SQL2009

Autoren: Tanja Bossert, Ulrike Böttcher, Peter Teich

Inhaltliches Lektorat: Ricardo Hernández Garcia

1. Ausgabe, 1. Aktualisierung März 2010

Internet: www.herdt.com

SQL

Grundlagen und Datenbankdesign

Neubearbeitung 2009

SQL2009

6 Tabellen erstellen und verwalten

In diesem Kapitel erfahren Sie

- wie Sie neue Tabellen erstellen
- welche Datentypen es gibt
- wie Sie vorhandene Tabellen anzeigen, ändern und löschen
- wie Sie Domänen erstellen und verwenden
- wie Sie Constraints in Tabellen verwenden

Voraussetzungen

- ✔ Datenbanken erstellen

6.1 Tabellen erstellen

Tabellen (Relationen) sind die einzigen Objekte einer Datenbank, in denen die Daten gespeichert werden. Jeder Zugriff auf die Daten erfolgt über die Tabellen. Bei jeder Abfrage oder Auswertung muss der Name der Tabelle angegeben werden. Die Datenbank dient dabei als Container für die logisch zusammengehörigen Tabellen. Der Tabellenname muss innerhalb einer Datenbank eindeutig sein, kann aber in mehreren Datenbanken eines DBS verwendet werden.

Eine Tabelle besteht aus einzelnen Feldern (Datenfeldern, Attributen, Spalten). Durch die Namen und Datentypen der Felder wird die Struktur bzw. das Schema der Tabelle festgelegt. Zusammengehörige Daten werden in eine Zeile der Tabelle eingetragen. Sie bilden einen Datensatz bzw. ein Tupel.

Um eine Tabelle erstellen zu können, benötigen Sie das Ausführungsrecht für die Anweisung `CREATE TABLE`. Der Administrator vergibt dieses Recht mit der Anweisung `GRANT CREATE TABLE`. Der Benutzer, der die Tabelle erstellt, wird danach zu ihrem Eigentümer.

Einfache Tabellen erstellen

Eine Tabelle mit den dazugehörigen Datenfeldern erstellen Sie mit der Anweisung `CREATE TABLE`. Die Anweisung besitzt sehr viele zusätzliche Optionen, die z. B. das Erstellen von Indizes oder das Definieren von Standardwerten erlauben.

Beispiel: *Mitarbeiter.sql*

Im Folgenden wird eine Tabelle mit vier Feldern zum Speichern von Mitarbeiterdaten erzeugt.

```
① CREATE TABLE t_ma
② (id INTEGER NOT NULL,
③  vname VARCHAR(100),
   name VARCHAR(100),
④  adr TEXT);
```

① Die neue Tabelle erhält den Namen t_ma. In Klammern folgt die Definition der Datenfelder.

② Das erste Datenfeld heißt *id* und ist vom Typ `INTEGER`. Es kann ganze Zahlen mit einer Länge von vier Bytes speichern. Da anhand der Einträge im Feld *id* die einzelnen Datensätze identifiziert werden sollen, wird zusätzlich festgelegt, dass das Feld immer einen Wert enthalten muss (`NOT NULL`).

③ Die Felder *vname* und *name* werden mit dem Datentyp `VARCHAR` und einer maximalen Länge von 100 Zeichen definiert. Bei diesem Datentyp werden nur so viele Zeichen in der Tabelle gespeichert wie tatsächlich benötigt.

④ Hier wird ein Datenfeld vom Typ `TEXT` angelegt. Es ermöglicht das Speichern von größeren Textmengen mit variabler Länge.

- ✔ Damit Sie einen Tabellennamen besser von anderen Datenbankobjekten (z. B. Abfragen oder Stored Procedures) unterscheiden können, empfiehlt es sich, ein Präfix als Kennzeichen voranzustellen, z. B. *t_*.
- ✔ In diesem Buch werden kurze Namen für Datenbankobjekte verwendet, um den Schreibaufwand möglichst gering zu halten. Wie lang der Name eines Datenbankobjektes sein kann, ist vom jeweiligen DBS abhängig.

```
mysql> CREATE TABLE t_ma
    -> (id INTEGER NOT NULL,
    -> vname VARCHAR (100),
    -> adr TEXT);
Query OK, 0 rows affected (0.12 sec)

mysql>
```

Erstellen einer neuen Tabelle mit MySQL

Syntax

```
CREATE TABLE tabellenname
   (datenfeld1 datentyp1 [DEFAULT standardwert1 | NULL | NOT NULL] [AUTO_INCREMENT],
    ...
    datenfeldX datentypX [DEFAULT standardwertX | NULL | NOT NULL] [AUTO_INCREMENT],
    PRIMARY KEY(datenfeldname));
```

- ✔ Mit der Anweisung `CREATE TABLE` wird eine neue, leere Tabelle in der aktiven Datenbank erstellt. Danach folgt der gewünschte Tabellenname.
- ✔ In runden Klammern folgen die Definitionen der einzelnen Datenfelder. Für jedes Datenfeld müssen dabei ein Name und ein Datentyp angegeben werden.
- ✔ Mit der Angabe `PRIMARY KEY` kann ein Datenfeld als Primärschlüssel festgelegt werden. Der Primärschlüssel ermöglicht es, einen Datensatz eindeutig zu identifizieren.

