Datenmanagement II

Prof.Wolfgang Kowarschick

Sommersemester 2012

Interaktive Medien IV

Dokumentation

Onlineshop-Datenbank

Autorin: Daniela Huber

Matrikelnr: 924323

Inhaltsverzeichnis

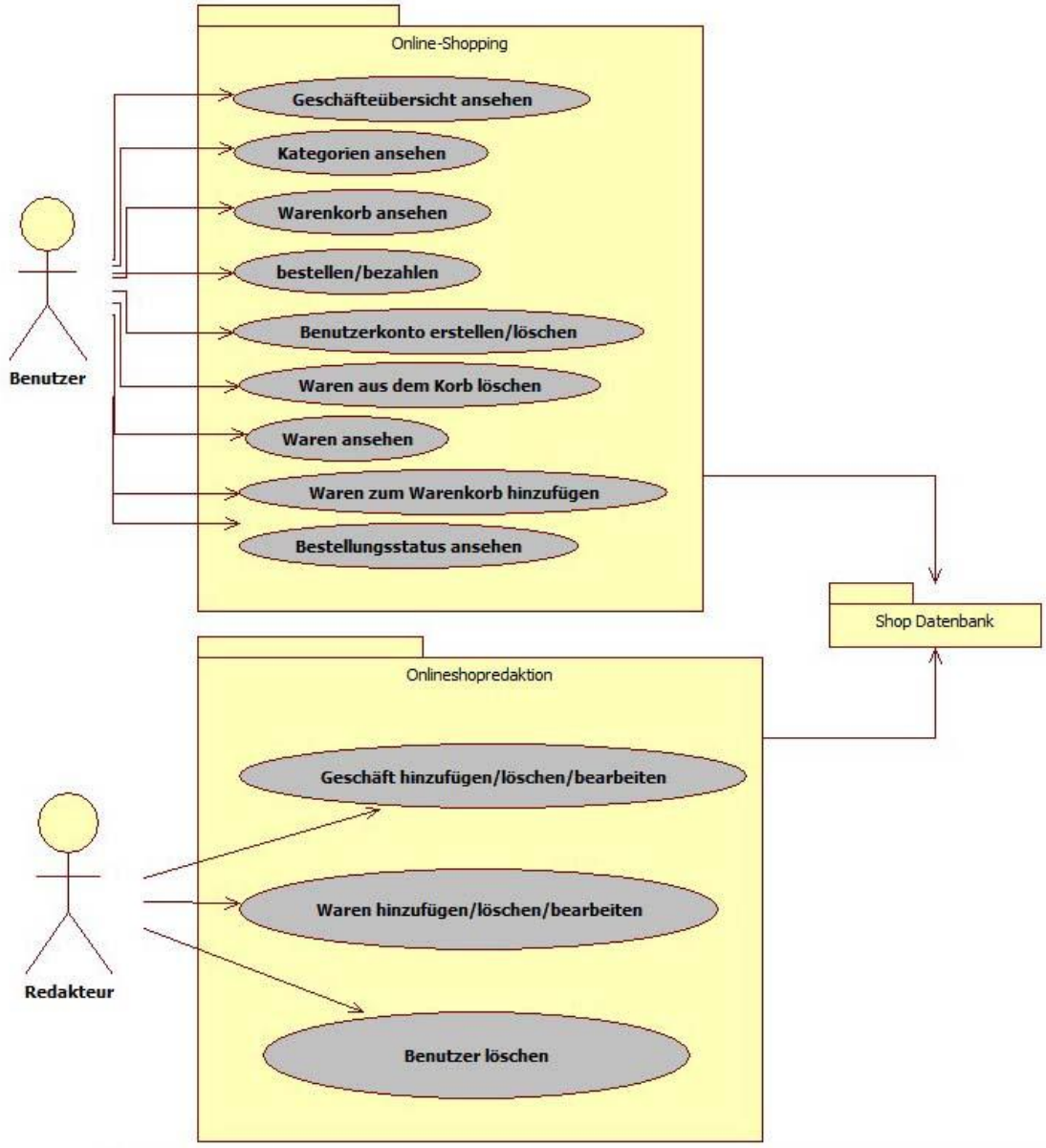
Kurzbeschreibung

Die Datenbank des Onlineshops speichert Daten zu den einzelnen Geschäften, Waren und Benutzern. Auch Bestellungen, sowie Bestellstatus, Datum, etc. werden in der Datenbank erfasst.

Im Onlineshop soll ein Benutzer die Möglichkeiten haben, die Geschäfte, Kategorien und Waren anzusehen. Außerdem soll jeder Benutzer einen Warenkorb bekommen, aus dem er Waren löschen oder in den er Waren hinzufügen kann. Außerdem kann er den Status seiner Bestellung verfolgen und kann sich ein Profil erstellen.

Der Betreiber des Onlineshops kann Benutzer, Waren, Kategorien und Geschäfte aus der Datenbank löschen oder hinzufügen. Außerdem kann er sich Übersichten zu den Benutzern, Bestellungen und wichtigen Informationen erstellen.

