	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

1 Präsenzaufgabe: Schemadefinition

Gegeben sei folgendes Relationenschema:

Buch(ISBN, Titel, Erscheinungsjahr, Seitenzahl, Verlag → Verlag.VID)

Verlag(VID, Name, Leiter → Mensch.PID)

Mensch(PID, Vorname, Nachname, Lieblingsbuch → Buch.ISBN)

schreibt(Autor → Mensch.PID, Buch → Buch.ISBN)

begutachtet(Lektor → Mensch.PID, Buch → Buch.ISBN)

Um die Konsistenz der Daten sicherzustellen, sollen zudem folgende Integritätsbedingungen gelten:

IB1: Die Seitenzahl eines Buches muss zwischen 0 und 4.000 liegen.

IB2: Der Nachname eines Menschen ist eindeutig.

IB3: Alle Felder bis auf Mensch.Lieblingsbuch sind Pflichtfelder.


Definieren Sie das angegebene Schema mithilfe von Befehlen der SQL DDL (Data Definition Language). Zur Prüfung Ihrer Lösung führen Sie die DDL-Befehle bitte in MariaDB aus.

Hinweis: Legen Sie Fremdschlüssel bitte als benannte Constraints an.

Lösungsvorschlag:

```
CREATE TABLE Mensch(
  PID          int          PRIMARY KEY,
  Vorname      varchar(50) NOT NULL,
  Nachname     varchar(50) UNIQUE NOT NULL,
  Lieblingsbuch char(13)
);
```

```
CREATE TABLE Verlag(
  VID          int          PRIMARY KEY,
```

	Lehrveranstaltung	Grundlagen von Datenbanken WS 2020/21		
	Aufgabenzettel	5 (Lösungsvorschläge)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021

```

    Name          varchar(50) NOT NULL,
    Leiter int     NOT NULL,
CONSTRAINT fk_leiter FOREIGN KEY (Leiter) REFERENCES Mensch (PID)
);


CREATE TABLE Buch(
    ISBN          char(13) PRIMARY KEY,
    Titel         varchar(50) NOT NULL,
    Erscheinungsjahr int     NOT NULL,
    Seitenzahl    int     NOT NULL CHECK(Seitenzahl > 0 AND Seitenzahl < 4000),
    Verlag        int NOT NULL,
CONSTRAINT fk_verlag FOREIGN KEY (Verlag) REFERENCES Verlag (VID)
);

ALTER TABLE Mensch ADD CONSTRAINT fk_pers_lbuch
FOREIGN KEY (Lieblingsbuch) REFERENCES Buch (ISBN) ON DELETE SET NULL;

CREATE TABLE Schreibt(
    Autor          int,
    Buch           char(13),
CONSTRAINT pk_schreibt PRIMARY KEY (Autor, Buch),
CONSTRAINT fk_schreibt_autor FOREIGN KEY (Autor) REFERENCES Mensch (PID),
CONSTRAINT fk_schreibt_buch FOREIGN KEY (Buch) REFERENCES Buch (ISBN)
);

CREATE TABLE Begutachtet(
    Lektor         int,
    Buch           char(13),
CONSTRAINT pk_begutachtet PRIMARY KEY (Lektor, Buch),
CONSTRAINT fk_begutachtet_lektor FOREIGN KEY (Lektor) REFERENCES Mensch (PID),
CONSTRAINT fk_begutachtet_buch FOREIGN KEY (Buch) REFERENCES Buch (ISBN)
);

```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

2 Präsenzaufgabe: Referentielle Aktionen

a) Welche Anforderung erfüllt ein (bzgl. der referentiellen Aktionen) sicheres Schema?