Folgende optionale Parameter sind bei der Definition von Datenfeldern möglich:

NOT NULL	Mit diesem Parameter wird die Eingabe eines Wertes für das entsprechende Datenfeld erzwungen. Die Angabe `NOT NULL` ist für Schlüsselfelder unbedingt anzugeben.
NULL	Mit diesem Parameter wird festgelegt, dass das Datenfeld standardmäßig keinen Wert (auch nicht 0 oder eine leere Zeichenkette) enthält.
DEFAULT standardwert	Der Parameter `DEFAULT` definiert einen Standardwert für das Datenfeld. Erhält dieses Datenfeld bei der Eingabe der Daten keinen Wert, wird der Standardwert verwendet.
AUTO_INCREMENT (MySQL) **SERIAL** (PostgreSQL)	Der Wert dieses Datenfeldes wird automatisch beim Anlegen eines neuen Datensatzes aus dem Wert des Datenfeldes des vorherigen Datensatzes plus eins errechnet. Dieser Wert kann vom Benutzer nicht geändert werden. Diese Einstellung ist besonders für den Primärschlüssel empfehlenswert, da dadurch automatisch ein eindeutiger Schlüsselwert erzeugt wird.

- ✔ In verschiedenen Datenbanksystemen sind nicht immer alle angegebenen Möglichkeiten zur Definition einer Tabellenstruktur vorhanden. Es ist auch möglich, dass diese einen anderen Namen für die betreffende Einstellung verwenden.

Beispiele für die Definition von Datenfeldern

Definition in der CREATE-TABLE-Anweisung	Erklärung
`plz_ort` **VARCHAR**(100)	Das Datenfeld *plz_ort* wird mit dem Datentyp `VARCHAR` definiert und ist maximal 100 Zeichen lang.

`id` **`INTEGER NOT NULL`**	Für das Datenfeld *id* wird der Integer-Datentyp zum Speichern von ganzen Zahlen festgelegt. Die Angabe `NOT NULL` erzwingt die Eingabe eines Wertes beim Anlegen eines neuen Datensatzes.
`anzahl` **`DEFAULT`** `1`	Für das Datenfeld *anzahl* wird ein Standardwert `1` definiert.
`ort` **`DEFAULT`** `"Berlin"`	Im Datenfeld *ort* wird der Standardwert *Berlin* verwendet.

Der Wert `NULL` bedeutet, dass ein Datenfeld keinen Inhalt besitzt. Er ist nicht mit der Zahl 0 und je nach Datenbanksystem auch nicht mit einer leeren Zeichenkette identisch. Null-Werte können entstehen, wenn beim Einfügen von Daten in eine Tabelle für bestimmte Spalten keine Werte angegeben werden. In einer Abfrage kann auf den Wert `NULL` geprüft werden.

Wenn Sie eine Tabelle erstellen, müssen Sie einen eindeutigen Tabellennamen angeben. Zusätzlich ist es notwendig, mindestens ein Datenfeld zu definieren.

6.2 Datentypen festlegen

Beim Erstellen einer Tabelle muss für jedes Datenfeld ein Datentyp angegeben werden. Der verwendete Datentyp entscheidet, ob in einem Feld Zahlen, Zeichenketten oder andere Daten gespeichert werden. SQL bietet eine große Anzahl an Datentypen, im Folgenden werden einige typische Datentypen beispielhaft vorgestellt. Einige der vorgestellten Datentypen werden nicht von allen DBMS unterstützt.

Numerische Datentypen für ganzzahlige Werte

In Feldern dieses Datentyps werden ganzzahlige Werte (Integer-Zahlen) ohne Kommastellen gespeichert.

Datentyp	Speicherbedarf	Wertebereich
`SMALLINT`	2 Byte	-32768 bis +32767
`INTEGER`	4 Byte	-2147483648 bis +2147483647
`BIGINT`	8 Byte	-9223372036854775808 +9223372036854775808

Integer-Datentypen eignen sich besonders für eindeutige Identifikationsnummern, z. B. Primärschlüssel einer Tabelle. Der Zugriff auf solche Datenfelder erfolgt besonders schnell, da alle Prozessoren in modernen Computern für diese Zahlenwerte optimiert sind.

Numerische Datentypen für Fließkommazahlen

Mit diesen Datentypen können Sie Zahlenwerte mit Nachkommastellen speichern. Die Datentypen unterscheiden sich dabei im Wertebereich.