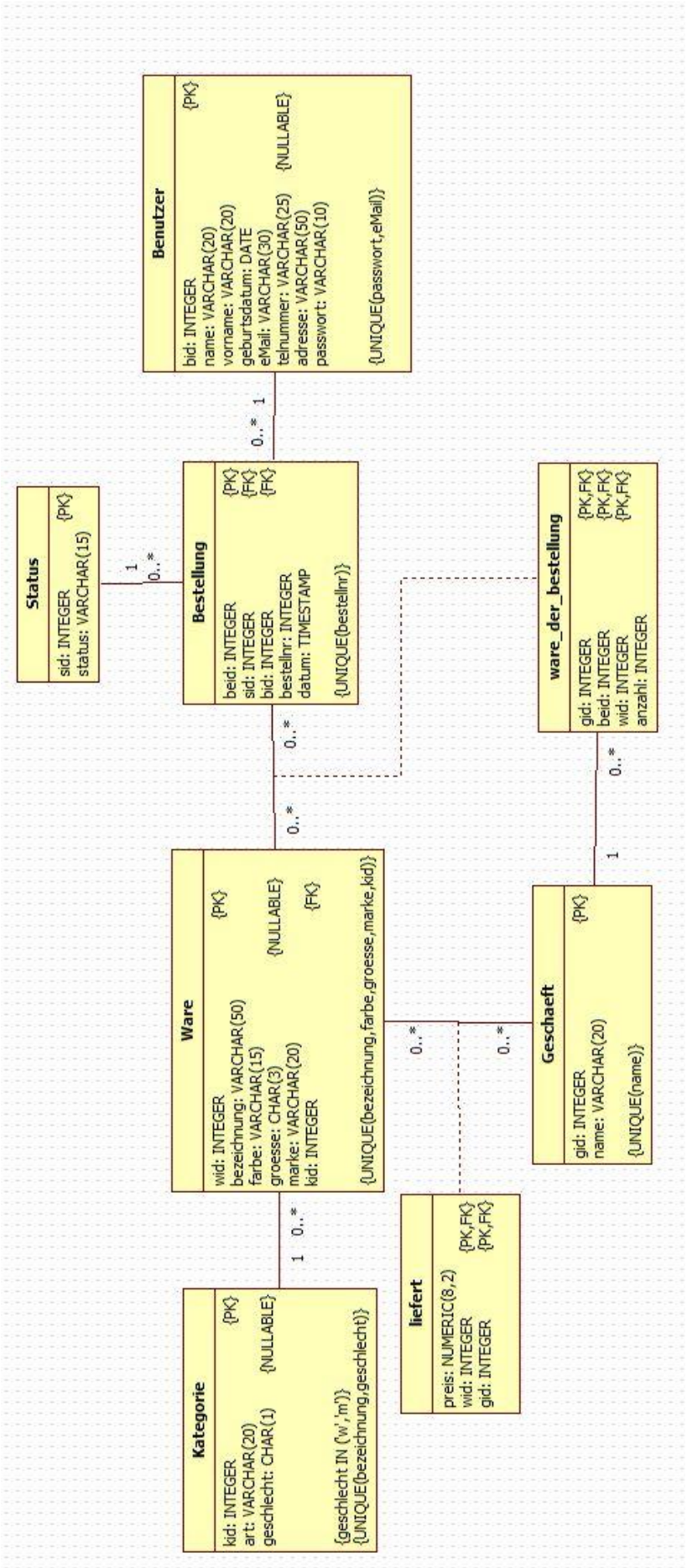
Use-Cases-Diagramm und Beschreibung



Ein Benutzer soll sich eine Gesamtübersicht aller Geschäfte, Kategorien und Waren ansehen können. Außerdem muss er zunächst einmal ein Benutzerkonto anlegen, um eine Bestellung aufnehmen zu können. In seine Bestellung/Warenkorb kann der Benutzer Waren hinzufügen und diese auch wieder löschen. Nach dem Absenden der Bestellung, kann er über den Bestellstatus verfolgen in welchem Prozessabschnitt seine Bestellung gerade steckt. Außerdem hat der Benutzer die Möglichkeit seine Bestellung sofort oder erst später zu bezahlen.

Der Redakteur des Onlineshops kann Geschäfte, Waren und Kategorien hinzufügen, löschen und bearbeiten. Außerdem steht es ihm immer zu, einen Benutzer zu löschen. Er kann sich darüber hinaus Übersichten zu Informationen über den Benutzer, Bestellungen oder Geschäfte anzeigen lassen.

ER-Diagramm und Beschreibung



Es gibt acht Tabellen.
Die Tabelle Ware enthält alle Waren, die es gibt. Dabei wird jeder Ware eine Id, eine Bezeichnung, eine Farbe, eine Größe, eine Marke und eine Kategorie zugewiesen. Die Einzigartigkeit einer Ware machen dabei alle Angaben aus. Eine Ware darf dabei nur eine Kategorie besitzen, aber in null oder mehr Bestellungen vorhanden sein. Außerdem kann eine Ware in null oder unendlich Geschäften angeboten werden.

Die Tabelle Kategorie enthält alle Kategorien mit Id, Art und Geschlecht. Dabei kann das Geschlecht auch NULL, also neutral sein (z.B. Accessoires). UNIQUE ist die Kategorie durch die Bezeichnung und das Geschlecht. Eine Kategorie kann null oder mehreren Waren zugeordnet sein.

Die Tabelle Geschäft enthält alle vorhandenen Geschäfte mit ID und Geschäftsnamen. Dabei bestimmt der Name die Einzigartigkeit des Geschäfts. Ein Geschäft kann logischerweise null oder mehrere Waren anbieten und zu null oder mehreren „Warenkörben“ gehören.

Zwischen den Tabellen Ware und Geschäft besteht eine Beziehungstabelle „liefert“. Diese ordnet einer Ware ein Geschäft zu und speichert den Preis des Geschäfts für diese Ware.

Kommen wir zur Tabelle Bestellung. Für jede Bestellung werden eine ID, ein Status, ein Benutzer, ein Datum und eine Bestellnummer gespeichert. Eine Bestellung kann dabei null oder unendlich viele Waren, aber immer nur einen aktuellen Status besitzen. Außerdem besitzt jede Bestellung genau einen Benutzer, bzw ist ihm zugeordnet. Durch die Bestellnummer kann jede Bestellung eindeutig bestimmt werden.

Die Tabelle Status speichert einen bestimmten Status und seine ID. Dabei kann ein Status zu null oder mehreren Bestellungen gehören.

Die Tabelle ware_der_bestellung ist eine Beziehungstabelle zwischen den Tabellen Bestellung und Ware. Diese Tabelle übernimmt die Funktion eines Warenkorbes, da hier ein Geschäft, eine Bestellung, eine bestimmte Ware und die Warenanzahl gespeichert werden. Dabei besitzt ein Eintrag nur ein Geschäft, da eine Ware nicht gleichzeitig von mehreren Geschäften stammen kann.

Die letzte Tabelle ist die Tabelle Benutzer. Hier werden für jeden Benutzer eine ID, Name und Vorname, das Geburtsdatum, die eMail Adresse, die Telefonnummer, die Adresse und ein individuelles Passwort abgespeichert. Durch das Passwort und die eMail ist der Benutzer eindeutig zu bestimmen. Jeder Benutzer der Tabelle kann null oder mehrere Bestellungen besitzen.

Relationales Schema

```
kategorie
     kid                       INTEGER             NOT NULL
     art                       VARCHAR(20)         NOT NULL
     geschlecht                CHAR(1)

     UNIQUE(art, geschlecht)
     CHECK(geschlecht IN('w','m))

ware
     wid                       INTEGER             NOT NULL
     bezeichnung               VARCHAR(50)         NOT NULL
     farbe                     VARCHAR(15)         NOT NULL
     groesse                   CHAR(3)
     marke                     VARCHAR(20)         NOT NULL
     kid                       INTEGER             NOT NULL => REFERENCES kategorie(kid)

     UNIQUE(bezeichnung, farbem groesse, marke)

geschaeft
     gid                       INTEGER             NOT NULL
     name                      VARCHAR(20)         NOT NULL

     UNIQUE(name)

liefert
     preis                     NUMERIC(8,2)        NOT NULL
     wid                       INTEGER             NOT NULL => REFERENCES ware(wid)
     gid                       INTEGER             NOT NULL => REFERENCES geschaeft(gid)

status
     sid                       INTEGER             NOT NULL
     status                    VARCHAR(15)         NOT NULL

bestellung
     beid                      INTEGER             NOT NULL
     sid                       INTEGER             NOT NULL => REFERENCES status(sid)
     bid                       INTEGER             NOT NULL => REFERENCES benutzer(bid)
     bestellnr                 INTEGER             NOT NULL
     datum                     TIMESTAMP           NOT NULL

     UNIQUE(bestellnr)

ware_der_bestellung
     gid                       INTEGER             NOT NULL => REFERENCES geschaeft(gid)
     beid                      INTEGER             NOT NULL => REFERENCES bestellung(beid)
     wid                       INTEGER             NOT NULL => REFERENCES ware(wid)
     anzahl                    INTEGER             NOT NULL

benutzer
     bid                       INTEGER             NOT NULL
     beid                      INTEGER             NOT NULL => REFERENCES bestellung(beid)
     name                      VARCHAR(20)         NOT NULL
     vorname                   VARCHAR(20)         NOT NULL
     geburtsdatum              DATE                NOT NULL
     eMail                     VARCHAR(30)         NOT NULL
     telnummer                 VARCHAR(25)
     adresse                   VARCHAR(50)         NOT NULL
     passwort                  VARCHAR(10)

     UNIQUE(eMail, passwort)
```