Lösungsvorschlag:

Bei einem sicheren Schema ist das Ergebnis einer Änderungsoperation unabhängig von der Reihenfolge, in der die referentiellen Aktionen ausgeführt werden.

b) Gegeben sei folgende Datendefinition:

```

CREATE TABLE Websites(
    WID            INT            PRIMARY KEY,
    URI            VARCHAR(200) NOT NULL,
    Titel          VARCHAR(50)  NOT NULL,
    EingestelltVon INT            NOT NULL
);


CREATE TABLE BenutzerIn(
    UID            INT            PRIMARY KEY,
    Name           VARCHAR(20)  NOT NULL,
    Homepage       INT            REFERENCES Websites (WID) ON DELETE SET NULL
);

ALTER TABLE Websites ADD FOREIGN KEY (EingestelltVon)
    REFERENCES BenutzerIn (UID) ON DELETE RESTRICT;

CREATE TABLE Rubriken(
    RID            INT            PRIMARY KEY,
    Bezeichnung    VARCHAR(30)  NOT NULL,
    VerwalterIn    INT            NOT NULL
                                REFERENCES BenutzerIn (UID) ON DELETE CASCADE
);

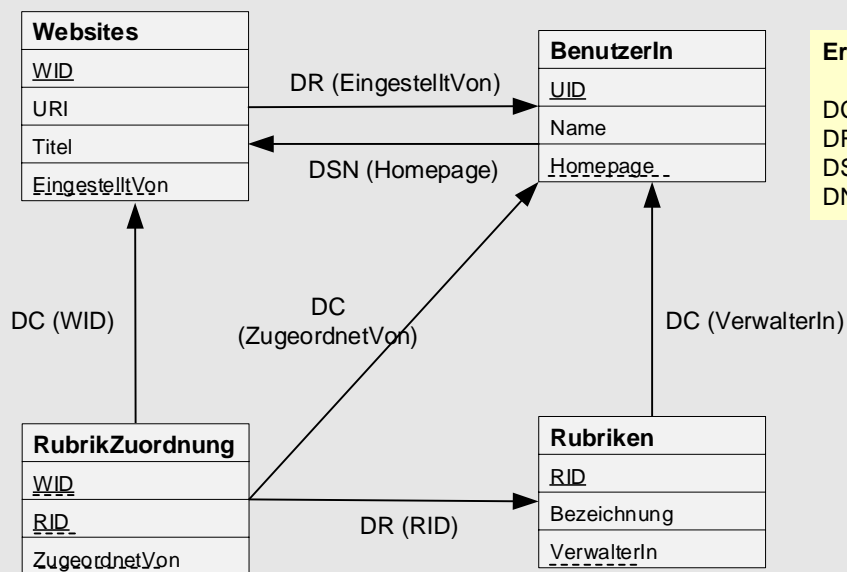
CREATE TABLE RubrikZuordnung(
    WID            INT            REFERENCES Websites (WID) ON DELETE CASCADE,
    RID            INT            REFERENCES Rubriken (RID) ON DELETE RESTRICT,
    ZugeordnetVon INT NOT NULL REFERENCES BenutzerIn (UID) ON DELETE CASCADE,
    PRIMARY KEY (WID, RID)
);

```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	


Zeichnen Sie den zugehörigen Referenzgraphen und beschriften Sie alle Kanten mit den entsprechenden referentiellen Aktionen.

Lösungsvorschlag:



Erklärung:

DC = ON DELETE CASCADE
 DR = ON DELETE RESTRICT
 DSN = ON DELETE SET NULL
 DNA = ON DELETE NO ACTION

	Lehrveranstaltung	Grundlagen von Datenbanken WS 2020/21		
	Aufgabenzettel	5 (Lösungsvorschläge)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021

c) Erläutern Sie, in welchen Fällen das vorliegende Schema unsicher bezüglich referenzieller Aktionen ist.