Datentyp	Speicherbedarf	Wertebereich
`FLOAT (n)`	4 oder 8 Byte	Die Gesamtstellenanzahl kann durch n festgelegt werden. Je nach Angabe von n können 7 bis 15 signifikante Stellen plattformunabhängig gespeichert werden.
`REAL`	4 Byte	7 signifikante Stellen (keine plattformunabhängige Speicherung)
`DOUBLE PRECISION`	8 Byte	15 signifikante Stellen (keine plattformunabhängige Speicherung)

Die Datentypen können einen Zahlenwert mit der angegebenen Anzahl an Gleitkommastellen speichern. Alle Ziffern nach der Anzahl signifikanter Stellen werden abgeschnitten, daher wird bei diesen Datentypen nicht immer die genaue Größe gespeichert, sondern die näherungsweise Größe.

Numerische Datentypen für Festkommazahlen

Diese Datentypen eignen sich für das Speichern formatierter Zahlen mit einer festen Anzahl von Nachkommastellen. Damit können z. B. Währungsangaben oder Messwerte gespeichert werden.

Datentyp	Erklärung
NUMERIC(Präzision, Skalierung) **DECIMAL(Präzision, Skalierung)**	Der Parameter `Präzision` (1-15) legt die Gesamtzahl der signifikanten Stellen der Zahl fest. Der Parameter `Skalierung` (1-15) bestimmt die Anzahl der Nachkommastellen, die kleiner oder gleich der Gesamtzahl der Stellen sein muss. Der Unterschied zwischen `NUMERIC` und `DECIMAL` liegt darin, dass `NUMERIC` die exakte Anzahl und `DECIMAL` die minimale Anzahl der signifikanten Stellen angibt.
Beispiel: `DECIMAL(8,3)` speichert Zahlen zwischen -99999,999 und 99999,999	

Datentypen für Datums- und Zeitwerte

Für Datums- und Zeitwerte gibt es spezielle Datentypen. Abhängig vom Datenbank-Server werden unterschiedliche Wertebereiche zugelassen.

Datentyp	Speicherbedarf	Wertebereich
DATE	4 Byte	unterschiedlich: 01.01.1000 bis 31.12.9999 (MySQL) 4713 BC bis 5874897 AD (PostgreSQL)
TIME	3 bzw. 8 Byte	unterschiedlich: -838:59:59 bis 838:59:59 (MySQL) 00:00:00 bis 23:59:99 (PostgreSQL)

- ✔ Bei der Eingabe können Datumswerte in verschiedenen Formaten angegeben werden. Die Umwandlung in das interne Format erfolgt automatisch.
- ✔ Zeitwerte geben Sie in der Form *hh:mm:ss* an.

Eingabeformat	Beispiel
tt.mm.jj	12.07.09
tt-mmm-jj	12-JUL-09
mm-tt-jj	07-12-09

Für die Angabe des Monatsnamens können Sie folgende Syntax verwenden: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec. Die Groß- und Kleinschreibung wird dabei nicht beachtet.

Jahresangaben können zwei- oder vierstellig erfolgen. Bei zweistelliger Angabe wird automatisch das aktuelle Jahrhundert angenommen.

Es gibt auch Datentypen, die sowohl das Datum als auch die Zeit speichern. Die einzelnen Datenbanksysteme haben für diesen Fall jedoch alle ihre eigenen Datentypen entwickelt. MySQL bietet hierfür z. B. den Datentyp `DATETIME` und PostgreSQL den Datentyp `TIMESTAMP`.

Datentypen für Zeichen und Texte

SQL sieht verschiedene Datentypen vor, um Textinformationen zu speichern. Es gibt dabei Datentypen mit variabler und fester Länge.

Datentyp	Erklärung
CHAR(Länge) Länge = 1..255	Dieser Datentyp dient zum Speichern beliebiger Textinformationen. Die maximale Länge wird dabei als Parameter übergeben. Unabhängig von der tatsächlichen Länge der gespeicherten Information wird stets die bei der Definition angegebene Anzahl Zeichen gespeichert.
VARCHAR(Länge) Länge = 1..255	Zum Speichern beliebiger Textinformationen wird auch dieser Typ verwendet. Auch hier wird die maximale Länge als Parameter übergeben. In ein Datenfeld dieses Typs eingegebener Text wird in seiner tatsächlichen Länge gespeichert; in zwei zusätzlichen Bytes wird die Länge des Textes abgelegt.
BLOB (MYSQL) **BYTEA (POSTGRESQL)**	Dieser Datentyp (Binary Large Objects) wird zum Speichern großer, auch binärer Datenmengen eingesetzt, z. B. sehr großer Textdateien, Grafiken, Bilder oder Videos. Bei PostgreSQL gibt es den Datentyp BLOB nicht, dafür kann jedoch der Datentyp BYTEA verwendet werden, der vergleichbar mit BLOB ist.
TEXT	Wie auch in BLOB-Feldern können Sie hier größere Informationsmengen mit variabler Länge speichern. Der Unterschied zu BLOB-Feldern liegt in der anderen Sortierreihenfolge, die bei TEXT-Feldern unabhängig von der Groß- und Kleinschreibung ist.