CREATE-TABLE Befehle

```
DROP TABLE IF EXISTS kategorie            CASCADE;
DROP TABLE IF EXISTS ware                 CASCADE;
DROP TABLE IF EXISTS geschaeft            CASCADE;
DROP TABLE IF EXISTS bestellung           CASCADE;
DROP TABLE IF EXISTS benutzer             CASCADE;
DROP TABLE IF EXISTS liefert              CASCADE;
DROP TABLE IF EXISTS ware_der_bestellung  CASCADE;
DROP TABLE IF EXISTS status               CASCADE;

CREATE TABLE kategorie
(
 kid           INTEGER     NOT NULL,
 art           VARCHAR(20) NOT NULL,
 geschlecht    CHAR(1),
 PRIMARY KEY   (kid),
 UNIQUE        (art,geschlecht),
 CHECK         (geschlecht IN ('m', 'w'))
);

CREATE TABLE ware
(
 wid             INTEGER     NOT NULL,
 bezeichnung     VARCHAR(50) NOT NULL,
 farbe           VARCHAR(15) NOT NULL,
 groesse         CHAR(3),
 marke           VARCHAR(20) NOT NULL,
 kid             INTEGER     NOT NULL,
 PRIMARY KEY     (wid),
 FOREIGN KEY     (kid)       REFERENCES kategorie(kid),
 UNIQUE          (bezeichnung,farbe,groesse,marke)
);

CREATE TABLE geschaeft
(
 gid             INTEGER     NOT NULL,
 name            VARCHAR(20) NOT NULL,
 PRIMARY KEY     (gid),
 UNIQUE          (name)
);

CREATE TABLE liefert
(
 Preis           NUMERIC(8,2) NOT NULL,
 wid             INTEGER      NOT NULL,
 gid             INTEGER      NOT NULL,
 PRIMARY KEY     (wid,gid),
 FOREIGN KEY     (wid)        REFERENCES ware(wid),
 FOREIGN KEY     (gid)        REFERENCES geschaeft(gid),
 CHECK (preis > 0)
);
```

```
CREATE TABLE status
(
 Sid              INTEGER     NOT NULL,
 Status           VARCHAR(15) NOT NULL,
 PRIMARY KEY      (sid)
);

CREATE TABLE benutzer
(
 bid              INTEGER     NOT NULL,
 name             VARCHAR(20) NOT NULL,
 vorname          VARCHAR(20) NOT NULL,
 geburtsdatum     DATE        NOT NULL,
 eMail            VARCHAR(30) NOT NULL,
 telnummer        VARCHAR(25),
 adresse          VARCHAR(50) NOT NULL,
 passwort         VARCHAR(10) NOT NULL,
 PRIMARY KEY      (bid),
 UNIQUE           (eMail, passwort)
);

CREATE TABLE bestellung
(
 beid             INTEGER     NOT NULL,
 bid              INTEGER     NOT NULL,
 sid              INTEGER     NOT NULL,
 bestellnr        INTEGER     NOT NULL,
 datum            TIMESTAMP   NOT NULL,
 PRIMARY KEY      (beid),
 FOREIGN KEY      (sid)       REFERENCES status(sid),
 FOREIGN KEY      (bid)       REFERENCES benutzer(bid),
 UNIQUE           (bestellnr)
);

CREATE TABLE ware_der_bestellung
(
 gid         INTEGER     NOT NULL,
 beid        INTEGER     NOT NULL,
 wid         INTEGER     NOT NULL,
 anzahl      INTEGER     NOT NULL,
 PRIMARY KEY (gid,beid,wid),
 FOREIGN KEY (gid)       REFERENCES geschaeft(gid)    ON DELETE CASCADE,
 FOREIGN KEY (beid)      REFERENCES bestellung(beid)  ON DELETE CASCADE,
 FOREIGN KEY (wid)       REFERENCES ware(wid)         ON DELETE CASCADE
)
```