Lösungsvorschlag:

Es können reihenfolgeabhängige Ergebnisse auftreten, wenn eine BenutzerIn gelöscht wird, die zwar keine Websites erstellt hat, die aber Rubriken verwaltet, denen nur diese BenutzerIn Websites zugeordnet hat. Die Löschoperation kann auf zwei Pfaden von 'Benutzer' zu 'RubrikZuordnung' kaskadieren:

- Pfad 1: BenutzerIn → Rubriken → RubrikZuordnung
- Pfad 2: BenutzerIn → RubrikZuordnung

Es gibt zwei mögliche Auswertungsreihenfolgen:


- Pfad 1 zuerst:
 - Alle von der gelöschten BenutzerIn verwalteten 'Rubriken'-Tupel werden gelöscht.
 - Die Löschung der zugehörigen 'RubrikZuordnung'-Tupel wird zurückgewiesen.
 - ⇒ Der Löschvorgang ist nicht erfolgreich.
- Pfad 2 zuerst:
 - Die 'RubrikZuordnung'-Tupel der BenutzerIn werden gelöscht.
 - Alle von der gelöschten BenutzerIn verwalteten 'Rubriken'-Tupel werden gelöscht.
 - ⇒ Der Löschvorgang ist erfolgreich.

d) Beschreiben Sie, welche Vorkehrungen (Änderungen der referenziellen Aktionen) getroffen werden könnten, um diesen Missstand zu beheben.

Lösungsvorschlag:

Das Problem kann durch veränderte Referenzdefinitionen behoben werden. Dabei gibt es mehrere Möglichkeiten. Drei davon werden hier kurz skizziert.

- Wird die Referenz RubrikZuordnung.RID → Rubriken.RID als ON DELETE NO ACTION deklariert, können keine reihenfolgeabhängigen Ergebnisse mehr auftreten. Effektiv wird zunächst die referentielle Aktion ON DELETE CASCADE für die Referenz RubrikZuordnung.ZugeordnetVon → BenutzerIn.UID ausgeführt, bevor schließlich die Referenz RubrikZuordnung.RID → Rubriken.RID überprüft wird.
- Wird die Referenz RubrikZuordnung.RID → Rubriken.RID als ON DELETE CASCADE deklariert, können keine reihenfolgeabhängigen Ergebnisse mehr auftreten. Es werden nun automatisch alle Websites aus den Rubriken entfernt, die von der zu löschenden BenutzerIn verwaltet werden.

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

- Wird die Referenz Rubriken.Verwalter → BenutzerIn.UID als ON DELETE RESTRICT deklariert, können keine reihenfolgeabhängigen Ergebnisse mehr auftreten. Es können nun jedoch keine BenutzerInnen mehr gelöscht werden, die noch Rubriken verwalten.

3 Übungsaufgabe: SQL

[10 P.]

Fahrzeug(SerienNr, *Modell*, *Hersteller* → *Hersteller.HNr*, *Fabrik* → *Fabrik.FNr*)

Person(PNr, *Vorname*, *Nachname*, *Age*, *Lieblingsautomarke* → *Hersteller.HNr*)

FZSchein(Kennzeichen, *Anmeldedatum*, *Fahrzeug* → *Fahrzeug.SerienNr*, *Halter* → *Person.PNr*)

Hersteller(HNr, *Name*, *Firmensitz*, *GewinnInEuro*)

Fabrik(FNr, *Standort*, *Leiter* → *Person.PNr*, *Firma* → *Hersteller.HNr*, *AutosProJahr*)


Hinweis: *FZSchein* steht für *Fahrzeugschein*.

Formulieren Sie entsprechende SQL-Anweisungen für die in den nachfolgenden Teilaufgaben angeführten natürlichsprachlich formulierten Mengenbeschreibungen. **Verwenden Sie den in der Vorlesung verwendeten SQL-Standard.** Das SQL-Schlüsselwort JOIN darf dabei nur zur Formulierung eines äusseren Verbundes verwendet werden.

- a) Die Nachnamen (ohne Duplikate) aller Personen, die noch nie Halter eines Autos waren, welches von einer Firma mit dem Namen 'TWP' hergestellt wurde.

