese Klausel ist standardmäßig eingestellt und braucht nicht explizit aufgeführt werden.

4.1.2.2 GROUP BY-Klausel

Aufgabe der »GROUP BY«-Klausel ist es, die Ergebnistabelle in Gruppen zusammenzufassen. Werden nun in einer Spalte gleiche Werte gefunden, beispielsweise indem sie mit der »COUNT()«-Funktion gezählt werden, so werden diese der durch »GROUP BY« angegebenen Gruppe zugeordnet. Man bezeichnet die Ergebnistabelle auch als gruppierte Tabelle. Diese Klausel wird vorwiegend zu

analytischen Zwecken verwendet. Wir können beispielsweise abfragen, welche Band in unserer Tabelle »alben« wie viele Alben veröffentlicht hat. Die Bandnamen können wir uns dann mit folgender SQL-Anweisung anzeigen lassen:

```
mysql> SELECT id, kuenstler AS Kuenstler
        FROM alben;
```

id	Kuenstler
1	a-ha
2	Enigma 3
3	Billy Idol
4	Midnight Oil
5	Goldfrapp
6	Mylene Farmer
7	Depeche Mode
8	Depeche Mode
9	Propaganda
10	Regenerator
11	Adam Green
12	Moby

In diesem Fall zeigen wir alle Datensätze an. Doppelte Datensätze können wir nun mit Hilfe der »GROUP BY«-Klausel eliminieren:

```
mysql> SELECT id, kuenstler AS Kuenstler
        FROM alben
        GROUP BY kuenstler;
```

id	Kuenstler
1	a-ha
11	Adam Green
3	Billy Idol
7	Depeche Mode
2	Enigma 3
5	Goldfrapp
4	Midnight Oil
12	Moby
6	Mylene Farmer
9	Propaganda
10	Regenerator

Gleichzeitig wird das Ergebnis auch alphabetisch sortiert. Aber erst wenn wir die »GROUP BY«-Klausel in Verbindung mit der »COUNT()«-Funktion verwenden, wird das Potential Klausel deutlich. Das folgende Beispiel zählt die Anzahl der Veröffentlichungen von jeder Band:

```
mysql> SELECT kuenstler, COUNT(kuenstler) AS Anzahl,
FROM alben
GROUP BY kuenstler;
```

kuenstler	Anzahl
a-ha	1
Adam Green	1
Billy Idol	1
Depeche Mode	2
Enigma 3	1
Goldfrapp	1
Midnight Oil	1
Moby	1
Mylene Farmer	1
Propaganda	1
Regenerator	1

Bisher haben wir immer nur die unterschiedlichen Werte einer Spalte gezählt, genauso können natürlich auch die Werte mehrerer Spalten gezählt werden. Beispielsweise kann die Anzahl der Veröffentlichungen jeder Band und die Anzahl der Labels gezählt werden. Folgendes Beispiel zeigt dies:

```
mysql> SELECT kuenstler,COUNT(kuenstler) AS Anzahl,
COUNT(label) AS Labelanzahl
FROM alben
GROUP BY kuenstler,label;
```

kuenstler	Anzahl	Labelanzahl
a-ha	1	1
Adam Green	1	1
Billy Idol	1	1
Depeche Mode	2	2
Enigma 3	1	1
Goldfrapp	1	1
Midnight Oil	1	1
Moby	1	1
Mylene Farmer	1	1
Propaganda	1	1
Regenerator	1	1

4.1.2.3 HAVING-Klausel

Bisher steht uns noch keine Möglichkeit zur Verfügung, die durch eine Gruppierung erhaltenen Datensätze einzuschränken. Hierfür steht uns die »HAVING«-Klausel zur Verfügung. Sie schränkt nur mit Hilfe von »GROUP BY« gebildete Gruppen ein. Man bezeichnet diesen Vorgang auch als Gruppen-

restriktion. Möchten wir, aus unserem Beispiel des vorherigen Abschnitt nur Bands anzeigen lassen, die mehr als ein Album veröffentlicht haben, müssen wir folgende Anweisung absetzen:

```
mysql> SELECT kuenstler,COUNT(kuenstler) AS Anzahl,
COUNT(label) AS Labelanzahl
FROM alben
GROUP BY kuenstler,label
HAVING COUNT(kuenstler)>1;
```

```
+-----+-----+-----+
| kuenstler | Anzahl | Labelanzahl |
+-----+-----+-----+
| Depeche Mode | 2 | 2 |
+-----+-----+-----+
```

4.1.3 Daten ändern – UPDATE

Manchmal ist es notwendig, einzelne Datensätze zu ändern oder zu aktualisieren. Der SQL-Befehl hierfür heißt »UPDATE«. Zu beachten ist hierbei, dass der zu ändernde Datensatz oder die zu ändernden Datensätze mit eindeutigen Kriterien ausgewählt werden, um keine unerwünschten Änderungen zu verursachen. Es gibt zwei Möglichkeiten, eine Änderung durchzuführen, die Methode der absoluten Änderung und die der relativen Änderung. Eine absolute Änderung wird wie folgt durchgeführt:

```
UPDATE Tabellenname
SET Spaltenname1 = neuer Wert 1,
    Spaltenname2 = Wert 2, ...,
    Spaltenname n = Wert n
WHERE Auswahlkriterium;
```