INSERT-Befehle

```
INSERT INTO kategorie(kid, art, geschlecht)
VALUES
  (1,  'Sportkleidung'  ,       NULL),
  (2,  'Unterwäsche'    ,       NULL),
  (3,  'Kinderkleidung' ,       NULL),
  (4,  'Damenkleidung'  ,       'w'),
  (5,  'Herrenkleidung' ,       'm'),
  (6,  'Unterwäsche'    ,       'm'),
  (7,  'Damenkleidung'  ,       NULL)
 ;

INSERT INTO ware (wid, kid, bezeichnung, farbe, groesse, marke)
VALUES
  ( 1,  1,      'Sporthose',    'blau',         '38',   'Adidas'),
  ( 2,  2,      'BH',           'grün',         '80C',  'Triumph'),
  ( 3,  3,      'Pullover',     'schwarz',      '54',   'Buffalo'),
  ( 4,  5,      'Sakko',        'beige',        '52',   'Bruno Banani'),
  ( 5,  4,      'Bluse',        'blau',         '40',   'Esprit'),
  ( 6,  2,      'Sakko',        'grün',         NULL,   'Adidas'),
  ( 7,  3,      'Pullover',     'rot',          '54',   'Buffalo'),
  ( 8,  7,      'Jeans',        'blau',         NULL,   'Esprit'),
  ( 9,  1,      'Sport-BH',     'weiß',         '75A',  'Nike'),
  (10,  3,      'Mütze',        'orange-gelb',  NULL,   'TCM'),
  (11,  3,      'Hose',         'grau',         '174',  'dopodopo'),
  (12,  7,      'Rock',         'weiß-gelb',    '38',   'Esprit'),
  (13,  2,      'BH',           'rot',          '80C',  'Triumph'),
  (14,  6,      'Unterhose',    'weiß',         '34',   'Triumph')
;

INSERT INTO geschaeft (gid, name)
VALUES
  ( 1,  'Sport Conrad'),
  ( 2,  'C&A'),
  ( 3,  'Hugo Boss'),
  ( 4,  'H&M'),
  ( 5,  'Armani'),
  ( 6,  'Tchibo'),
  ( 7,  'Vero Moda'),
  (10,  'New Yorker'),
  (11,  'Esprit'),
  (12,  'Mister&Lady')
;

INSERT INTO liefert(wid,gid,preis)
VALUES
  ( 1,  1,      25.00),
  ( 1,  11,     50.00),
  (12,  7,      30.95),
  ( 2,  4,      25.40),
  ( 2,  5,      27.40),
  (13,  4,      30.35),
  ( 5,  12,     80.00),
  (10,  6,      5.00),
  ( 4,  5,      135.00),
  (13,  2,      17.98),
  ( 8,  11,     60.00),
```

```
  (11,  4,      10.50),
  ( 4,  3,      15.45),
  ( 2,  1,      45.00),
  ( 3,  7,      34.89),
  ( 1,  6,      24.95),
  ( 3,  1,      35.23),
  ( 5,  4,      26.93),
  (10,  4,      15.00),
  ( 1,  4,      55.00),
  ( 5,  1,      27.34),
  ( 9,  2,      15.49),
  ( 1,  10,     30.99),
  ( 3,  2,      45.00),
  ( 2,  2,      75.22),
  (10,  7,      45.45),
  ( 5,  5,      34.28),
  ( 8,  3,      27.99),
  (14,  11,     77.77)
;

INSERT INTO status(sid,status)
VALUES
 (1,    'nichts'),
 (2,    'bezahlt'),
 (3,    'verschickt'),
 (4,    'in Bearbeitung'),
 (5,    'zugestellt'),
 (6,    'storniert')
;

INSERT INTO benutzer (bid, name, vorname, geburtsdatum, eMail, telnummer,
adresse, passwort)
VALUES
  (1, 'Müller',  'Hans',     '1987-02-17', 'hans@gmx.net',        NULL,
'Landsbergerstraße 10 82141 München',  'Herzchen'),

  (2, 'Huber',   'Anne',     '1991-10-05', 'anne@yahoo.com',    '08212938',
'Müllerstraße 3 93872 Schrobenhausen', '29739'),

  (3, 'Hansen',  'Johann',   '1954-12-24', 'hansen@sky10.de',     NULL,
'Marktplatz 5 72038 Land',             'w98wnd'),

  (4, 'Hermann', 'Susanne',  '1991-03-31', 'susi@yahoo.com',
'08987263989','Leostraße 8 92739 Möhring', 'l%pma3'),

  (5, 'Glas',    'Uschi',    '1944-03-02', 'uschi@aol.com',       NULL,
'Müllerstr 10 81241 München',          'uschi44'),

  (6, 'Lahm',    'Philipp',  '1983-11-11', 'philipp.lahm@fcb.de', NULL,
'Am Gärtnerplatz 2 München',           'phil83'),

  (7, 'Kurz',    'Reinhard', '1935-09-22', 'reini@yahoo.com',  '023792873',
'Hänselstraße 72038 Land',             '12345'),

  (8, 'Schweiß', 'Axel',     '1984-10-13', 'Axelschweiss@aol.com',NULL,
'Müllerstraße 4 93872 Schrobenhausen', '2084msl'),

  (9, 'Kirk',    'Captain',  '2063-01-01', 'kirki@stars.com',     NULL,
'Kabine 3 Raumschiff Enterprise',      'star100');
```

```
INSERT INTO bestellung (beid, bid, sid, bestellnr, datum)
VALUES
 (1,    1,    2,    '124',    '2010-09-12'),
 (2,    1,    5,    '926',    '2012-01-01'),
 (3,    3,    2,    '2000',   '2012-09-03'),
 (4,    8,    1,    '1093',   '2012-05-17'),
 (5,    5,    4,    '983',    '2012-04-13'),
 (6,    7,    6,    '2',      '2008-08-02'),
 (7,    2,    4,    '925',    '2012-01-01'),
 (8,    2,    4,    '910',    '2011-11-28'),
 (9,    8,    1,    '1962',   '2012-07-28'),
 (10,   3,    1,    '10828',  '2012-10-02'),
 (11,   5,    3,    '9273',   '2012-08-01')
;

INSERT INTO ware_der_bestellung (gid, beid, wid, anzahl)
VALUES
 ( 1,   1,    5,    3),
 ( 3,   5,    4,    1),
 ( 1,   5,    2,    2),
 ( 6,   5,    1,    1),
 ( 7,   5,    3,    9),
 (12,   2,    5,    2),
 ( 2,   4,    9,    2),
 (10,   3,    1,    1),
 ( 2,   6,    3,    2),
 ( 2,   6,    2,    1),
 ( 7,   6,   10,    3),
 ( 5,  10,    5,    1),
 ( 1,  11,    3,    2),
 (10,   7,    1,    1),
 ( 1,   7,    2,    2),
 ( 7,   7,    3,    1),
 ( 3,   7,    4,    3),
 ( 1,   7,    5,    1),
 ( 3,   7,    8,    1),
 (11,   7,   14,    3)
```