Da der Datentyp VARCHAR zwei zusätzliche Bytes für die Längenspeicherung benötigt, sollten Sie ihn nicht für die Definition sehr kurzer Datenfelder einsetzen. In einigen Datenbanksystemen erfolgt der Zugriff auf die mit fester Länge gespeicherten CHAR-Datenfelder schneller als auf VARCHAR-Datenfelder. VARCHAR-Datenfelder eignen sind besonders für die Speicherung von unterschiedlich langen Texten.

Logischer Datentyp

SQL bietet auch die Möglichkeit, logische Werte zu speichern.

Datentyp	Erklärung
BOOLEAN	Logische Werte Ja/Nein bzw. True/False. NULL ist ebenfalls möglich und wird als UNKNOWN interpretiert.

Datentypumwandlung

Datentypen lassen sich ineinander umwandeln. Das ist z. B. bei Vergleichen von Werten unterschiedlichen Datentyps oder bei Abfragen erforderlich, in denen zwei Tabellen über Datenfelder unterschiedlichen Typs verknüpft werden. Es wird zwischen impliziter (automatischer) und expliziter Typumwandlung unterschieden.

Eine implizite Typumwandlung findet beispielsweise statt, wenn Sie eine Zahl vom Typ SMALLINT mit einer INTEGER-Zahl vergleichen. Die Konvertierung erfolgt immer vom "niedrigeren" zum "höheren" Datentyp, z. B. von SMALLINT zu INTEGER.

Explizit wird eine Typumwandlung über die Funktion CAST durchgeführt.

Syntax der CAST-Anweisung

```
CAST (Wert AS Datentyp)
```

- ✔ Die Funktion wandelt den angegebenen Wert in den Datentyp um, der nach dem Schlüsselwort AS aufgeführt ist.
- ✔ Es können folgende Typen ineinander umgewandelt werden:

```
NUMERIC          →   CHAR, VARCHAR, DATE
CHAR, VARCHAR    →   NUMMERIC, DATE
DATE             →   CHAR, VARCHAR, DATE
BLOB, TEXT       →   nicht möglich
```

6.3 Constraints in Tabellen verwenden

Beim Erstellen von Tabellen können Sie bereits einfache Bedingungen für die zu speichernden Daten definieren. Dazu gehört z. B., dass NULL-Werte nicht zugelassen sein sollen, dass ein Feld als Primärschlüssel dienen soll oder dass eine Gültigkeitsprüfung für einen Wertebereich eines Datenfeldes durchgeführt werden soll. Wenn diese sogenannten Constraints (engl. Zwang, Nebenbedingung) bei der Datenbearbeitung nicht erfüllt sind, wird eine Fehlermeldung ausgegeben.

Sie können Constraints entweder bei der Definition einer Spalte festlegen (Spaltenconstraint) oder nach Angabe aller Spalten als Zusatz angeben (Tabellenconstraint).

Beispiel: *Produkte.sql*

```
CREATE TABLE produkte
 (p_id NOT NULL,
  name VARCHAR(50), lagernr INTEGER, anzahl INTEGER
  PRIMARY KEY(p_id));
```

- ✔ Die Spalte *p_id* beinhaltet die Bedingung, dass NULL-Werte nicht zugelassen sind (Spaltenconstraint).
- ✔ Nach Angabe aller Spalten wird die Bedingung festgelegt, dass die Spalte *p_id* als Primärschlüssel dient (Tabellenconstraint).

In diesem Beispiel könnte der Primärschlüssel auch als Spaltenconstraint festgelegt werden.

```
CREATE TABLE produkte
 (p_id PRIMARY KEY,
  name VARCHAR(50), lagernr INTEGER, anzahl INTEGER);
```

Nähere Informationen zur Angabe der Schlüsselfelder finden Sie in Kap 9.

Integritätsregeln für Tabellen mit der CONSTRAINT-Anweisung festlegen

Möchten Sie für Ihre Tabellen eine bestimmte Gültigkeitsprüfung angeben, die die Integritätsregeln Ihrer Datenbank gewährleistet, können Sie die CONSTRAINT-Anweisung verwenden, die nach der Definition der Datenfelder angegeben wird.

Beispiel: *MitarbeiterAbteilungen.sql*

In der Tabelle *t_ma_abt* wird die Beziehung zwischen den Mitarbeitern und den Abteilungen gespeichert. Jeder Mitarbeiternummer wird eine Abteilungsnummer zugeordnet. Der Wert einer Mitarbeiternummer muss immer größer als 1 sein. Diese Bedingung wird als Gültigkeitsprüfung definiert.

```
   CREATE TABLE t_ma_abt
      (id INTEGER NOT NULL,
①      abtname VARCHAR(15) DEFAULT "Produktion",
②      ma_nr INTEGER,
③      CONSTRAINT mpruef CHECK(ma_nr > 1));
```

① An dieser Stelle wird für das Datenfeld *abtname* der Standardwert *Produktion* definiert. Wenn bei einer späteren Dateneingabe keine Angabe erfolgt, wird dieser Wert verwendet.