Wir müssen also den Tabellennamen angeben, die Spalte bzw. die durch Kommata getrennten Spalten, denen ein neuer Eintrag zugeordnet wurde, und ein eindeutiges Auswahlkriterium, welches wir mit der »WHERE«-Klausel formulieren. Ist das Auswahlkriterium nicht eindeutig, so werden mehrere Datensätze geändert. Die Auswahl kann zusätzlich mit »LIMIT« auf einen Datensatz beschränkt werden, falls man sicher gehen möchte, dass nur ein Datensatz geändert wird.

```
mysql> UPDATE kuenstlerdaten
      SET name = 'Mac Ken'
      WHERE id = 1 LIMIT 1;
```

relative Änderung

Bei einer relativen Änderung wird eine Berechnungsformel eingesetzt und der neue Wert relativ zum alten Wert geändert.

```
UPDATE Tabellenname
      SET Spaltenname 1 = mathematischer Ausdruck
      WHERE Auswahlkriterium;
```

In unserem Beispiel würde das, angewendet auf die Preisspalte in der Tabelle »alben«, wie folgt aussehen:

```
mysql> UPDATE alben
      SET preis = preis*1.2
      WHERE preis < 14;
```

Es werden also die Preise aller Datensätze um 20% erhöht, deren Wert unter 14 liegt. Die Bedingung »preis < 14« stellt sicher, dass nur die Datensätze bearbeitet werden, die einen Preis kleiner als 14 enthalten.

4.1.4 Daten löschen – DELETE

Auch die Löschung von Datensätzen aus einer Tabelle ist möglich. Man sollte sich jedoch genau überlegen, ob und wann ein Datensatz gelöscht wird, weil dieser Vorgang nicht rückgängig gemacht werden kann. Man verwendet hierfür den SQL-Befehl »DELETE«.

```
DELETE [LOW_PRIORITY] [QUICK] FROM Tabellenname
      [ORDER BY Sortierkriterium]
      [WHERE Auswahlkriterium];
```

Ist die Option »LOW_PRIORITY« gesetzt, wird die »DELETE«-Anweisung erst ausgeführt, wenn die Tabelle von keinem Client mehr gelesen wird. Die Option »QUICK« kann nur in Verbindung mit dem Tabellentyp »MyISAM« verwendet werden und beschleunigt dort die Ausführung des Befehls, weil der Tabellenindex nicht reorganisiert wird.

»ORDER BY« ermöglicht eine Sortierung der zu löschenden Datensätze. Die Datensätze werden dann in der Reihenfolge der Sortierung gelöscht.

QUICK-Option nur
bei MyISAM-
Tabellen

Mit der »FROM«-Klausel wird angegeben, aus welcher Tabelle der Datensatz gelöscht werden soll. Die »WHERE«-Klausel schließlich spezifiziert, welche Datensätze gelöscht werden sollen. Wird sie weggelassen, dann werden alle Datensätze aus der Tabelle gelöscht. Die Anweisung

```
DELETE FROM Tabellenname;
```

löscht demnach alle Datensätze einer Tabelle. Seien Sie also damit sehr vorsichtig. Die »WHERE«-Klausel schränkt die zu löschenden Datensätze ein. Hierbei wird eine Bedingung formuliert, die auf diese Datensätze zutrifft. Eine Hilfe ist es manchmal, mit einer »SELECT«-Anweisung zuerst zu prüfen, welche Datensätze von der Bedingung in der »WHERE«-Klausel erfasst werden, um sicherzustellen, dass diese richtig formuliert wurde. Möchten wir aus unserer Tabelle »kuenstlerdaten« den Datensatz löschen, der die »id« »3« besitzt, so müssen wir schreiben:

```
mysql> DELETE FROM kuenstlerdaten  
      WHERE id = 3;
```

Es ist auch möglich, eine komplette Tabelle mit einer anderen Anweisung zu löschen. Der Befehl hierfür lautet »TRUNCATE«:

```
TRUNCATE Tabellenname ;
```

Der Befehl »TRUNCATE« löscht nicht einzelne Zeilen, sondern erstellt die Tabelle aufgrund der Beschreibungen in der ».frm«-Datei neu. D.h., alle Datensätze werden entfernt und die Tabelle hat dieselben Eigenschaften wie vor der Eintragung des ersten Datensatzes. Dadurch ist dieser Befehl extrem schnell. Ein weiterer Vorteil ist, dass die Autoincrement-Werte auf null zurückgesetzt werden. Im Gegensatz dazu wird bei einem »DELETE-Befehl die laufende Nummer des gelöschten Datensatzes, die durch eine Autoincrement-Spalte erzeugt wurde, nicht wieder verwendet.

4.2 Komplexe Abfragen von Datenbanken

Bisher haben wir einfache »SELECT«-Abfragen auf eine Tabelle abgesetzt. Das Ergebnis bestand meistens in der Ausgabe aller Datensätze. Sinnvoll wäre es, nur diejenigen Datensätze auszuwählen, die bestimmten Kriterien genügen. Man bezeichnet das auch als Vergleich oder Suchbedingungen (engl. *search conditions*). Ein Vergleich kann sowohl über eine Tabelle als auch über mehre-