SQL-Abfragen

```
--a) Geben Sie die Vornamen,Namen und Adressen aller Benutzer aus

SELECT DISTINCT vorname || ' ' || name as benutzername, adresse
FROM   benutzer



--b) Welcher Benutzer hat mindestens 2, aber höchstens 4 Teile in einer Bestellung bestellt?

--Selektion des Namen des Benutzers, auf den Kriterien zutreffen
SELECT DISTINCT  vorname || ' ' || name as Benutzer
FROM   (
       --Selektion der Bestellungsid(beid) und der Anzahl der Waren, deren Anzahl in EINER
       --Bestellung zwischen mindestens 2 und höchstens 4 liegt.
       SELECT    beid, SUM(anzahl) as anzahl_der_waren
       FROM      ware_der_bestellung
       --es wird nach Bestellung gruppiert, um die Waren zählen zu können
       GROUP BY  beid
       HAVING    SUM(anzahl) >= 2 AND SUM(anzahl) <=4
       )as dummy JOIN bestellung USING(beid) JOIN benutzer USING (bid)
```

```
--c) Wie viele Benutzer haben in einer Bestellung mindestens 2 Teile aus der Unterwäsche-Kategorie bestellt?

--Zählen der Anzahl der Benutzer(bid), auf die alle Kriterien zutreffen
SELECT COUNT(bid)
FROM   (
       --Selektion der Bestellungsids(beid), die als Kategorieart "Unterwäsche" besitzen
       --und deren Unterwäschestückzahl mehr als 2 beträgt.
       SELECT DISTINCT  beid
       FROM   ware
              JOIN kategorie USING(kid)
              JOIN ware_der_bestellung USING(wid)
       WHERE  art='Unterwäsche'
       AND    anzahl >=2
       )as dummy JOIN bestellung USING (beid)



--d) Bei welchen Benutzern, die in einer Bestellung mindestens ein Teil aus der Kategorie Sportkleidung bestellt
haben, ist die Bestellung noch in Bearbeitung?

--Es werden Name und Vorname der Benutzer selektiert, deren Bestellung in Bearbeitung ist und die mindestens
--ein Teil aus der Kategorie "Sportkleidung" bestellt haben

SELECT  vorname || ' ' || name as benutzer
FROM    ware JOIN kategorie USING(kid)
             JOIN ware_der_bestellung USING(wid)
             JOIN bestellung USING (beid)
             JOIN benutzer USING(bid)
             JOIN status USING(sid)
WHERE   art='Sportkleidung'
AND anzahl >=1
AND status ='in Bearbeitung'
```

```
--e) Welcher Benutzer bestellt in einer Bestellung aus jeder vorhandenen Kategorie?

--Selektion aller Benutzer, die etwas bestellt haben. Dadurch, dass alle Benutzer,
--die NICHT aus jeder Kategorie bestellt haben durch das EXCEPT abgezogen werden, bleiben nur die Benutzer übrig,
--die aus JEDER Kategorie bestellt haben.

SELECT  vorname || ' ' ||name as Benutzer
FROM    (
        SELECT  bid
        FROM    bestellung

        EXCEPT

        --Selektion der Benutzerid(bid)
        SELECT DISTINCT bid
        FROM   (
               --Selektion der bids mit allen Kategorieids(kid) selektiert. So wird zunächst angenommen,
               --dass alle Benutzer aus jeder vorhandenen Kategorie bestellen.
               --Diese Tabelle wird eingegrenzt durch die EXCEPT-Klausel darunter.

               SELECT DISTINCT bid,kid
               FROM   (
                      --Selektion aller Benutzerids(bid), die eine Bestellung aufgegeben haben.
                       SELECT DISTINCT bid
                       FROM   bestellung,ware_der_bestellung
                       )as dummy,ware

               --Ausschließen der wirklich vorhandenden Bestellungen, sodass nur noch diejenigen übrig bleiben,
               --die NICHT vorhanden sind.

               EXCEPT

               SELECT DISTINCT bid,kid
               FROM   bestellung JOIN ware_der_bestellung USING(beid) JOIN ware USING(wid)
               )as dummy
        )as dummy JOIN benutzer USING(bid)
```

```
--f) Geben Sie alle Benutzer mit ihren Bestellungsnummern an, wobei auch die Benutzer ausgegeben werden, die noch nie
eine Bestellung aufgegeben haben.

--Durch den Natural Left Join werden auch die Benutzer miteingebogen, die noch nie etwas bestellt haben.

SELECT    vorname || ' ' || name || ':' as benutzer, bestellnr
FROM      benutzer NATURAL LEFT JOIN bestellung
ORDER BY  name



--g) Fügen Sie das Geschäft "S.Oliver" hinzu.

INSERT INTO   geschaeft(gid, name)
VALUES        ((SELECT COALESCE(MAX(gid)+1,1)FROM geschaeft),'S.Oliver')



--h) Wieviel muss Uschi Glas für ihre Bestellungen einzeln bezahlen?

--Die Summe einer Bestellung berechnet sich aus der Summe der einzelnen Preise einer Bestellung
SELECT   SUM(preise)as Gesamtsumme_der_Bestellung
FROM     (
         --Der Preis berechnet sich aus der Summe der Anzahl einer Ware "mal" ihrem Preis.
         SELECT   SUM(anzahl)*SUM(preis) as preise,beid
         FROM     bestellung
         JOIN     benutzer USING(bid)
         JOIN     ware_der_bestellung USING (beid)
         JOIN     liefert USING(wid,gid)
         WHERE    name ='Glas' AND vorname='Uschi'
         --Gruppieren nach wid und beid, sodass der Preis einer Bestellung
         --für einen Benutzer berechnet werden kann.
         GROUP BY wid,beid
         )as dummy
--Gruppieren nach Bestellung, damit Bestellungspreis berechnet werden kann.
GROUP BY beid
```

```
--i) H&M möchte den Preis all seiner Artikel um 8% erhöhen. Außerdem soll es keine CENT- sondern nur noch Euro-Beträge
geben. Bringen Sie die Preise auf den neuen Stand.

UPDATE  liefert
--Der Preis der Waren der ausgewählten Geschäfte wird um 8% erhöht und auf Euro-Beträge gerundet.
SET     preis = ROUND(preis + (preis*0.08),0)
WHERE   gid IN (
               --Selektion aller Geschäftsids(gid) mit Namen "H&M", die in der liefer-Tabelle vorkommen
               SELECT   gid
               FROM     liefert JOIN geschaeft USING(gid)
               WHERE    name = 'H&M'
               )



--j) Erstellen Sie eine Übersicht, aus der ersichtlich wird, ob und in welcher Bestellung eine "Jeans" enthalten ist.

SELECT  bestellnr,
--Durch den Case wird geprüft ob die aktuelle beid, bei den selektierten beids enthalten ist.
--Ist das der Fall, wird "JA" ausgegeben, ansonsten "NEIN"
CASE    WHEN  beid = SOME(
                         --Selektion der Bestellungen(beid), die eine Jeans enthalten
                         SELECT    beid
                         FROM      bestellung
                         JOIN      ware_der_bestellung USING(beid)
                         JOIN      ware USING(wid)
                         WHERE     bezeichnung='Jeans') THEN 'ja'
        ELSE 'nein'
END
FROM    bestellung
```

```
--k) Uschi Glas möchte aus all ihren Bestellungen, die noch nicht bearbeitet wurden oder in Bearbeitung sind, alle
Artikel löschen, deren Einzelpreis mehr als 25€ beträgt.

--Entfernt alle Waren(wid) aus der Tabelle ware_der_bestellung, die die genannten Kriterien erfüllen.
--Dabei muss auch darauf geachtet werden, dass NUR die wids gelöscht werden,
--die in den Unteranfragen selektiert wurden
--UND dass noch eine AND-Anweisung angehängt wird, die nochmals sicherstellt,
--dass nur aus Bestellungen von Uschi Glas
--gelöscht wird, deren Status zudem "nichts" oder "in Bearbeitung" ist.

DELETE
FROM    ware_der_bestellung