Lösungsvorschlag:

```
SELECT DISTINCT p.Nachname
FROM Person p
WHERE p.PNr NOT IN
  (SELECT s.Halter
   FROM FZSchein s, Fahrzeug f, Hersteller h
```

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

```

WHERE s.Fahrzeug = f.SerienNr
AND f.Hersteller = h.HNr
AND h.Name = 'TWP');

```

Oder

```

SELECT DISTINCT p.Nachname
FROM Person p
WHERE NOT EXISTS
(SELECT *
FROM FZSchein s, Fahrzeug f, Hersteller h
WHERE s.Fahrzeug = f.SerienNr
AND f.Hersteller = h.HNr
AND h.Name = 'TWP'
AND s.Halter = p.PNr);

```

b) Die Standorte (ohne Duplikate) an denen mindestens zwei Firmen eine Fabrik haben.

Lösungsvorschlag:

```

SELECT DISTINCT f.Standort
FROM Fabrik f
GROUP BY f.Standort
HAVING COUNT(DISTINCT f.Firma) > 1;

```

c) Die Anzahl Fabriken für alle Firmen (HNr und Name) deren Name mit einem 'T' beginnt.


Lösungsvorschlag:

```

SELECT h.HNr, h.Name, COUNT(*)
FROM Hersteller h, Fabrik f
WHERE f.Firma = h.HNr
AND h.Name LIKE 'T%'
GROUP BY h.HNr, h.Name;

```

d) Die PNr und Namen (Vor- und Nachname) aller Personen, sowie die Anzahl an Fabriken, die diese leiten.
Hinweis: Das Ergebnis soll auch solche Personen beinhalten, die keine Fabrik leiten.

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

Lösungsvorschlag:

```
SELECT p.PNr, p.Vorname, p.Nachname, COUNT(f.FNr)
  FROM Person p LEFT OUTER JOIN Fabrik f ON p.PNr = f.Leiter
 GROUP BY p.PNr, p.Vorname, p.Nachname;
```

Hier ist kein DISTINCT im COUNT nötig, da wir nur die Nicht-Nullwerte zählen wollen und hier auch keine Fabrik 2 mal pro Person vorkommen kann.

- e) Die PNr und Namen (Vor- und Nachname) aller Personen, bei denen unbekannt ist wie alt sie sind.

Lösungsvorschlag:

```
SELECT p.PNr, p.Vorname, p.Nachname
  FROM Person p
 WHERE p.Age is Null;
```

- f) Die letzte Anfrage produziert ein leeres Ergebnis. Versuchen sie ein passendes Tupel hinzuzufügen und beschreiben Sie warum das nicht so einfach möglich ist.

Lösungsvorschlag:

Die passende Anweisung


```
INSERT INTO Person (PNr, Vorname, Nachname, Age, Lieblingsautmarke)
  VALUES (14, 'Hans', 'Knopf', null, 1);
```