② Das Datenfeld *ma_nr* wird als Integer-Zahl deklariert.

③ Zusätzlich erfolgt nach der Definition der Datenfelder die Angabe einer Gültigkeitsprüfung (*mpruef*). Der eingegebene Wert für das Feld *ma_nr* wird nur akzeptiert, wenn er größer ist als 1.

```
mysql> USE mav
Database changed
mysql> CREATE TABLE t_ma_abt
    -> (id INTEGER NOT NULL,
    -> abtname VARCHAR(15) DEFAULT "Produktion",
    -> ma_nr INTEGER,
    -> CONSTRAINT mpruef CHECK(ma_nr > 1));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

Erstellen der Tabelle t_ma_abt innerhalb der Datenbank mav in MySQL

Syntax

```
CREATE TABLE tabellenname
   (datenfeld1 datentyp1 [DEFAULT standardwert1 | NULL | NOT NULL] [AUTO_INCREMENT],
    ...
    datenfeldX datentypX [DEFAULT standardwertX | NULL | NOT NULL] [AUTO_INCREMENT],
    CONSTRAINT constraintname1 CHECK (gültigkeitsprüfung),
    ...
    PRIMARY KEY(datenfeldname));
```

- ✔ Die Tabellendefinition erfolgt mit der Anweisung `CREATE TABLE`. Danach werden in runden Klammern die Datenfelder und Tabellenoptionen angegeben.
- ✔ Die Angabe einer Gültigkeitsbedingung wird mit dem Schlüsselwort `CONSTRAINT` eingeleitet. Danach wird der gewünschte Name der Bedingung angegeben. Nach dem Schlüsselwort `CHECK` folgt in runden Klammern die Gültigkeitsbedingung.

Die Gültigkeitsbedingung muss erfüllt (wahr) sein, damit der Wert in der Tabelle gespeichert wird. Die folgende Tabelle zeigt einige Möglichkeiten:

Bedingung	Erklärung
`nummer <= 100`	wahr, wenn `nummer` kleiner oder gleich 100 ist
`name NOT LIKE "%Ä%"`	wahr, wenn `name` den Buchstaben *Ä* nicht enthält
`abteilung IN ("Einkauf", "Verkauf")`	wahr, wenn `abteilung` den Wert *Einkauf* oder *Verkauf* besitzt

6.4 Domänen verwenden

Domänen definieren

Domänen (engl. Domains) sind benutzerdefinierte Datentypen, die Sie einmalig definieren und in verschiedenen Tabellen zum Definieren von Feldtypen verwenden können. Mit der Verwendung von Domänen kann die Definition von Integritätsregeln deutlich vereinfacht werden, da die Regeln dort nur einmalig und nicht bei jeder Tabellendefinition angegeben werden müssen. Im Falle einer Änderung müssen Sie nicht jede Tabelle einzeln bearbeiten, sondern nur die entsprechende Domäne verändern. Domänen sind in der gesamten Datenbank verfügbar, in der sie definiert wurden.

Der MySQL-Server unterstützt die Verwendung von Domänen nicht.

Beispiel

```
① CREATE DOMAIN d_ma_nr AS INTEGER;
② CREATE DOMAIN d_benutzer AS VARCHAR(15) DEFAULT "gast";
③ CREATE DOMAIN d_abt AS VARCHAR(15)
    CHECK(VALUE IN ("Einkauf","Verkauf","Marketing","Verwaltung","Produktion"));
```

① Die Domäne d_ma_nr wird als Integer-Datentyp definiert.

② Die Domäne *d_benutzer* wird mit dem Datentyp `VARCHAR` und 15 Zeichen Länge definiert. Zusätzlich wird ein Standardwert angegeben.

③ Auch die Domäne *d_abt* wird als `VARCHAR` festgelegt. Darüber hinaus wird eine Prüfung integriert, die sicherstellt, dass der zu speichernde Wert in der angegebenen Liste enthalten ist.

Syntax

```
CREATE DOMAIN domänenname AS datentyp [DEFAULT standardwert];
```

- ✔ Die Definition einer Domäne wird mit den Schlüsselwörtern `CREATE DOMAIN` eingeleitet. Danach folgen der Name der Domain und nach der Angabe `AS` der gewünschte Datentyp.
- ✔ Ein Standardwert wird mit dem Schlüsselwort `DEFAULT` angegeben.
- ✔ Wie bei der Namensvergabe von Tabellen vorgeschlagen, können Sie Domänen mit dem Präfix *d_* kennzeichnen.

Eine Domäne kann neben dem Datentyp auch mit verschiedenen Bedingungen und Prüfungen definiert werden. Dafür wird das Schlüsselwort `CHECK` verwendet.