WHERE   wid IN (
                --Selektion aller Warenids(wid), deren Benutzer Uschi Glas ist,
                --die den Status "nichts" oder "in Bearbeitung" besitzen und deren Preis größer als 25€ ist.
                --Beim Join wird darauf geachtet, dass bei der liefer-Tabelle nach wid UND gid gejoint wird,
                --sodass die richtigen Preise verwendet werden,
                --und nicht zB der Preis der Ware aus einem falschen Geschäft gewählt wird.

                SELECT  wid
                FROM    bestellung
                JOIN    benutzer USING(bid)
                JOIN    status USING(sid)
                JOIN    ware_der_bestellung USING(beid)
                JOIN    liefert USING(wid,gid)
                WHERE   name='Glas' AND vorname='Uschi' AND (status='nichts' OR status='in Bearbeitung')
                AND     preis > '25.00')
AND     beid IN (
                SELECT beid
                FROM   bestellung
                JOIN   benutzer USING(bid)
                JOIN   status USING(sid)
                WHERE  name='Glas' AND vorname='Uschi' AND (status='nichts' OR status='in Bearbeitung')
                )
```

```
--l) Aus welchem Geschäft wird am meisten bestellt?

SELECT  name
FROM    ware_der_bestellung JOIN geschaeft USING(gid)
GROUP   by name
--Es wird der Name des Geschäfts selektiert, dessen Anzahl an Waren mit der höchsten Warensumme aus der
--inneren Answeisung übereinstimmt.
HAVING  (
        SUM(anzahl) =  (
                       --Höchste Warensumme wird selektiert
                       SELECT   MAX(warensumme)
                       FROM     (
                                --Anzahl der Waren, die aus jedem Geschäft verkauft werden, werden gezählt
                                SELECT   SUM(anzahl)as warensumme
                                FROM     ware_der_bestellung JOIN geschaeft USING(gid)
                                --Gruppieren nach Geschäft, um die Waren zu zählen
                                GROUP BY name
                                )as dummy
                        )
         )



--m) Der Shop möchte dem ältesten Benutzer gratulieren. Welcher Benutzer ist der älteste und wie alt ist er?

--Selektion von Name und Alter des Benutzers, der unten ausgewählt wird.
--Aus dem Alter wird nur das Jahr entnommen, da MOnate und Tage eher uninteressant sind.
SELECT  vorname || ' ' || name as benutzer,EXTRACT(YEAR FROM age(geburtsdatum))as alter
FROM    benutzer
WHERE   (
        age(geburtsdatum) =    (
                               --das Alter jedes Benutzers selektiert, und davon aber
                               --die höchste Zahl. Also der älteste Benutzer.
                               SELECT   MAX(age(geburtsdatum))as geburtstag
                               FROM     benutzer
                              )
        )
```

```
--n) Erstellen Sie eine View bei der jede Bestellung, mit Bestellnummer, Benutzer, Gesamtpreis und Status angezeigt
wird.

CREATE VIEW uebersicht (beid,bestellnr,vorname,name,status,gesamtpreis) AS
SELECT  beid,bestellnr,vorname,name,status,gesamtpreis
FROM    bestellung
JOIN    benutzer USING(bid)
JOIN    status USING(sid)
--In dieser Unteranfrage wird der Gesamtpreis einer Bestellung ermittelt.
--Dieser berechnet sich aus der Summe der Einzelpreise aus der Unteranfrage.
JOIN   (SELECT  SUM(preise)as gesamtpreis,beid
        FROM    (
                 --Hier wird der Preis einer Ware in einer Bestellung berechnet, je nach dem wie oft
                 --sie bestellt wurde.
                 SELECT    SUM(anzahl)*SUM(preis) as preise,beid
                 FROM      bestellung
                 JOIN      ware_der_bestellung USING (beid)
                 JOIN      liefert USING(wid,gid)
                 GROUP BY  wid,beid
                 )as dummy
        GROUP BY beid
        )as dummy USING(beid)



--o) Bei welchem Benutzer liegt der Gesamtpreis der Bestellung zwischen 100€ und 300€? Benutzen Sie die oben erstelle
View für Ihre Aufgabe.

SELECT vorname || ' ' || name as name
FROM   uebersicht
WHERE  gesamtpreis BETWEEN 100 AND 300
```

```
--p) Löschen Sie alle Bestellungen, die schon 2 oder mehr Jahre zurückliegen.

--Wo eine Bestellung in den in der Unteranfrage selektierten Bestellungen enthalten ist, wird der Eintrag gelöscht
DELETE
FROM    bestellung
WHERE   beid IN (
                --Es werden die Bestellungen(beid) selektiert, deren Differenz des aktuellen Jahres MINUS
                --des angegebenen Jahres größer gleich 2 ist
                SELECT beid
                FROM   bestellung
                WHERE  EXTRACT('year' FROM current_date) - EXTRACT('year' FROM datum) >= 2
                )



--q) Welche 3 Artikel sind bisher in den meisten Bestellungen vorgekommen und wie oft?

--Die Anzahl der Waren werden gezählt und mit Bezeichnung ausgegeben
SELECT   RANK() OVER (ORDER BY COUNT(wid) DESC) AS platz, COUNT(wid)as anzahl,bezeichnung
FROM     ware_der_bestellung JOIN ware USING(wid)
--Es wird nach Bezeichnung gruppiert um die Anzahl des Auftretens zu zählen
GROUP BY bezeichnung
--Um die höchsten 3 zu selektieren, wird nach Warenid(wid) falsch herum sortiert, damit die
--höchsten Zahlen oben stehen und nur die ersten 3 Einträge entnommen.
ORDER BY COUNT(wid) DESC
LIMIT    3
```

```
--r) An welchem Wochentag wird am meisten bestellt?

SELECT    to_char(datum,'Day')as wochentag
FROM      bestellung
GROUP BY  to_char(datum,'Day')
--Stimmt die Anzahl des Auftretens eines Wochentages mit der höchsten Anzahl aus der Unteranfrage
--überein, so wird dieser Wochentag als am häufigsten vorkommender ausgegeben
HAVING    COUNT(to_char(datum,'Day')) = (
                                        --Die höchste Anzahl wird selektiert
                                        SELECT MAX(anzahl)
                                        FROM   (
                                                --Die Anzahl des Auftretens eines Tagesnamens wird gezählt
                                                SELECT   COUNT(to_char(datum,'Day'))as anzahl
                                                FROM     bestellung
                                                --Es wird nach dem Tagesnamen gruppiert um die Anzahl
                                                --des Auftretens zählen zu können
                                                GROUP BY to_char(datum,'Day')
                                                )as dummy
                                         )



--s) Haben alle Benutzer eine Münchner Telefonvorwahl (089)?
--Wenn ein Benutzer keine Telefonnummer angegeben und alle anderen eine Münchner Vorwahl haben, soll das Ergebniss
NULL lauten.


SELECT  DISTINCT substring(telnummer from 1 for 3) = ALL (SELECT substring(telnummer from 1 for 3)
                                                          FROM   benutzer
                                                          )
FROM    benutzer
WHERE   telnummer LIKE'089%'
```