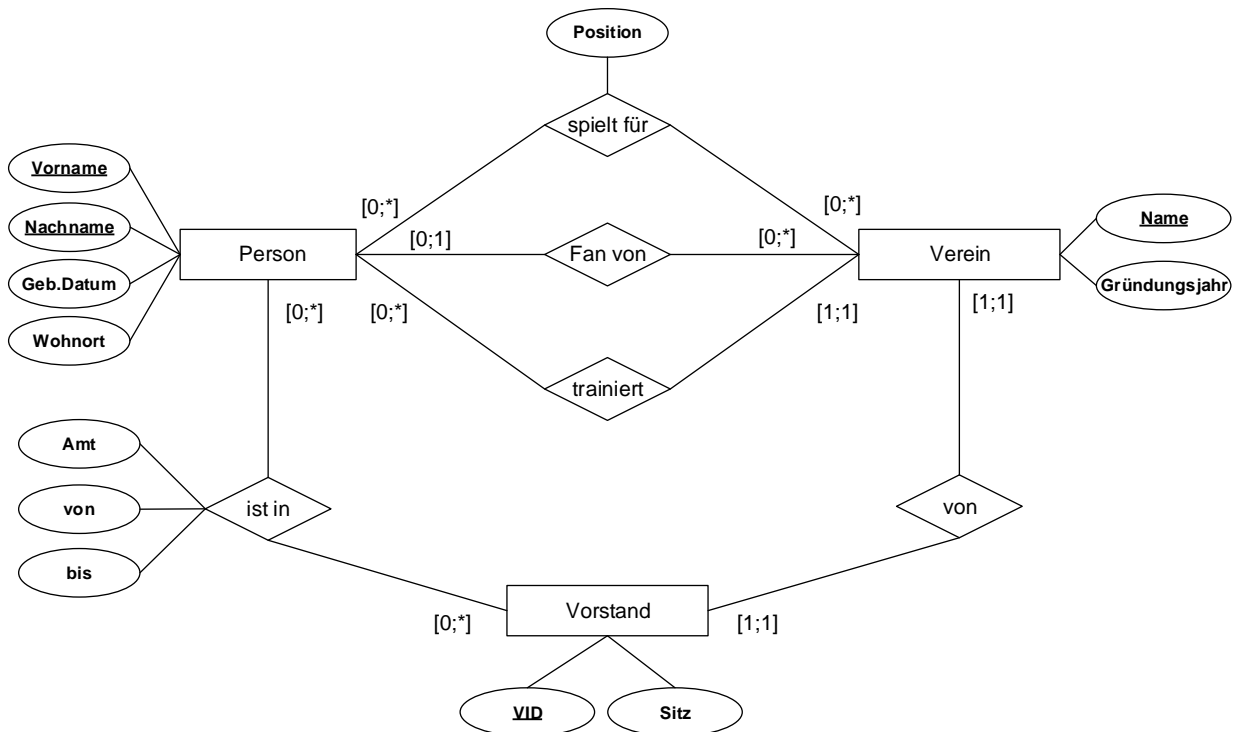
funktioniert nicht, da Age mit NOT NULL definiert wurde.

4 Übungsaufgabe: Schemadefinition

[12 P.]


Geben Sie die SQL-DDL-Anweisungen an, die notwendig sind, um das DB-Schema für das nachfolgend dargestellte Entity-Relationship-Diagramm zu erstellen. Wählen Sie dabei geeignete SQL-Standard-Datentypen. Beachten Sie, dass die Kardinalitätsrestriktionen durch geeignete Constraints exakt abzubilden sind. Weiterhin ist bei 1:1-Beziehungen die Symmetrie sicherzustellen (Tipp: Fremdschlüssel in beiden Relationen). Testen Sie die SQL-Ausdrücke auf der Übungsdatenbank.

	Lehrveranstaltung	Grundlagen von Datenbanken WS 2020/21		
	Aufgabenzettel	5 (Lösungsvorschläge)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021



Weiterhin gelten folgende Integritätsbedingungen:

- **IB1**: Der Sitz eines Vorstandes ist eindeutig.
- **IB2**: Gründungsjahr, Geburtsdatum und das Datum, bis zu welchem eine Person ein Amt eines Vorstandes belegt, sind optional. Alle anderen Attribute sind verpflichtend anzugeben.
- **IB3**: Das Geburtsdatum einer Person muss (sofern angegeben) kleiner als das Datum '31.12.2020' sein.

	Lehrveranstaltung	Grundlagen von Datenbanken WS 2020/21		
	Aufgabenzettel	5 (Lösungsvorschläge)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021