```
CREATE DOMAIN domänenname AS datentyp CHECK (gültigkeitsbedingung);
```

- ✔ Mit dem Schlüsselwort `CHECK` kann eine Bedingung hinzugefügt werden. In Klammern folgt danach ein Ausdruck, der einen Wahrheitswert (wahr oder falsch) liefert. Liefert der Ausdruck den Wert Falsch, wird die entsprechende Operation nicht durchgeführt.

Domänen verwenden

Sie können Domänen in Tabellen auf die gleiche Weise wie Standarddatentypen verwenden.

Beispiel

Es wird eine neue Domäne definiert, die in einer Tabellendefinition eingesetzt wird. Der Vorteil des Einsatzes einer Domäne ist in diesem Fall, dass Sie die Liste der möglichen Werte erweitern können, ohne die Tabellendefinition zu ändern.

```
① CREATE DOMAIN d_abt AS VARCHAR(15)
     CHECK(VALUE IN ("Einkauf","Verkauf","Marketing","Verwaltung","Produktion"));
② CREATE TABLE t_ma_abt
     (id INTEGER,
③     abtname d_abt,
     ma_nr INTEGER);
```

① Hier wird die Domäne d_abt mit dem Datentyp Varchar definiert. Zusätzlich wird eine Bedingung angegeben, die sicherstellt, dass nur die angegebenen Abteilungsnamen verwendet werden können.

② Mit der Anweisung `CREATE TABLE` wird die Tabelle *t_ma_abt* definiert. Sie soll für jeden Mitarbeiter die zugehörige Abteilung speichern.

③ An dieser Stelle wird für das Datenfeld *abtname* als Datentyp die Domäne *d_abt* verwendet.

Domänen ändern

Der Datentyp einer Domäne kann nicht geändert werden, da darauf auch die Typen der Datenfelder in den Tabellen beruhen. Sie können jedoch den Standardwert und die Gültigkeitsprüfung über die Anweisung `ALTER DOMAIN` verändern.

Beispiel

Die Definition der Domäne *d_abt* wird geändert. Es wird ein neuer Standardwert festgelegt und über die Anweisung `DROP CONSTRAINT` die bisherige Gültigkeitsprüfung der Domäne entfernt.

```
ALTER DOMAIN d_abt SET DEFAULT "Produktion";
ALTER DOMAIN d_abt DROP CONSTRAINT;
```

Syntax

```
ALTER DOMAIN domänenname [SET DEFAULT standardwert]
    [DROP DEFAULT]
    [ADD CHECK (gültigkeitsbedingung)]
    [DROP CONSTRAINT];
```

- ✔ Über die Anweisung `ALTER DOMAIN` können Sie eine Domäne bearbeiten. Als erster Parameter wird dabei der Domänenname angegeben.
- ✔ Mit dem danach folgenden Parameter wird angegeben, was geändert oder gelöscht werden soll.

SET DEFAULT standardwert	Legt einen neuen Standardwert fest
DROP DEFAULT	Löscht den bisherigen Standardwert
ADD CHECK (gültigkeitsbedingung)	Fügt eine Gültigkeitsbedingung hinzu
DROP CONSTRAINT	Entfernt die bestehende Gültigkeitsprüfung

Beachten Sie, dass Sie keine vorhandene Gültigkeitsprüfung bearbeiten oder ergänzen können. Sie müssen zuerst die bestehende Prüfung mit `DROP CONSTRAINT` löschen und danach eine neue Prüfung definieren.

Domänen löschen

Eine Domäne löschen Sie mit der Anweisung `DROP DOMAIN`. Danach können Sie die Domäne beispielsweise mit einem anderen Datentyp neu erstellen. Um eine Domäne zu löschen, darf sie in keiner Tabelle als Datentyp verwendet werden. Die entsprechende Datenfelddefinition muss zuerst gelöscht werden.

Syntax

```
DROP DOMAIN domänenname [CASCADE/RESTRICT];
```

- ✔ Die Anweisung `DROP DOMAIN` löscht eine Domäne. Als Parameter muss ein gültiger Domänenname angegeben werden.
- ✔ Der Parameter `CASCADE` löscht alle Objekte, die von der Domäne abhängig sind.
- ✔ Mit der Anweisung `RESTRICT` (Standardwert) können Sie verhindern, dass die Domäne gelöscht wird, sofern noch andere Objekte von ihr abhängen.

6.5 Vorhandene Tabellen anzeigen, ändern und löschen

Vorhandene Tabellen anzeigen

Über die Anweisung `SHOW TABLES` können Sie die in der Datenbank vorhandenen Tabellen auflisten. Die Abbildung zeigt das Ergebnis der Auflistung unter MySQL.