Studienaufgabe

Spielfilm-Verwaltung

```
--
-- Aufgabe a
--
-- Bei welchen Spielfilmen sind Produzent und Regisseur
-- ein und dieselbe Person?
--
Lösung:

--Suche alle Titel aus der Film-Tabelle in denen String des Produzenten und Regisseurs gleich sind

SELECT  titel
FROM    film
WHERE   produzent = regisseur

--
-- Aufgabe b
--
-- Welche Sender senden "Star Wars - Episode I"?
--
Lösung:

--Deshalb Distinct weil ein Sender öfter auftreten kann.Join zwischen allen 3 Tabellen

SELECT  DISTINCT name
FROM    sendet JOIN film USING(fid) JOIN sender USING (sid)
WHERE   titel = 'Star Wars - Episode I'

--
-- Aufgabe c
--
-- Wie viele Spielfilme sendete die 'ARD' am  25.1.2010?
-- Beachten Sie: Mit DATE_TRUNC('day', mein_timestamp) können
-- Sie das Datum aus einem Timestamp extrahieren.
--
```

Lösung:

```
--Count, da die Anzahl gesucht wird, nur der 25.1.2010 und Sendername Ard interessieren

SELECT  COUNT(fid) as Anzahl_Spielfilme
FROM    sender JOIN sendet USING (sid)
WHERE   name = 'ARD' AND DATE_TRUNC('day', sendezeit) = '25-01-2010'

--
-- Aufgabe d
--
-- Welche Regisseure bevorzugt der Sender "RTL"?
-- Ein Regisseur wird bevorzugt, wenn innerhalb eines Kalendermonats
-- mindestens vier Spielfilme von ihm gezeigt werden.
--
-- Fleißaufgabe: Wenn Sie eine etwas herausfordernde Aufgabe
--               bearbeiten wollen, können Sie "bevorzugt"
--               auch als "innerhalb von vier aufeinanderfolgenden
--               Wochen mindestens vier Spielfilme" definieren.  :-)
--               Deutlich einfacher wird es, wenn Sie "bevorzugt"
--               als "mindestens vier Filme innerhalb der nächsten
--               vier Wochen" definieren.
--
Lösung:


--Distinct, da ein Regisseur auch öfter auftritt
--Zunächst Heraussuchen aller Sender die RTL heißen
--Diese Tabelle wird nach Monat und Regisseur gruppiert
--es wird nur der Regisseur selektiert, der mindestens 4mal in dieser Tabelle auftritt


SELECT DISTINCT regisseur
FROM            sendet JOIN film USING (fid) JOIN sender USING(sid)
WHERE           name='RTL'
GROUP BY        DATE_TRUNC('month', sendezeit), regisseur
HAVING          COUNT(regisseur) >= 4
```

```
--
-- Aufgabe e
--
-- Wurden alle Star-Wars-Filme (die in der Datenbank enthalten sind!)
-- vom selben Produzenten produziert? Das Ergebnis Ihrer Anfrage soll
-- true oder false lauten!
-- (Gehen Sie davon aus, dass alle Star-Wars-Titel in der Form 'Star Wars - ...'
-- in der Datenbank gespeichert sind.)
--
Lösung:


--Es wir eine Unteranfrage erstellt, in der alle Produzenten aus der Tabelle film selektiert werden,
--bei denen der Titel "Star Wars" enthält und deren Produzent bekannt ist.
--Danach wird jeder Produzent aus der Filmtabelle, bei dem der Filmtitel "Star Wars" enthält und einen
--Produzenten besitzt mit jedem Produzenten aus der Unteranfrage verglichen.
--Stimmen alle Produzenten überein, sind alle Star-Wars-Filme vom gleichen Produzenten.
--DISTINCT deshalb, weil nur der Name interessant ist und nicht die Anzahl.


SELECT  DISTINCT produzent = ALL(SELECT produzent
                                 FROM film
                                 WHERE titel LIKE 'Star Wars -%' AND produzent is not NULL)
                                 as Sind_alle_StarWarsFilme_vom_selben_Produzenten
FROM    film
WHERE   titel LIKE 'Star Wars -%'AND produzent is not NULL


--
-- Aufgabe f
--
-- Welche Ausgabe produziert Ihre Anfrage, wenn in der Datenbank für einen
-- Star-Wars-Film kein Produzent (Wert NULL) und für alle anderen Star-Wars-Filme
-- ein- und derselbe Produzent eingetragen wurde? Begründen Sie Ihre Antwort
-- kurz.
```

```
Antwort:
Das Ergebnis lautet Wahr/True, da nichtangegebene Produzenten nicht in den Vergleich miteinbezogen werden.

-- Verbessern Sie Ihren Code aus Aufgabe e, falls Ihre Anfrage im
-- zuvor geschilderten Fall 'true' ausgibt.
--
Lösung:


--Die erste Anweisung "AND produzent is NOT NULL" wird weggelassen, so werden auch NULL-werte miteinbezogen.
--Besitzt ein Produzent den Wert NULL, liefert die Anfrage den Wert "NULL" also "UNKNOWN"


SELECT  DISTINCT produzent = ALL(SELECT produzent
                                 FROM film
                                 WHERE titel LIKE 'Star Wars -%')
                                 as Sind_alle_StarWarsFilme_vom_selben_Produzenten
FROM    film
WHERE   titel LIKE 'Star Wars -%' AND produzent is not NULL
```

```
--
-- Aufgabe g
--
-- Welche Spielfilme werden an zwei Tagen hintereinander gesendet?
--
Lösung:


--Hauptanfrage, die die JOIN-Tabelle aus sendet und film t1 nennt.
--Unteranfrage, die die gleiche JOIN-Tabelle t2 nennt. Aus dieser werden alle Titel selektiert,
--wenn die Sendezeit aus t2 gleich der Sendezeit aus t1 + 1Tag, und somit der Folgetag ist.
--In der Hauptanfrage werden die Titel mit DISTINCT aus der JOIN-Tabelle selektiert.


SELECT  DISTINCT titel
FROM    (sendet JOIN film USING (fid)) as t1
WHERE   EXISTS  (
                SELECT  titel
                FROM    (sendet JOIN film USING(fid)) as t2
                WHERE   t2.sendezeit = (t1.sendezeit + INTERVAL '1 day')
                AND     t2.titel = t1.titel
                )


--
-- Aufgabe h
--
-- Welcher Sender sendet alle (in der Datenbank vorhandenen)
-- Filme des Regisseurs Lucas?
--
Lösung:

--Erklärung von innen nach außen:
--Zunächst werden in Zeile 14-16 alle sids selektiert, die überhaupt in Frage kommen,
--also folglich einen Film mit dem Regisseur "Lucas" senden.
--Die Select-Anweisung der Zeile 11 simuliert den Fall, dass jeder Sender bei jedem Film den Regisseur "Lucas"
besitzt.
```

```
--Durch die EXCEPT-Anweisung in Zeile 18 werden die Sender selektiert, die NICHT bei jedem Film den Regisseur "Lucas"
besitzen.
--Die Select-Anweisung(ab Zeile 9) selektiert aus den Unteranfragen alle sids,
--die NICHT bei jedem Film den Regisseur "Lucas" besitzen.
--Diese Select-Anweisung wird mit EXCEPT an die Select-Anweisung der dritten Zeile angehängt.
--Die Select-Anweisung der dritten Zeile selektiert wieder alle Sender, die irgendeinen Film mit dem Regisseur "Lucas"
senden.
--Somit bekommt man am Ende alle Sender, die alle Filme des Regisseurs "Lucas" senden.
--Am Ende wird nur noch der Name des Senders selektiert.

SELECT name
FROM   (
       SELECT   sendet.sid
       FROM     sendet JOIN film USING(fid) JOIN sender USING(sid)
       WHERE    regisseur='Lucas'

       EXCEPT

       SELECT   dummy.sid
       FROM     (
                SELECT  sid,fid
                FROM    (
                         SELECT  DISTINCT sendet.sid
                         FROM    sendet JOIN film USING(fid)
                         WHERE   film.regisseur ='Lucas'
                         )as dummy,film
       WHERE   regisseur='Lucas'

       EXCEPT

       SELECT  sendet.sid,sendet.fid
       FROM    sendet,film
       WHERE   regisseur='Lucas'
                )as dummy,sender
       )as dummy JOIN sender USING(sid)
```

```
--
-- Aufgabe i
--
-- Erstellen Sie eine Übersicht über alle Spielfilme, die am
-- 30. Januar 2010 gesendet werden. Die Übersicht soll enthalten:
-- Sendername, Film-Titel, Uhrzeit (TO_CHAR(mein_timstamp, 'HH24:MI')), Fassung
--
-- Sortieren Sie die Liste  SQL:2003-konform nach
-- Sendername und Sendebeginn (Uhrzeit).
--
-- Beachten Sie, dass es Sender geben kann für die aus irgendwelchen
-- Gründen die Sende-Daten noch nicht vorliegen. Listen Sie auch diese
-- Sender in der Übersicht auf.
--
Lösung:


--Die Tabelle wird mit einem Left-Join erstellt, da auch die Sender aufgeführt werden sollen,
--zu denen noch keine Sendedaten existieren.
--Wichtig hierbei ist, dass sowohl alle Daten des 30.1.2010, als auch alle Daten der Sendezeit
--bedacht werden, deren Wert gleich NULL ist


SELECT          name AS sendername, titel AS Filmtitel, TO_CHAR(sendezeit, 'HH24:MI') AS Uhrzeit, fassung
FROM            sender LEFT JOIN sendet USING(sid) LEFT JOIN film USING(fid)
WHERE           DATE_TRUNC('day', sendezeit) = '30-01-2010' OR DATE_TRUNC('day', sendezeit) IS NULL
ORDER BY        sendername,uhrzeit
```