Lösungsvorschlag:

```


CREATE TABLE Person(
  Vorname varchar(50) NOT NULL,
  Nachname varchar(50) NOT NULL,
  Geburtsdatum date CHECK(Geburtsdatum < '31.12.2020'),
  Wohnort varchar(50) NOT NULL,
  Lieblingsverein varchar(50),
  CONSTRAINT pk_person PRIMARY KEY (Vorname, Nachname)
);

CREATE TABLE Verein(
  Name varchar(50) PRIMARY KEY NOT NULL,
  Gruendungsjahr date,
  TrainerInVorname varchar(50) NOT NULL,
  TrainerInNachname varchar(50) NOT NULL,
  Vorstand int UNIQUE NOT NULL,
  CONSTRAINT fk_verein_trainer FOREIGN KEY (TrainerInVorname,TrainerInNachname)
  REFERENCES Person (Vorname,Nachname)
);

CREATE TABLE spielt_fuer(
  Vorname varchar(50) NOT NULL,
  Nachname varchar(50) NOT NULL,
  Verein varchar(50) NOT NULL,
  Position varchar(50) NOT NULL,
  CONSTRAINT pk_spiel_fuer PRIMARY KEY (Vorname, Nachname, Verein),
  CONSTRAINT fk_spiel_fuer_pers FOREIGN KEY (Vorname, Nachname)
  REFERENCES Person (Vorname, Nachname),
  CONSTRAINT fk_spiel_fuer_verein FOREIGN KEY (Verein) REFERENCES Verein(Name)
);

CREATE TABLE Vorstand(
  VID int PRIMARY KEY NOT NULL,
  Sitz varchar(50) UNIQUE NOT NULL,
  CONSTRAINT fk_vorstand_verein FOREIGN KEY (VID) REFERENCES Verein (Vorstand)
);

```


	Lehrveranstaltung	Grundlagen von Datenbanken WS 2020/21		
	Aufgabenzettel	5 (Lösungsvorschläge)		
	Gesamtpunktzahl	30		
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021

```
CREATE TABLE ist_in_Vorstand(
  Vorname varchar(50) NOT NULL,
  Nachname varchar(50) NOT NULL,
  Vorstand int NOT NULL,
  Amt varchar(50) NOT NULL,
  Von date NOT NULL,
  Bis date,
  CONSTRAINT pk_istinvorstand PRIMARY KEY (Vorname, Nachname, Vorstand),
  CONSTRAINT fk_istinvorstand_pers FOREIGN KEY (Vorname, Nachname)
  REFERENCES Person (Vorname, Nachname),
  CONSTRAINT fk_istinvorstand_verein FOREIGN KEY (Vorstand) REFERENCES Vorstand(VID)
);
```

```
ALTER TABLE Person
ADD CONSTRAINT fk_person_lv FOREIGN KEY (Lieblingsverein) REFERENCES Verein(Name);
```

```
ALTER TABLE Verein
ADD CONSTRAINT fk_verein_vorstand FOREIGN KEY (Vorstand) REFERENCES Vorstand (VID);
```

Bei den beiden ALTER TABLE Statements darf jeweils noch INITIALLY IMMEDIATE DEFERRABLE stehen. Diese Klausel kennt MariaDB allerdings nicht.

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

5 Übungsaufgabe: Referentielle Aktionen

[8 P.]

Betrachten Sie das folgende Datenbankschema:

Person(Vorname, Nachname, *DOB*, *Wohnort*, Lieblingsfilm → *Film.FID*)

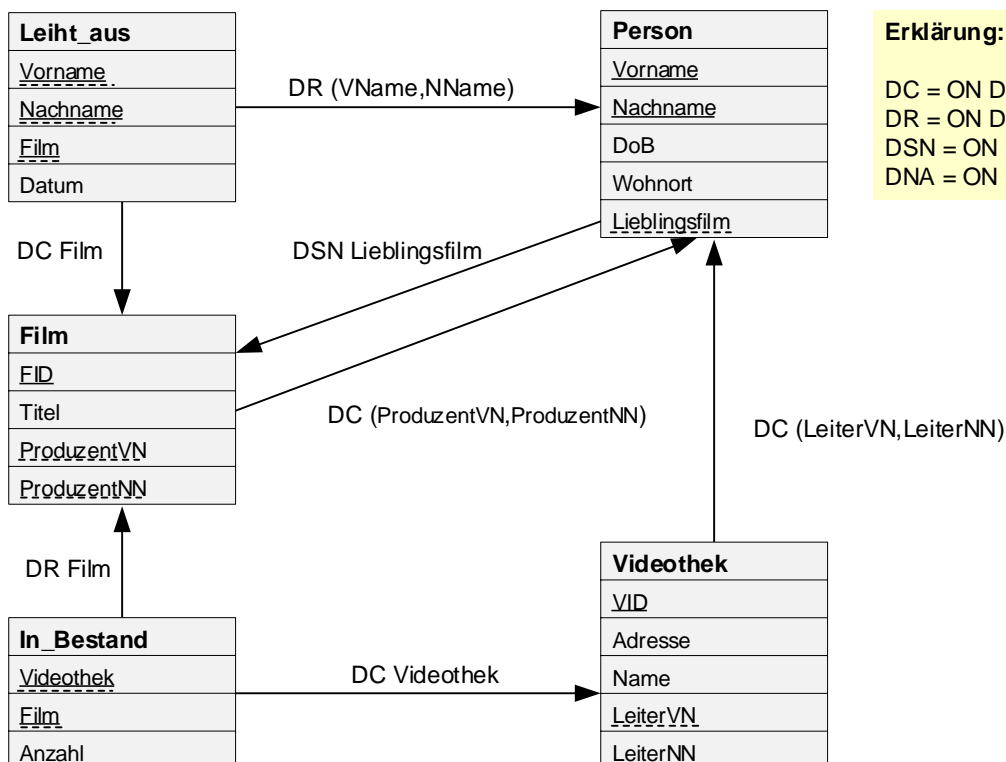
Film(FID, *Titel*, (*ProduzentVN*, *ProduzentNN*) → (*Person.Vorname*, *Person.Nachname*))