```
C:\>mysql mav
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.55-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW TABLES;
+---------------+
| Tables_in_mav |
+---------------+
| t_lager       |
| t_ma          |
| t_ma_abt      |
+---------------+
3 rows in set (0.00 sec)

mysql>
```

Syntax

```
SHOW TABLES [FROM datenbankname] [LIKE "muster"];
```

- ✔ Die Anweisung `SHOW TABLES` zeigt eine Liste der vorhandenen Tabellen in der aktuell geöffneten Datenbank an.
- ✔ Mit dem Schlüsselwort `FROM` kann eine andere Datenbank benannt werden.
- ✔ Mithilfe von `LIKE` kann die Anzeige der Tabellen auf ein bestimmtes Muster im Tabellennamen beschränkt werden. Durch Verwendung des Musters `m%` werden beispielsweise nur die Tabellen angezeigt, die mit dem Buchstaben `m` beginnen.

Der Befehl `SHOW TABLES` ist in PostgreSQL nicht hinterlegt. Sie können sich in psql die Tabellen der aktuellen Datenbank jedoch anzeigen lassen, indem Sie den Befehl `"\dt"` bzw. `"\d"` eingeben.

Tabellenstruktur ändern

Die Struktur einer Tabelle können Sie jederzeit über die Anweisung `ALTER TABLE` ändern. Dabei können Sie

- ✔ Datenfelder hinzufügen oder löschen
- ✔ Datenfelddefinitionen verändern
- ✔ Gültigkeitsprüfungen hinzufügen oder löschen
- ✔ Schlüssel und Indizes hinzufügen oder löschen

Mit dem Schlüsselwort `ADD` fügen Sie einer bestehenden Tabelle ein neues Datenfeld hinzu. Die bereits in der Tabelle vorhandenen Datensätze enthalten danach ein neues leeres Datenfeld, das gegebenenfalls den festgelegten Standardwert enthält. Für das Löschen eines Datenfelds, Schlüssels oder Indizes wird die Anweisung `DROP` verwendet.

Beispiel: *AlterTableBeispiele.sql*

Im Folgenden werden einige Anweisungen zum Ändern der Tabellenstruktur vorgestellt.

```
① ALTER TABLE t_lager
     ADD artikel VARCHAR(20) DEFAULT "unbekannt";
② ALTER TABLE t_ma_abt
     ADD PRIMARY KEY (id);
③ ALTER TABLE t_lager
     ADD CONSTRAINT ppruef CHECK (preis > 0);
④ ALTER TABLE t_lager
     DROP wert;
```

① Der Tabelle *t_lager* wird das neue Datenfeld *artikel* mit dem Standardwert unbekannt hinzugefügt.
② Mit dieser Anweisung wird das Feld *id* der Tabelle *t_ma_abt* nachträglich als Primärschlüssel definiert.
③ Hier wird eine neue Gültigkeitsbedingung für die Tabelle *t_lager* definiert. Der Wert für das Feld *preis* wird bei der Eingabe überprüft; er muss größer als 0 sein.
④ Mithilfe der Anweisung `DROP` wird das angegebene Datenfeld in der Tabelle gelöscht.

Beim Löschen eines Datenfelds werden die enthaltenen Werte in allen Datensätzen gelöscht.

Syntax

```
ALTER TABLE tabellenname
   [ADD datenfelddefinition]
   [ADD indexdefinition]
   [ADD CONSTRAINT constraintname CHECK (gültigkeitsbedingung)]
   [DROP objektname];
```

- ✔ Für das Ändern einer Tabelle wird die Anweisung `ALTER TABLE` verwendet. Danach folgt der Name der zu ändernden Tabelle.
- ✔ Mit dem Schlüsselwort `ADD` wird das Hinzufügen eines Datenfelds, Indizes oder Schlüssels oder einer Gültigkeitsbedingung eingeleitet.
- ✔ Mit der Klausel `DROP` wird ein Datenfeld, Index, Schlüssel oder eine Gültigkeitsbedingung gelöscht.
- ✔ Die Definition eines Datenfelds, Indexes, Schlüssels oder einer Gültigkeitsbedingung folgt der gleichen Syntax wie die Erstellung einer Tabelle.
- ✔ Gültigkeitsprüfungen gelten nur für die nach der Änderung neu hinzugefügten oder geänderten Datensätze.

Möchten Sie sich die Tabellenstruktur in der Konsole anzeigen lassen, geben Sie bei PostgreSQL den Befehl "*\d tabellenname*" ein.

Bei MySQL können Sie sich die Tabellenstruktur mithilfe des Befehls `SHOW TABLE STATUS` anzeigen lassen.

