Kapitel 1

SQL - DDL (Data Definition Language)

1.1 Datentypen in SQL

SQL kennt verschiedene Arten von Datentypen, die sie sich auf folgender Seite durchlesen sollen:

http://de.wikibooks.org/wiki/Einf%C3%BChrung_in_SQL:_Datentypen

1.2 Anlegen von Tabellen

```
CREATE TABLE-Anweisung :=
   CREATE TABLE Tabellenname
   (    <Spaltendefinition> [, <Spaltendefinition> ...]
        [, <Constraint>]
   );
```

wobei

```
<Spaltendefinition> :=
   Spaltenname <Datentyp>
   [DEFAULT <Ausdruck> ] [NOT NULL ][NULL]
   [, <Constraint>]
```

```
<Constraint> :=
   { [CONSTRAINT Constraintname ]
   | [NULL | NOT NULL]
   | [UNIQUE | PRIMARY KEY]
   | [FOREIGN KEY (Spaltenname [,...] ) ]
   | [REFERENCES Tabellenname ( Spaltenname[,...] ) [ON DELETE CASCADE]]
   | [CHECK (<Checkbedingung>)] }
```

```
<Checkbedingung> :=
   {<Ausdruck> <Vergleichsoperator> <Ausdruck>
   | <Ausdruck> [NOT] BETWEEN <Ausdruck> AND <Ausdruck>
   | <Ausdruck> [NOT] IN ( Konstante [, Konstante ...] )
   | <alphanumerischer Ausdruck> [NOT] LIKE <Schablone>
   | <Ausdruck> IS [NOT] NULL }
```

Anmerkung:

NULL	legt fest, dass eine Spalte NULL-Werte haben kann
NOT NULL	legt fest, dass eine Spalte keine NULL-Werte haben kann
UNIQUE	bestimmt eine oder mehrere Spalten als eindeutige Schlüssel
PRIMARY KEY	bestimmt eine oder mehrere Spalten als Primärschlüssel
FOREIGN KEY	bestimmt eine oder mehrere Spalten als Fremdschlüssel, die der referentiellen Integrität genügen müssen
REFERENCES	identifiziert den Primärschlüssel, der als FOREIGN-KEY festgelegt wurde
ON DELETE CASCADE	Referentielle Integrität wird gesichert, indem automatisch Zeilen, die diese Integrität verletzen, entfernt werden. (Löschen eines Datensatzes führt zum kaskadierenden Löschen der über foreign key constraints verbundenen Datensätze)
CHECK	legt eine Bedingung fest, die jede Zeile der Tabelle erfüllen muss

1.3 Ändern von Tabellen

Mit der ALTER-TABLE-Anweisung können nachträglich Spalten in Tabellen verändert, gelöscht oder hinzugefügt werden. Insbesondere können auch CONSTRAINTS nachträglich bearbeitet oder zeitweise außer Kraft gesetzt werden.

```
ALTER-TABLE-Anweisung :=
   ALTER TABLE Tabellenname
   { ADD { (<Spaltendef> [, <Spaltendef> ... ] ) | CONSTRAINT Constraintname }
   | DROP { (<Spaltendef> [, <Spaltendef> ... ] ) | CONSTRAINT Constraintname }
   | MODIFY (<Spaltendef> [, <Spaltendef> ... ] ) }

   [ENABLE | DISABLE ] {
   ALL TRIGGERS
   | UNIQUE (Spaltenname, [, Spaltenname ...])
   | PRIMARY KEY
   | CONSTRAINT Constraintname } ]
```

1.4 Löschen von Tabellen

```
DROP-TABLE-Anweisung :=
  DROP TABLE Tabellenname;
```

1.5 Aufgabe 9

Entwickeln Sie die CREATE TABLE - Statements für nachfolgende Relationen.

```
Weinsorte   (  WeinsorteName, Anmerkung                                          )
Winzer      (  WinzerCode, Name, Adresse, StartJahr                              )
Ernte       (  WeinsorteName, WinzerCode, ErnteJahr, Menge                       )
Kunde       (  KundenID, Name, Adresse, Kontostand                               )
bevorzugt   (  KundenID, WeinsorteName, WinzerCode                               )
Bestellung  (  KundenID, Nummer, WeinsorteName, WinzerCode, ErnteJahr, Anzahl    )
Geschenk    (  KundenID, Datum, Beschreibung                                     )
```

1.6 Aufgabe 10

Entwickeln Sie die nötigen SQL-Anweisungen um die soeben erstellen Relationen abzuändern:

- Fügen Sie zur Relation Ernte eine CHECK-Klausel hinzu (`Menge>0`)
- Fügen Sie zur Relation Ernte ein Attribut Preis (inkl. CHECK-Klausel) hinzu
  `Ernte (WeinsorteName, WinzerCode, Jahr, Menge, Preis)`
- Entfernen Sie aus der Relation Winzer das Attribut StartJahr
  `Winzer (WinzerCode, Name, Adresse)`
- Fügen Sie zur Relation Bestellung ein Attribut Datum zum Primärschlüssel hinzu
  `Bestellung(Datum, KundenId, Nummer, WeinsorteName, WinzerCode, ErnteJahr, Anzahl)`
- Entfernen Sie aus der Relation bevorzugt das Attribut WinzerCode
  `bevorzugt (KundenId, WeinsorteName)`
  (Welches Problem könnte dabei entstehen?)
- Löschen Sie die Relation Geschenk