Videothek(VID, *Adresse*, *Name*, (*LeiterVN*, *LeiterNN*) → (*Person.Vorname*, *Person.Nachname*))

in_Bestand(*Videothek* → *Videothek.VID*, *Film* → *Film.FID*, *Anzahl*)

leiht_aus((*Vorname*, *Nachname*) → (*Person.Vorname*, *Person.Nachname*), *Film* → *Film.FID*, *Datum*)

Der SQL-Standard erlaubt die Definition von referentiellen Aktionen, um Verletzungen der referentiellen Integrität zu vermeiden. Der unten abgebildete Referenzgraph zeigt die im gegebenen Datenbankschema geltenden referentiellen Aktionen.



	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

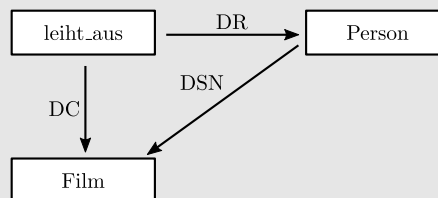
Handelt es sich im vorliegenden Fall um ein sicheres Schema?

Sollte dies nicht der Fall sein, diskutieren Sie alle Szenarien, bei denen reihenfolgeabhängige Ergebnisse auftreten können.

Lösungsvorschlag:

Reihenfolgeabhängige Ergebnisse können nur dann auftreten, wenn durch das Löschen eines Eintrages in einer Tabelle mehrere referentielle Aktionen ausgelöst werden, die zur gleichen Zieltabelle führen. Dies wiederum ist nur möglich, wenn mehrere Pfade von Fremdschlüsselreferenzen zwischen zwei Tabellen existieren. In unserem Beispiel ist dies in vier Fällen gegeben:

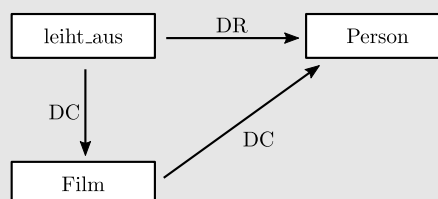
- **Fall 1 (Löschen eines Film-Eintrags F , betrachtete Zieltabelle `leiht_aus`): sicher**



- Pfad 1: Film \rightarrow leiht_aus
- Pfad 2: Film \rightarrow Person \rightarrow leiht_aus

Unabhängig von der Ausführungsreihenfolge werden alle `leiht_aus`-Einträge gelöscht, die F referenzieren (Pfad 1). Außerdem wird der Lieblingsfilm einer Person auf NULL gesetzt, wenn es sich dabei um F handelt (Pfad 2).


