Übungsblatt 10 - Lösungen

Gruppenaufgabe 1

a) **Professoren**
- Primärschlüssel-Constraint: PersNr muss eindeutig und nicht null sein
- not null: Name darf nicht null sein
- CHECK-Constraint: Rang darf nur die Werte ‚C2', C3' oder ‚C4' annehmen
- unique-Constraint: Raum muss eindeutig sein

Assistenten
- Primärschlüssel-Constraint: PersNr muss eindeutig und nicht null sein
- not null: Name darf nicht null sein
- Foreign Key Constraint: Boss referenziert die Tabelle Professoren. Der Wert Boss muss also als PersNr in der Tabelle Professoren existieren.
- Einschränkung des FK Constraints mit on delete set null: Wird ein referenzierter Professor gelöscht, wird der referenzierende Wert von Boss auf NULL gesetzt.

b) **SET NULL**: Wird eine referenzierte Zeile (z.B. Prof) gelöscht, wird in Zeilen mit referenzierendem Wert (Boss in Assistenten) dieser auf NULL gesetzt.

CASCADE: Wird eine referenzierte Zeile gelöscht (z.B. Student), werden Zeilen, die diesen referenzieren (in Tabelle hören) ebenfalls gelöscht.

Gruppenaufgabe 2

Die in Abbildung 2.3 angegebenen Schlüssel der Entitätstypen wurden in der Überführung ins relationale Schema (Abbildung 5.1) als Primärschlüssel realisiert. Dabei sind die Schlüssel der Relationen *hören*, *voraussetzen* und *prüfen* aufgrund der N:M-Multiplizitäten jeweils Tupel bestehend aus den Fremdschlüsseln beider Entitytypen der binären Beziehungen. Für *hören* ist dies (*MatrNr*, *VorlNr*). Zusätzlich wurden folgende Constraints realisiert:

- Relation *Studenten*: mittels des **check**-Constraints wird festgelegt, dass die Semesteranzahl eines Studenten minimal 1 und maximal 13 ist.
- Relation *Professoren*:
  - Der Rang eines Professors ist entweder C2, C3 oder C4.
  - Jedem Professor ist ein eigenes Büro zugeordnet, was über das **unique** Schlüsselwort erreicht wird.
- Relation *Assistenten*: Hier wird keine zusätzliche Integritätsbedingung realisiert. Mittels der Definition

  foreign key (Boss) **references** Professoren **on delete set null**

  wird die N:1-Beziehung der ER-Modellierung realisiert: Assistenten haben höchstens einen betreuenden Professor, können aber auch selbständig arbeiten, d.h. keinem Professor zugeordnet sein.
- Relation *Vorlesungen*: Es werden keine zusätzlichen Integritätsbedingungen umgesetzt.
- Relation *hören*: Es werden keine zusätzlichen Integritätsbedingungen umgesetzt. Die **on delete cascade**-Bedingungen sind notwendig, da *MatrNr* und *VorlNr* den Primärschlüssel bilden und deshalb implizit als **not null** definiert sind. **null**-Belegungen sind aber auch aus Anwendungssicht nicht sinnvoll.
- Relation *voraussetzen*: Es werden keine zusätzlichen Integritätsbedingungen umgesetzt.
- Relation *prüfen*:
  - Der Schlüssel der Relation ist {*MatrNr*, *VorlNr*}. Aufgrund des Constraints **on delete set null** ist es möglich, dass, wird der prüfende Professor später aus der Datenbasis gelöscht, Prüfungen ohne Prüferinformation im System enthalten sind.
  - Es ist nicht möglich, Vorlesungen, die abgeprüft wurden, zu löschen. Dies verhindert der für *VorlNr* definierte Fremdschlüssel-Constraint.
  - Aufgrund des **on delete cascade** Constraints können Studenten, die Prüfungen abgelegt haben, aus dem System entfernt werden. Die entsprechenden Prüfungsleistungen werden dann ebenfalls mit ausgetragen.
  - Eine Note ist größer oder gleich 0,7 und kleiner oder gleich 5,0.

Gruppenaufgabe 3

1. ```
   INSERT INTO Assistenten
   VALUES (9999, 'NeuerAssistent', 'AI', 1111);
   ```
   Fehler, da referenzierter Professor nicht existiert.

2. ```
   INSERT INTO Professor
   VALUES (1111, 'NeuerProf', 'C4', 1111);

   INSERT INTO Assistenten
   VALUES (9999, 'NeuerAssistent', 'AI', 1111);
   ```

3. ```
   DELETE FROM Professoren WHERE PersNr = 1111;
   DELETE FROM Assistenten WHERE PersNr = 9999;
   ```

4. ```
   DROP TABLE Vorlesungen;
   ```
   Fehler wegen Foreign Key Constraint: Tabelle `prüfen` referenziert Primärschlüssel von `Vorlesungen` mit dem default FK-Cosntraint, weshalb diese Tabelle nicht gelöscht werden kann.

