Handout
Themenfeld: Datenbanken und SQL
Abschnitt: 06.01. SQL DDL Grundlagen

Autor: Thomas Krause
Stand: 14.11.2022 12:03:00

Inhalt

1 Einführung und Überblick zu SQL

1.1 SQL – Einführung, Überblick, Grundlagen

Der Begriff SQL:

- SQL (= Structured Query Language) ist Standardsprache für **relationale** Datenbanken
- Vorgänger war die Datenbankabfragesprache SEQUEL (structured english query language) → dieser Name ist im Zusammenhang mit SQL noch häufig in Gebrauch
- genormte Sprache mit definierter Syntax (aber mit Hersteller-Besonderheiten)
- SQL wurde 1986 durch ANSI (American National Standards Institute) und dann ab 1987 durch die ISO (International Organization for Standardization) standardisiert
  Der Standard besteht aus mehreren Teilen auf unterschiedlichen Aktualitätsständen.

1.2 Zweck, Aufbau/ Kategorien, Wirkungsweise

- **der Inhalt von SQL → man kann 4 Gruppen von Befehlen/ Funktionen unterscheiden:**
  - DDL = Data Definition Language (Datenbankbeschreibung, Definition von Datenstrukturen)
  - DML = Data Manipulation Language (Bearbeiten von Daten, Einfügen, Ändern, Löschen)
  - DQL = Data Query Language (Abfragen von Daten aus Datenbanken)
  - DCL = Data Control Language (Datenbank- und Nutzerverwaltung)
  - TCL = Transaction Control Language (Steuerung von Transaktionen)
- **die Dialekte von SQL:**
  - auf allen (wichtigen) Systemen verfügbar
  - SQL ist ein allgemeiner Standard → ISO-/ ANSI-Standard --> aber mit Hersteller-Besonderheiten
  - Microsoft: T-SQL (= Transact-SQL)
  - Oracle, OpenSource: MySQL (→ MariaDB)
  - Oracle, IBM, PostgreSQL

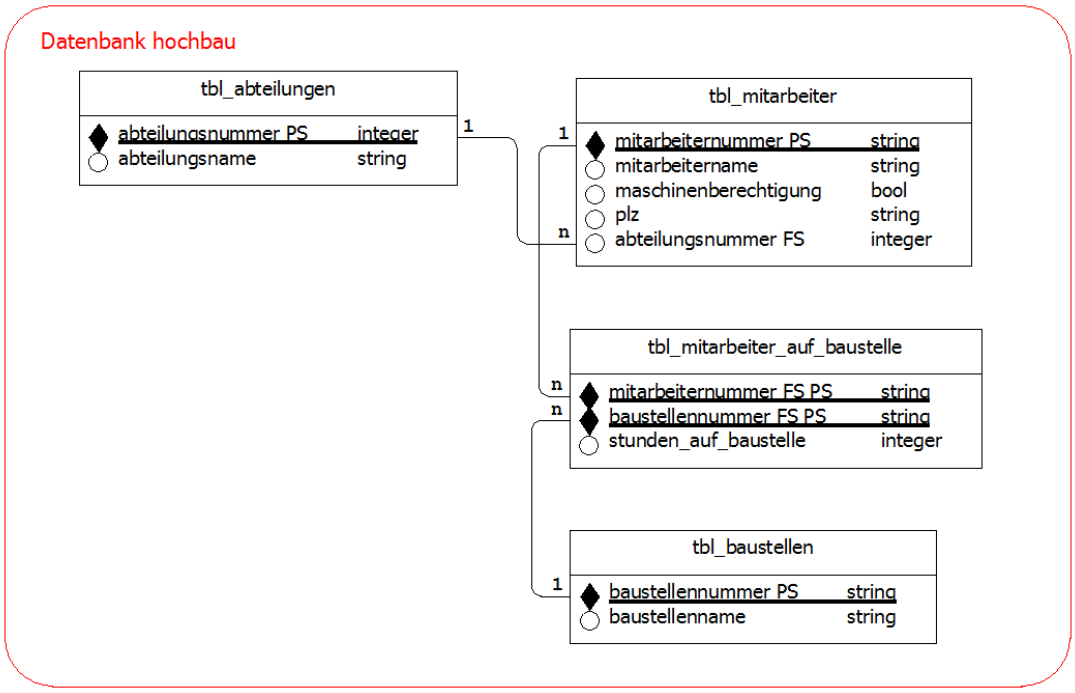
- **die Anwendung:**
  - Inline SQL/ Embedded SQL: in andere Programmiersprachen (C, C++, Pascal, Cobol, Ada, …) eingebundene SQL-Anweisungen
  - Programmierschnittstellen: z.B. ODBC, JDBC, ADO: Übergabe von Befehlen an ein Datenbanksystem
  - Dynamic SQL: Zusammensetzung von SQL-Anweisungen zur Laufzeit von Programmen
  - SQLCMD: Ausführung von Anweisungen oder auch ganzen Skripten auf einer Kommandozeile
  - MySQL-/ MariaDB Workbench: interaktive Ausführung von Anweisungen und Skripten in der Benutzeroberfläche
  - Microsoft SQL Server Management Studio: interaktive Ausführung von Anweisungen und Skripten in der Benutzeroberfläche