- **Fall 2 (Löschen eines Person-Eintrags P , betrachtete Zieltabelle `leiht_aus`): nicht sicher**



- Pfad 1: Person \rightarrow leiht_aus
- Pfad 2: Person \rightarrow Film \rightarrow leiht_aus

Das Ergebnis ist reihenfolgeabhängig, wenn die zu löschende Person P nur selbstproduzierte Filme ausgeliehen hat.

Zuerst Auswertung von Pfad 1:

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

- Die von P getätigten Ausleihen können *nicht* gelöscht werden.

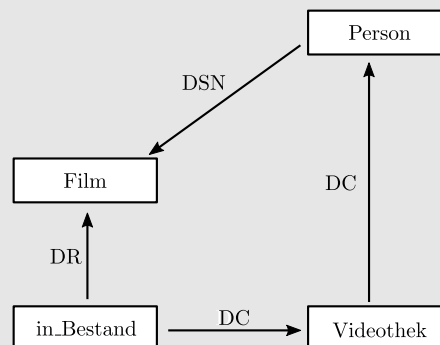
⇒ Der Löschvorgang wird abgebrochen.

Zuerst Auswertung von Pfad 2:

- Alle Filme, die von P produziert wurden, werden ebenfalls gelöscht.
- Alle Ausleihen dieser Filme werden gelöscht.
- P kann nun gelöscht werden, da für P keine Ausleihen mehr verzeichnet sind.


⇒ Der Löschvorgang ist erfolgreich.

- **Fall 3 (Löschen eines Film-Eintrags F , betrachtete Zieltabelle in_Bestand): sicher**

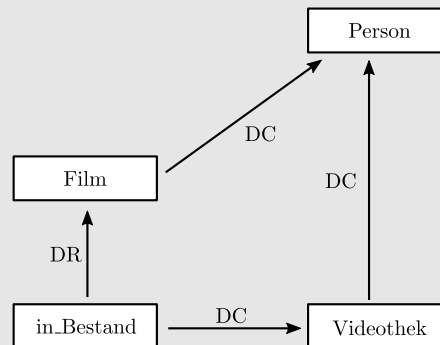


- Pfad 1: Film → in_Bestand
- Pfad 2: Film → Person → Videothek → in_Bestand

Sollte der zu löschende Film F bereits in den Bestand einer Videothek aufgenommen worden sein, wird die Löschung zurückgesetzt (Pfad 1). Ist F der Lieblingsfilm einer Person, wird der entsprechende Wert bei dieser Person auf NULL gesetzt (Pfad 2). Es liegen keine Reihenfolgeabhängigkeiten vor.

	Lehrveranstaltung	Grundlagen von Datenbanken			WS 2020/21
	Aufgabenzettel	5 (Lösungsvorschläge)			
	Gesamtpunktzahl	30			
	Ausgabe	Mi. 13.01.2021	Abgabe	Fr. 29.01.2021	

- Fall 4 (Löschen eines Person-Eintrags P , betrachtete Zieltabelle in_Bestand): nicht sicher



- Pfad 1: Person \rightarrow Videothek \rightarrow in_Bestand
- Pfad 2: Person \rightarrow Film \rightarrow in_Bestand

Das Ergebnis ist reihenfolgeabhängig, wenn eine Person P gelöscht wird, die Leiter einer Videothek ist und nur Filme produziert hat, die ausschließlich in den eigenen Videotheken vorhanden sind.

Zuerst Auswertung von Pfad 1:

- Die von P produzierten Filme werden gelöscht.
- Die entsprechenden in_Bestand-Einträge können jedoch *nicht* gelöscht werden, da diese noch in dem Bestand einer Videothek enthalten sind.

\Rightarrow Der Löschvorgang wird abgebrochen.

Zuerst Auswertung von Pfad 2:

- Die von P geleitete Videothek wird gelöscht.
- Alle in_Bestand-Einträge der betreffenden Videothek werden gelöscht.
- Die von P produzierten Filme werden gelöscht. Dies ist möglich, da diese im Bestand keiner Videothek enthalten sind.

\Rightarrow Der Löschvorgang ist erfolgreich.