```
uebungen=# \d t_person
        Tabelle "public.t_person"
   Spalte    |          Typ          | Attribute
-------------+-----------------------+-----------
 id          | integer               | not null
 vname       | character varying(150) | not null
 nname       | character varying(150) | not null
 lebenslauf  | character varying     |

uebungen=#
```

Tabellenstruktur der Tabelle t_person in PostgreSQL anzeigen

Tabellen löschen

Zum Löschen einer Tabelle verwenden Sie die Anweisung `DROP TABLE`. Dabei werden die Definition der Tabellenstruktur und alle in der Tabelle gespeicherten Datensätze gelöscht.

Syntax

```
DROP TABLE tabellenname;
```

- ✔ Mit der Anweisung `DROP TABLE` wird die angegebene Tabelle mit allen enthaltenen Daten gelöscht.

Beim Löschen einer Tabelle müssen Sie folgende Besonderheiten beachten:

- ✔ Wenn andere Datenbankobjekte die zu löschende Tabelle verwenden oder referenzieren, z. B. durch eine Sicht `(VIEW)` oder eine gespeicherte Prozedur `(STORED PROCEDURE)`, ist das Löschen nicht möglich. Es müssen zuerst alle Referenzen entfernt werden.
- ✔ Sie müssen entweder Eigentümer der Tabelle sein oder über die notwendigen Rechte zum Löschen der Tabelle verfügen.
- ✔ Es dürfen keine Transaktionen aktiv sein, die die betreffende Tabelle verwenden.

6.6 Schnellübersicht

Was bedeutet ...?	
Datenbank	Oberstes Objekt in der Datenbankhierarchie, welches einen Container für die darin gespeicherten Tabellen bzw. weiteren Datenbankobjekte darstellt
Tabelle	Stellt eine definierte Struktur für das Speichern von Daten zur Verfügung
Datenfeld (Attribut)	Entspricht der Spalte einer Tabelle; über den Datentyp des Datenfeldes wird die Art der zu speichernden Daten in dieser Spalte festgelegt.
Datensatz (Tupel)	Entspricht der Zeile einer Tabelle und enthält die Werte für alle dazugehörigen Datenfelder
Constraint	Einfache Bedingung für die zu speichernden Daten
Domäne	Benutzerdefinierter Datentyp mit der zusätzlichen Möglichkeit, Gültigkeitsprüfungen und Standardwerte festzulegen
Datentyp	Durch den Datentyp wird festgelegt, welche Art von Daten (Zahlen, Zeichenketten usw.) in einem Feld gespeichert werden können.

Sie möchten ...	
eine Tabelle erstellen	`CREATE TABLE tabellenname (datenfelddefinition);`
die Eingabe eines Werts für ein Datenfeld erzwingen	`NOT NULL`
einen Standardwert festlegen	`DEFAULT wert`
eine Gültigkeitsbedingung definieren	`CONSTRAINT name CHECK (bedingung)`
einen Primärschlüssel definieren	`PRIMARY KEY (datenfeld)`
die Werte in einem Feld automatisch erhöhen lassen	`AUTO_INCREMENT (MySQL) bzw. SERIAL (PostgreSQL)`
eine Tabelle ändern	`ALTER TABLE tabellenname;`
eine Tabelle löschen	`DROP TABLE tabellenname;`
eine Domäne erstellen	`CREATE DOMAIN domänenname;`
eine Domäne ändern	`ALTER DOMAIN domänenname;`
eine Domäne löschen	`DROP DOMAIN domänenname;`
einen Datentyp ändern	`CAST (Wert AS Datentyp)`

6.7 Übung

Tabellen erstellen und löschen

Übungsdatei: -- **Ergebnisdatei: *Uebung1.sql, Uebung2.sql, Uebung3.sql***

① Erstellen Sie eine Datenbank mit dem Namen *testdb*.
Wechseln Sie zu dieser neuen Datenbank.
Erstellen Sie eine neue Tabelle *t_artikel* mit den Datenfeldern *id*, *name* und *preis*. Verwenden Sie geeignete Datentypen.
Löschen Sie die Tabelle und anschließend die Datenbank.

② Wechseln Sie zur Datenbank *Uebungen*.
Erstellen Sie eine Tabelle *t_person* mit den Datenfeldern *id*, *vorname* und *nachname*. Verwenden Sie dabei geeignete Datentypen.
Für alle Datenfelder soll eine Eingabe erforderlich sein. Erweitern Sie die Definition der Datenfelder für diesen Zweck.
Fügen Sie ein neues Datenfeld *beschaeftigt_seit* in die Tabelle ein. Es soll einen Datumswert speichern.
Löschen Sie das Datenfeld wieder.

③ Wechseln Sie zur Datenbank *Uebungen*.
Löschen Sie die in Übung 2 erstellte Tabelle *t_person*.
Erstellen Sie eine geeignete Domäne, die als Datentyp für die Speicherung von Vorname und Nachname verwendet werden soll. Es soll für das Datenfeld unbedingt eine Eingabe erforderlich sein. Nennen Sie die Domäne *Namen*.
Erstellen Sie die Tabelle *t_person* mit folgenden Datenfeldern: *id*, *vname*, *name*, *strasse*, *plz* und *ort*. Verwenden Sie dabei geeignete Datentypen bzw. die neu definierte Domäne.