Multimedia
Datenbanksysteme

Erstellungserklärung

für das Fach Datenmanagement II (Multimedia-Datenbanksysteme) bei Prof. Wolfgang Kowarschick

Titel der Studienarbeit:	Onlineshop-Datenbank
Datum der 1. Zwischenabgabe:	20.04.2012
Datum der 2. Zwischenabgabe:	25.05.2012
Datum der Abgabe:	16.07.2012
DBMS:	PostgreSQL
Autoren (Vor- und Zuname):	Daniela Huber
	Matr.Nr. 924323

Hiermit erkläre ich/erklären wir, dass ich/wir die vorgelegte Arbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel verwendet sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe/haben.

Unterschrift aller Autoren

DHuber

Abgabeliste

Dokumentation (ausgedruckt)		ja	nein
1.	gebunden (Schnellhefter ohne Lochung; keine Klebe-/Spiralbindung)	☒	☐
2.	Titelblatt (mit Titel + Autoren)	☒	☐
3.	Kurzbeschreibung	☒	☐
4.	Use-Cases-Diagramm + textuelle Beschreibung	☒	☐
5.	ER-Diagramm (UML-Notation) + textuelle Beschreibung	☒	☐
6.	Relationales Schema	☒	☐
7.	SQL-Befehle: CREATE TABLE (DDL)	☒	☐
8.	SQL-Befehle: INSERT von Daten einer Beispiels-Datenbank	☒	☐
9.	SQL-Befehle: SELECT, UPDATE, VIEW, TIGGER, ...	☒	☐
10.	Lösung einer Studienaufgabe (pro Autor eine andere Aufgabe)	☒	☐
11.	*Nur bei mehreren Autoren*: Übersicht, welcher Autor welche Teile der Studienarbeit erstellt hat	☐	☒

CD/DVD/USB-Stick		ja	nein
12.	Das Root-Verzeichnis enthält eine Text-Datei oder PDF-Datei deren Name die Namen aller Autoren enthält. (`Name1_Vorname1__Name2_Vorname2-.txt`)	☒	☐
13.	*Nur bei mehreren Autoren:* Diese Datei enthält eine Übersicht, welcher Autor welchen Teil der Studienarbeit erstellt hat.	☐	☒
14.	Das Root-Verzeichnis enthält einen Ordner „`dokumentation`“.	☒	☐
15.	In diesem Ordner ist die vollständige Dokumentation der Studienarbeit enthalten (nur PDF, TXT, PNG, JPEG).	☒	☐
16.	Alle Dateien haben aussagekräftige Namen.	☒	☐
17.	Das Root-Verzeichnis enthält einen Ordner „`studienarbeit`“.	☒	☐
18.	In diesem Ordner ist eine vollständige und nicht gepackte (d.h. kein ZIP-, RAR-Archiv) SQL-Implementierung in UFT8-Text-Dateien enthalten.	☒	☐
19.	Der SQL-Code enthält aussagekräftige Inline-Kommentare.	☒	☐
20.	Der SQL-Code behandelt mindestens fünf komplexe Problemgebiete:	☒	☐

Gruppierung: ☒ Subqueries: ☒ Windowing: ☒ Views: ☒ Rekursion: ☐
Für-alle-Query: ☒ Volltextsuche: ☐ Null-Values: ☒ Outer-Join: ☒ Trigger: ☐
Sonstiges: ☒ Was genau?: Aggregation, Datum

Multimedia
Datenbanksysteme

		ja	nein
21.	Das Root-Verzeichnis enthält einen Ordner „`studienaufgabe`“.	☒	☐
22.	*Nur bei mehreren Autoren:* Dieser Ordner enthält für jeden Autor einen eigenen Unterordner mit dem Namen des jeweiligen Autors.	☐	☒
23.	Für jeden Autor wurde eine nur von ihm (d.h. nicht in Teamarbeit) erstellte Lösung einer Studienaufgabe in den passenden Ordner eingefügt.	☒	☐
24.	Auf der CD/DVD sind keine weiteren Dateien enthalten.	☒	☐

25. Falls „nein“ (d.h., falls weitere Dateien enthalten sind): Welche und wozu?

Dateiname	Begründung

26.	Alle Dateien, die sich auf der CD/DVD befinden, sind unter denselben Namen im Repository daniah (RZ-Name) enthalten.	☒	☐
27.	Das Repository wurde nach der zweiten Zwischenabgabe regelmäßig (mindestens zehn Mal) aktualisiert.	☒	☐

28. Für jedes „nein“ (mit Ausnahme von Frage 24 und – bei einem einzelnen Autor – mit Ausnahme der Fragen 11, 13, und 22) folgt eine Begründung (evtl. Rückseite benutzen). *Sollte für irgendeinen der wedentlichen Punkte keine triftige Begründung angegeben werden, so gilt die Arbeit als nicht abgegeben.*

Nr.	Begründung

Unterschrift aller Autoren

DHber

3 / 3