5. ```
   DELETE FROM Vorlesungen WHERE VorlNr = 5041;
   ```
   Fehler wegen Foreign Key Constraint: In der Tabelle `pruefen` referenziert eine Zeile die zu löschenden `Vorlesung` mit dem default FK-Constraint, weshalb diese nicht gelöscht werden kann.

6. Anzeigen über SQL Developer: Auswahl der Tabelle – Reiter „Constraints“

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	SYS_C0038520	Check	Note between 0.7 and 5.0
2	SYS_C0038521	Primary Key	(null)
3	SYS_C0038522	Foreign Key	(null)
4	SYS_C0038523	Foreign Key	(null)
5	SYS_C0038524	Foreign Key	(null)

   oder über Zugriff auf *user_constraints*:

   ```
   SELECT *
   FROM user_constraints
   WHERE table_name = ‚PRUEFEN’;
   ```

7. ```
   DROP TABLE pruefen;
   ```
   Tabelle wird gelöscht, da keine andere Tabelle sie referenziert bzw. keine default oder `restrict` Foreign Key Constraint auf diese Tabelle existiert.

8.
```
CREATE TABLE pruefen
(MatrNr INTEGER REFERENCES Studenten ON DELETE CASCADE,
 VorlNr INTEGER REFERENCES Vorlesungen ON DELETE CASCADE,
 PersNr INTEGER REFERENCES Professoren,
 Note NUMERIC(2,1) CHECK (Note between 0.7 and 5.0),
 PRIMARY KEY (MatrNr, VorlNr));
```

   Der default Foreign Key Constraint (kein Löschen einer Vorlesung, solange sie in `pruefen` referenziert wird) wird nun überschrieben. Wird eine Vorlesung aus `Vorlesungen` gelöscht, wird das Löschen in der `pruefen` Tabelle kaskadiert, indem diese Vorlesung referenzierende Zeilen ebenfalls gelöscht werden.

9.
```
DELETE FROM Vorlesungen WHERE VorlNr = 5041;
```
   Die Zeilen, die die Bestellung referenzieren, werden ebenfalls gelöscht.

Hausaufgabe 1

- Löschen der Vorlesung Ethik:
  - Auf diese Vorlesung (*VorlNr* 5041) existieren Referenzen in den Tabellen *hören*, *voraussetzen* und *prüfen*.
  - Referenzen in *hören* und *voraussetzen* sind als **on delete cascade** definiert, d.h., bei Löschen der Vorlesung werden die entsprechenden Tupel in diesen Relationen gelöscht.
  - In *prüfen* ist *VorlNr* ein Fremdschlüssel auf *Vorlesungen*. Es ist kein spezielles Verhalten definiert. Das Default-Verhalten der Datenbanksysteme verhindert jedoch das Entstehen ungültiger Referenzen (sog. *dangling references*) und blockiert die Löschoperation.
    Dieses Verhalten kann man mit dem Constraint **on delete no action** auch explizit erzwingen.
- Erstes Einfügen eines prüfen-Tupels:
  Das Einfügen scheitert, da es keinen Professor mit PersNr 2138 gibt, die Tabelle *prüfen* jedoch mit dem Constraint **references** Professoren **on delete ...** spezifiziert ist. Bei **insert**-Anweisungen wird also sichergestellt, dass der Eintrag existiert.
- Zweites Einfügen eines prüfen-Tupels:
  prüfen hat den Primärschlüssel (*MatrNr*, *VorlNr*). Da bereits ein Tupel mit (28106, 5001) als Primärschlüsselausprägung in *prüfen* existiert, scheitert auch diese **insert**-Anweisung.

- Löschen der Tabelle Studenten:
  Der **drop table**-Befehl scheitert, da in den Tabellendefinitionen von *prüfen* und *hören* **references**-Constraints definiert wurden, die ein Löschen verhindern. Dies ist auch dann der Fall, wenn die Constraints mit **on delete cascade** oder **on delete set null** definiert wurden; selbst dann, wenn die Tabelle *Studenten* leer ist.
  Man könnte also alle Studenten in der Tabelle *Studenten* löschen; nicht jedoch die Tabelle selbst.

Hausaufgabe 2

Anweisung	Liegt ein Verstoß vor, oder kann die Zeile eingefügt werden? • Verstoß (Name der Bedingung) • OK
a. INSERT INTO R VALUES (1, 'magenta', 45, 512)	Verstoß CR3
b. INSERT INTO R VALUES (4, 'orange', 32, 256)	Verstoß CR4
c. INSERT INTO R VALUES (11, 'magenta', 14, 8)	OK
d. INSERT INTO R VALUES (10, 'gruen', 15, 4)	Verstoß CR2
e. INSERT INTO S VALUES (1, 2, 3, 12, 256)	OK
f. INSERT INTO S VALUES (3, 4, 3, 1, 256)	Verstoß CS3
g. INSERT INTO S VALUES (9, 12, 3, 14, 32)	Verstoß CS1