2 SQL DDL Data Definition Language: Grundlagen

2.1 Überblick

- **die (wichtigsten) Aufgaben der DDL:**
  Erstellen bzw. Löschen von Datenbanken
  Erstellen, Ändern, Löschen der Tabellen in den Datenbanken
- grundlegende Befehle (Auswahl):
  - Erzeugen von Datenbanken: CREATE DATABASE
  - Löschen von Datenbanken: DROP DATABASE
  - Erzeugen von Relationen/ Basistabellen: CREATE TABLE
  - Ändern des Schemas einer Relation: ALTER TABLE
  - Löschen von Relationen/ Basistabellen: DROP TABLE
  - Erzeugen von Indizes: CREATE INDEX
  - Löschen von Indizes: DROP INDEX
- die Grundfunktionen existieren in allen Systemen → einige Namen können abweichen

Modell für die weiteren Erläuterungen:



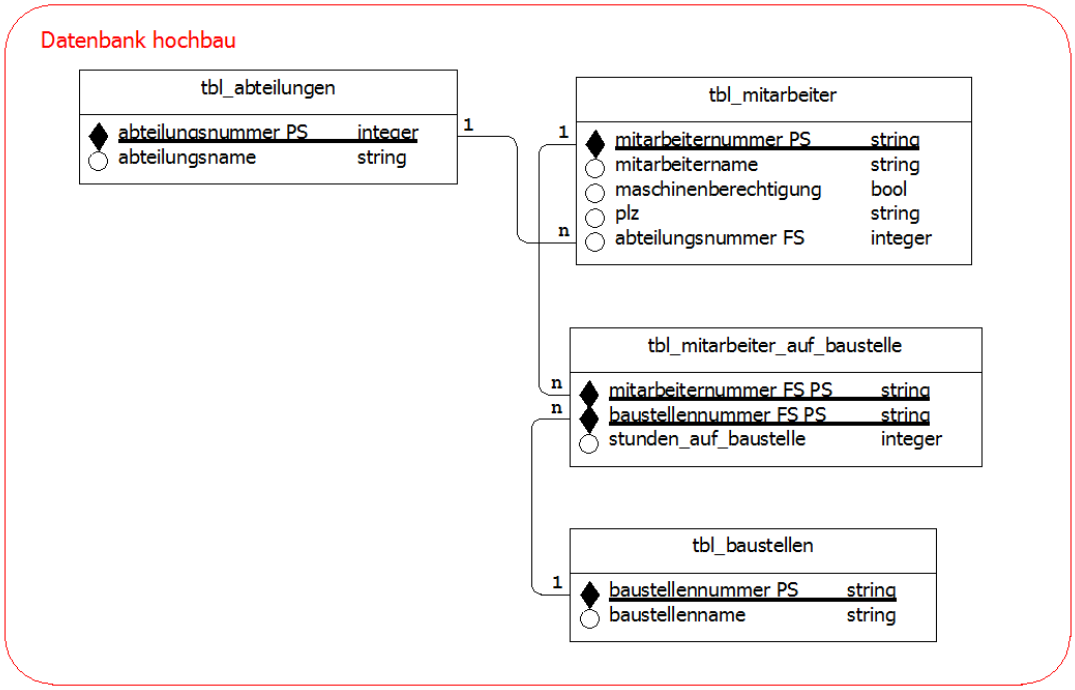
(Name der Datenbank kann individuell abweichen)

2.2 DDL: Datenbank neu anlegen

Aufgabe/ Beispiel:

Anlegen der neuen Datenbank hochbau (ohne Tabellen)

Gegebenes Tabellenmodell:



SQL-Anweisung:

```
CREATE DATABASE hochbau;
```

Syntax:

```
CREATE DATABASE <name_der_datenbank>;
USE <name_der_datenbank>;
```

Erläuterung:

- Name der Datenbank muss eindeutig sein
- soll die Datenbank genutzt werden, muss sie mit USE zur aktiven Datenbank gemacht werden

praktische Anwendung:

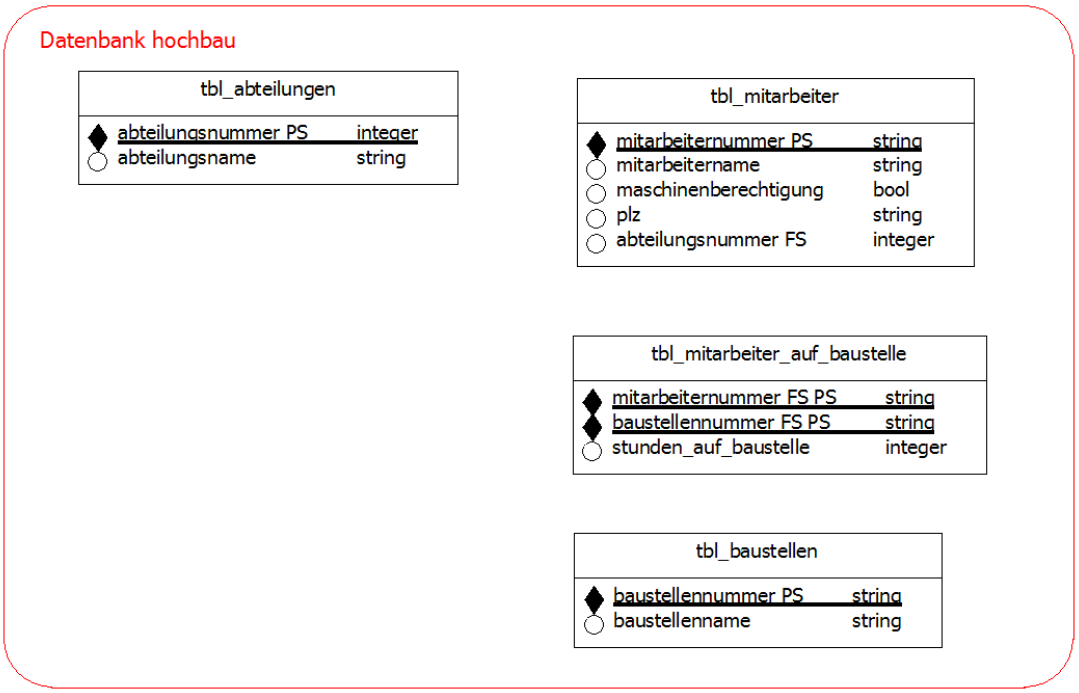
N/A

2.3 DDL: Tabelle neu anlegen (ohne Fremdschlüssel-Beziehungen)

Aufgabe/ Beispiel:

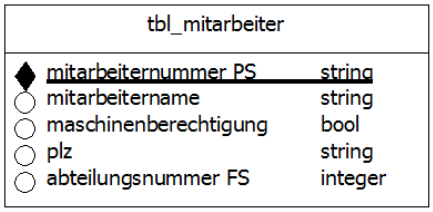
Neue Tabelle (mit Primärschlüssel, aber noch ohne Fremdschlüssel-Beziehungen) in einer vorhandenen Datenbank anlegen.

Gegebenes Tabellenmodell:



ACHTUNG: Es wird zunächst von einem vereinfachten Tabellen-Modell ohne Fremdschlüssel-Beziehungen ausgegangen.

Anzulegende Tabelle:



SQL-Anweisung:

VARIANTE 1:

```
CREATE TABLE tbl_mitarbeiter
      (mitarbeiternummer char(4) PRIMARY KEY,
      maschinenberechtigung bit,
      mitarbeitername char(50),
      mitarbeiterPLZ char(5),
      abteilungsnummer int);
```

VARIANTE 2:

```
CREATE TABLE tbl_mitarbeiter
       (mitarbeiternummer char(4),
       maschinenberechtigung bit,
       mitarbeitername char(50),
       mitarbeiterPLZ char(5),
       abteilungsnummer int,
       PRIMARY KEY (mitarbeiternummer));
```

-- die Fremdschlüsselbeziehungen für abteilungsnummer sind in diesen beiden Varianten noch nicht berücksichtigt

Syntax:

allgemeine Syntax:

```
CREATE TABLE <name_der_tabelle>
(<spalten_name> <datentyp> [<constraint>] [,
<weitere_spaltendefinition>] [,
<constraint>]);
```

Syntax speziell mit dem Constraint 'Primärschlüssel/ primary key':

Variante 1:

```
CREATE TABLE <name_der_tabelle>
(<spalten_name> <datentyp> PRIMARY KEY [<constraint>] [,
<weitere_spaltendefinition>] [,
<constraint>]);
```

Variante 2:

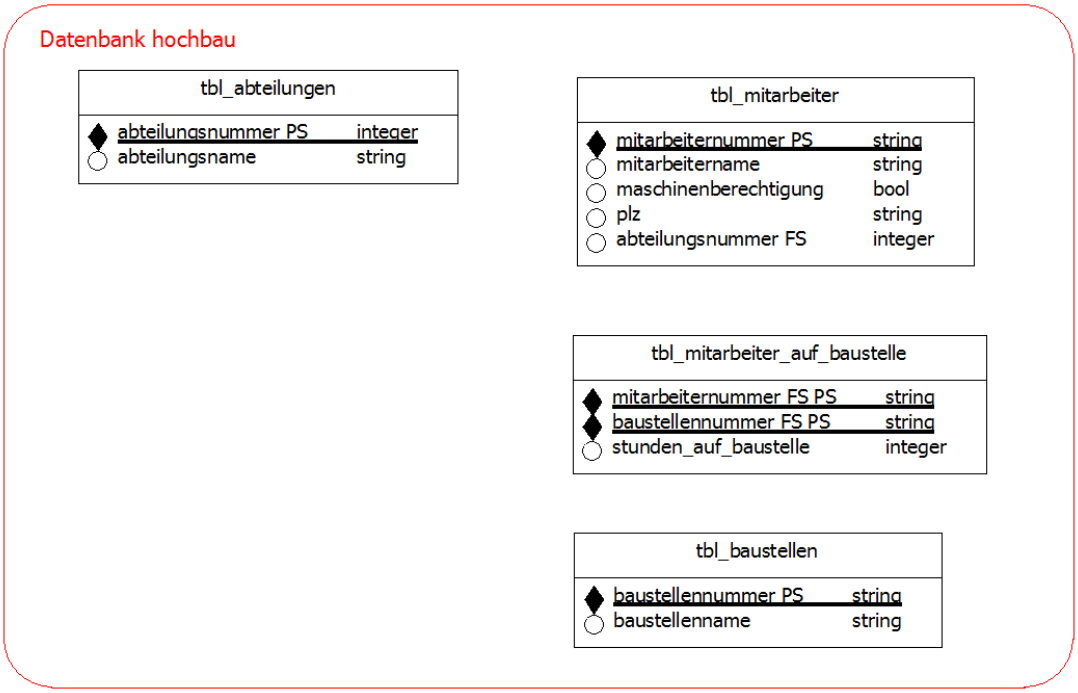
```
CREATE TABLE <name_der_tabelle>
(<spalten_name> <datentyp> [<constraint>] [,
<weitere_spaltendefinition>] [,
<constraint>],
PRIMARY KEY (<spalten_name1>[, <spaltenname2>, …]));
```

Erläuterung:

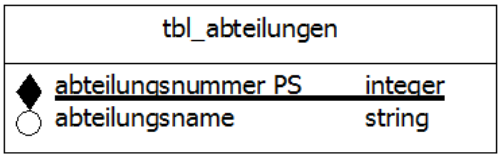
- Tabellennamen müssen innerhalb einer Datenbank eindeutig sein
- speziell für zusammengesetzte Primärschlüssel wird die Variante 2 benötigt
- weitere constraints werden nachfolgend noch vertieft
- die Reihenfolge der Angaben in der Klammer ist beliebig; es ist empfohlen, eine gewisse Reihenfolge für die bessere Lesbarkeit einzuhalten

praktische Anwendung:

Gegebenes Tabellenmodell:



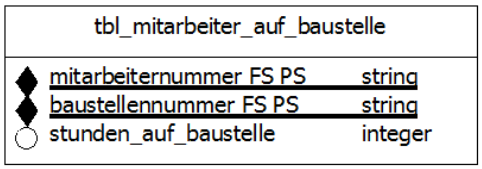
Anzulegende Tabelle:



SQL-Anweisung:

```
CREATE TABLE tbl_abteilungen
      (abteilungsnummer int PRIMARY KEY,
      abteilungsname char(50));
```

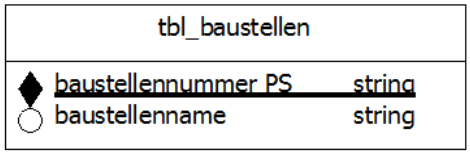
Anzulegende Tabelle:



SQL-Anweisung:

```
CREATE TABLE tbl_ma_auf_baustelle
       (mitarbeiternummer char(4),
       baustellennummer char(4),
       stunden_auf_baustelle decimal(8,2),
       PRIMARY KEY(mitarbeiternummer, baustellennummer));
```

Anzulegende Tabelle:



SQL-Anweisung:

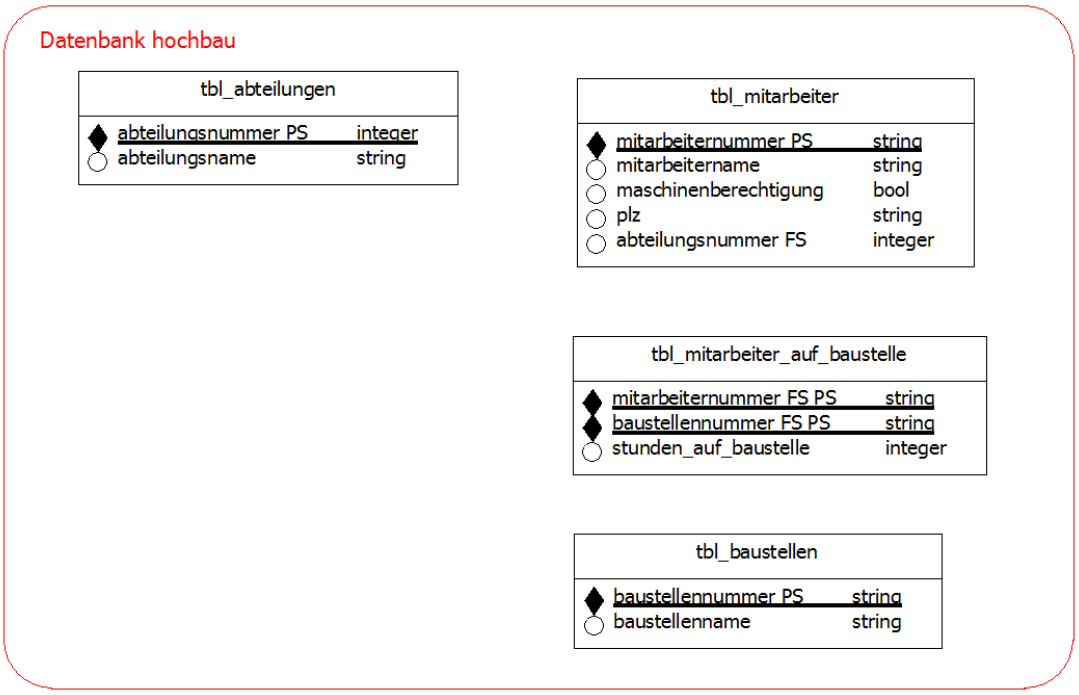
```
CREATE TABLE tbl_baustelle
       (baustellennummer char(4) PRIMARY KEY,
       baustellenname char(150));
```

2.4 DDL: Tabelle löschen

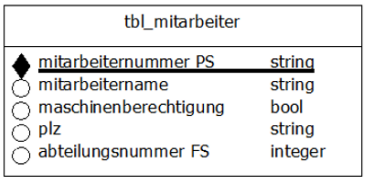
Aufgabe/ Beispiel:

Löschen einer Tabelle aus einer Datenbank

Gegebenes Tabellenmodell:



Zu löschende Tabelle:



Hinweis:

- alle Tabellen haben in dieser Version keine Fremdschlüsselbeziehungen

SQL-Anweisung:

```
DROP TABLE tbl_mitarbeiter;
```

Syntax:

```
DROP TABLE <name_der_tabelle>;
```

Erläuterungen:

- löscht die Tabelle komplett mit Inhalt OHNE Nachfrage

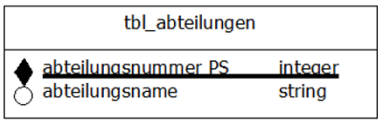
- sollte die zu löschende Tabelle an Fremdschlüsselbeziehungen beteiligt sein, wird das Löschen standardmäßig vom DBMS blockiert
- es gibt Sonderfälle, die später berücksichtigt werden

praktische Anwendung:

Gegebenes Tabellenmodell:

siehe oben

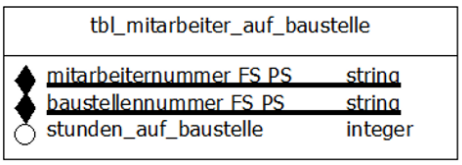
Zu löschende Tabelle:



SQL-Anweisung:

```
DROP TABLE tbl_abteilungen;
```

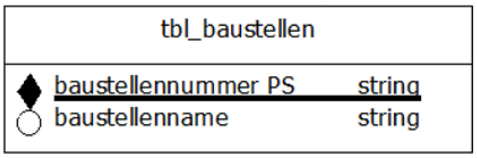
Zu löschende Tabelle:



SQL-Anweisung:

```
DROP TABLE tbl_mitarbeiter_auf_baustelle;
```

Zu löschende Tabelle:



SQL-Anweisung:

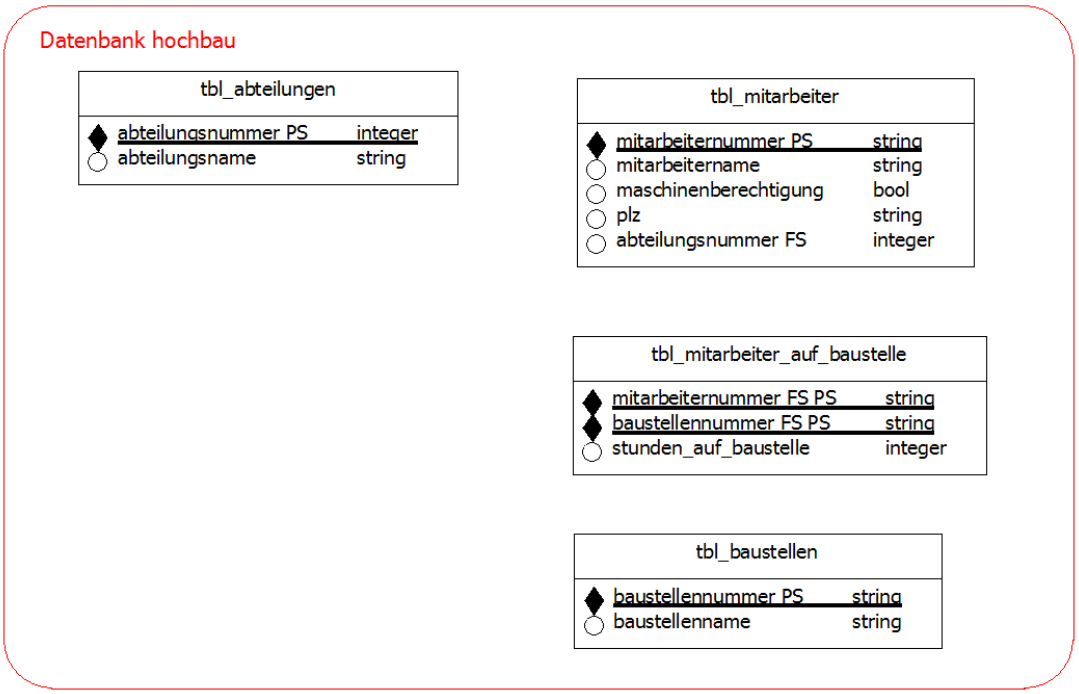
```
DROP TABLE tbl_baustellen;
```

2.5 DDL: Datenbank löschen

Aufgabe/ Beispiel:

Löschen einer vorhandenen Datenbank.

Gegebenes Tabellenmodell:



SQL-Anweisung:

```
DROP DATABASE hochbau;
```

Syntax:

```
DROP DATABASE <name_der_datenbank>
```

Erläuterungen:

- löscht OHNE Nachfrage die gesamte Datenbank
- u.U. kann das Löschen durch das DBMS blockiert werden, wenn die Datenbank durch eine andere Session mit USE <name_der_datenbank> aktiviert wurde und aktuell in Benutzung ist

praktische Anwendung:

N/A

2.6 DDL: Überblick über ausgewählte Constraints

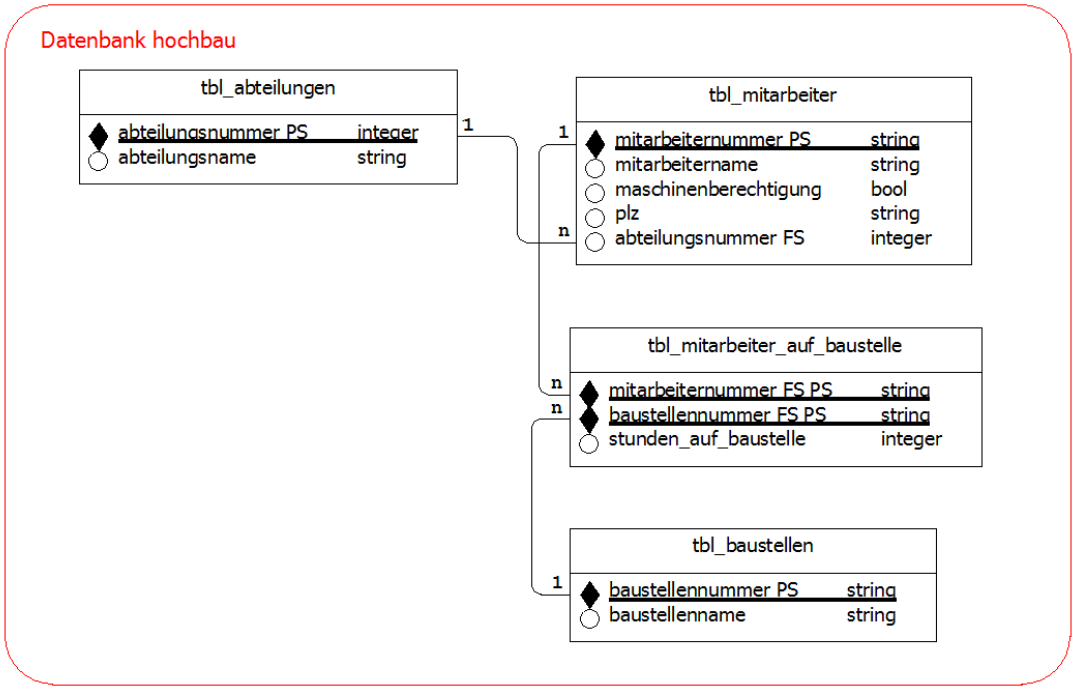
- Constraint = Bedingung, Einschränkung
- C. sind spezielle Zusätze für Datentypen zur Integritätssicherung:
  - **primary key** → Festlegung des Attributs als Primärschlüssel in der Tabelle → schließt die Eigenschaften „not null“ und „unique“ automatisch mit ein; **Siehe dazu Abschnitt 2.3**
  - **references** → definiert das Feld als Fremdschlüssel, muß also mit dem Primärschlüssel einer anderen Relation übereinstimmen **Siehe dazu Abschnitt 2.7**
  - **Die folgenden Constraints werden in einem späteren Abschnitt/ Tag eingeführt (zunächst hier also nur zur Info):**
  - **not null** → muß eingegeben werden, Feld darf nicht leer bleiben, muß definiert sein (NULL ≠ 0)
  - **autoincrement** → automatische Numerierung eines Feldes durch das DBMS
  - **default** → automatische Belegung eines Feldes mit einem vorgegebenen Wert
  - **unique** → muß eindeutig sein, Wert in diesem Feld darf in der Tabelle nicht mehrfach existieren → alle Werte in dieser Spalte müssen unterschiedlich sein
  - **check** → regelbasierte Überprüfung zulässiger Werte in einem Feld
- (Diese Auflistung ist nicht vollständig!)

2.7 DDL: Tabelle anlegen (mit Fremdschlüssel-Beziehungen)

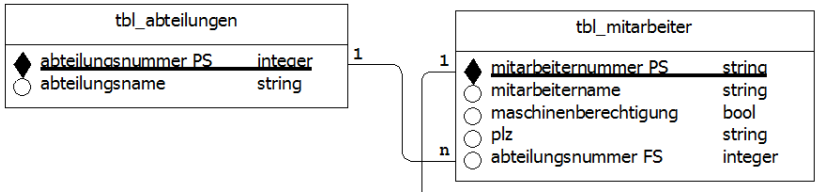
Aufgabe/ Beispiel:

Neue Tabelle in einer vorhandenen Datenbank anlegen **mit einer Fremdschlüsselbeziehung** zu einer anderen Tabelle.

Gegebenes Tabellenmodell:



Anzulegende Tabelle:



HINWEIS: Die neu anzulegende Tabelle ist tbl_mitarbeiter. Sie hat eine Fremdschlüsselbeziehung zu tbl_abteilungen

SQL-Anweisung:

```
CREATE TABLE tbl_mitarbeiter
        (mitarbeiternummer char(4) PRIMARY KEY,
        maschinenberechtigung bit,
        mitarbeitername char(50),
        mitarbeiterPLZ char(5),
        abteilungsnummer int,
        foreign key (abteilungsnummer) references tbl_abteilungen(abteilungsnummer));
```

Syntax:

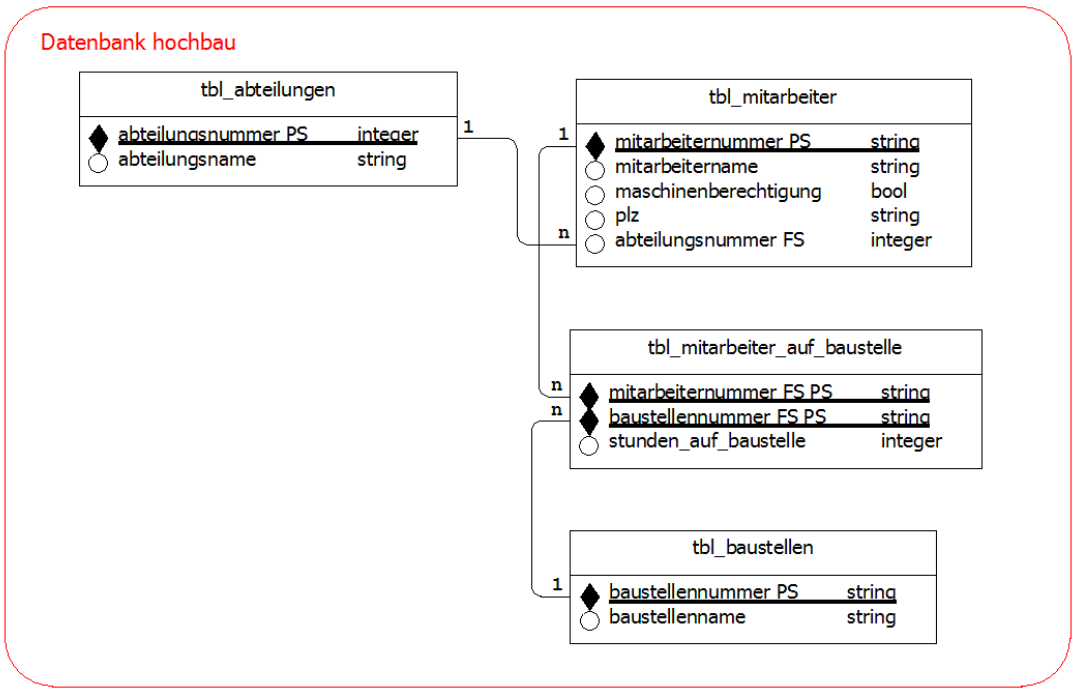
```
CREATE TABLE <name_der_tabelle>
(<spalten_name> <datentyp> PRIMARY KEY [<constraint>] [,
<weitere_spaltendefinition>] [,
<constraint>],
FOREIGN KEY (<name_fs_spalte>) REFERENCES <name_ps_tabelle>(<name_ps_spalte>));
```

Erläuterungen:

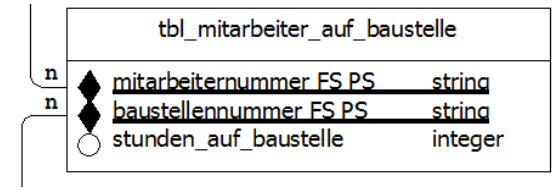
- fs_spalte = ist die Spalte in der aktuellen Tabelle, die eine Beziehung zu einer anderen Tabelle (ps_tabelle) erhalten soll;
  ist im Tabellenmodell mit n gekennzeichnet
- ps_spalte = ist die Spalte in der anderen Tabelle, der sogenannten Primärschlüsseltabelle, auf die die Beziehung "zeigt";
  die ps_spalte ist in der ps_tabelle der Primärschlüssel (also eindeutig und muss vorhanden sein);
  ist im Tabellenmodell mit 1 gekennzeichnet
- ps_tabelle = ist die Tabelle, die die ps_spalte enthält
- Beachte die spezielle Darstellung von constraints in DIA (ist nicht allgemeingültig)

praktische Anwendung:

Gegebenes Tabellenmodell:



Anzulegende Tabelle:



ACHTUNG: Grundsätzlich wie im Abschnitt zuvor, jetzt jedoch mit 2 Fremdschlüsselbeziehungen.

SQL-Anweisung:

```
CREATE TABLE tbl_ma_auf_baustelle
        (mitarbeiternummer char(4),
        baustellennummer char(4),
        stunden_auf_baustelle decimal(8,2),
        PRIMARY KEY(mitarbeiternummer, baustellennummer),
        FOREIGN KEY (mitarbeiternummer) references tbl_mitarbeiter(mitarbeiternummer),
        FOREIGN KEY (baustellennummer) references tbl_baustelle(baustellennummer));
```

ACHTUNG:

Die folgende Anweisung wird von **MySQL** als fehlerfrei angezeigt und ausgeführt. Trotzdem wird keine Fremdschlüsselbeziehung in der Tabelle angelegt (siehe Spalte 'abteilungsnummer').
Diese Syntax kann in anderen DBMS durchaus zulässig und erfolgreich sein (z.B. Microsoft SQL):

```
create table tbl_mitarbeiter
       (mitarbeiternummer char(4) PRIMARY KEY,
       maschinenberechtigung bit NOT NULL,
       mitarbeitername char(50) NOT NULL,
       mitarbeiterPLZ char(5) NOT NULL,
       abteilungsnummer int REFERENCES tbl_abteilung(abteilungsnummer));
```