Klausur Datenbanken
Wintersemester 2021/2022
Prof. Dr. Wolfgang May
2. März 2022, 11:15-12:45 Uhr
Bearbeitungszeit (Papier): 85 Minuten (Ilias: 90 Minuten)
(Ilias-basierte Klausur)

Vorname:

Nachname:

Matrikelnummer:

Zwecks besserer Lesbarkeit (insbesondere auch für Nicht-{Mut/d/Va}tersprachler*innen) wird in der Aufgabenstellung auf gegenderte Sprache verzichtet.

- Im Folgenden wird die Aufgabenstellung beschrieben. Für das ER-Diagramm, und auch für das relationale Schema ist ein Teil bereits vorgegeben. In den Aufgaben 1 und 2 (wahlweise, ER-Diagramm) und 3 und 4 (wahlweise, Umsetzung in das relationale Modell) müssen nur noch die fehlenden Teile ergänzt werden.
- Bearbeiten Sie zuerst *entweder* Aufgabe 1 *oder* 2 (ER-Diagramm), dann *entweder* Aufgabe 3 *oder* 4 (Umsetzung in das Relationale Modell), und dann die weiteren Aufgaben, die darauf aufbauen.

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell (Foto oder PDF-Upload))	14	12
Aufgabe 2 (ER-Modell (ASCII-Art-Modus))	0	0
Aufgabe 3 (Transformation in das Rel. Modell (Upload))	9	7
Aufgabe 4 (Transformation in das Rel. Modell (ASCII))	0	0
Aufgabe 5 (Relationales Modell: CREATE TABLE)	5	3
Aufgabe 6 (Anfragen (1))	6	5
Aufgabe 7 (Anfragen (2))	4	3
Aufgabe 8 (Anfragen (3))	4	2
Aufgabe 9 (Anfragen (4))	5	2
Aufgabe 10 (Anfragen (5))	6	1
Aufgabe 11 (Anfragen (6))	5	1
Aufgabe 12 (Anfragen (wahlweise: Bäume als Upload))	0	0
Aufgabe 13 (Anfragen (7))	6	0
Aufgabe 14 (Update an der Datenbank)	4	3
Aufgabe 15 (Interne Auswertungsstrategie)	6	0
Aufgabe 16 (Eine etwas kompliziertere SQL-Anfrage)	6	0
Summe	80	39

Note:

Themenstellung: Corona-Fallerfassung

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll eine Datenbank für die Verwaltung der Covid-19-Fälle entwickelt werden.

> *Die Aufgabenstellung setzt die Thematik der Klausur vom WS 2020/21 fort (wobei man diese garnicht kennen muss – hier wird alles wichtige nochmal beschrieben), in der eine Datenbank zur Erfassung der Impfungen erstellt wurde. Dieser Teil ist jetzt etwas reduziert vorgegeben. In dieser Klausur wird die Speicherung der Covid-19-Fälle hinzugefügt.*

Vorgegebener Teil (ähnlich, aber etwas reduziert gegenüber WS 2020/21):

1. In der Datenbank sollen alle in Deutschland lebenden Personen enthalten sein. Jede Person wird nur durch eine ID (In dieser Klausur besteht die ID zur Vereinfachung immer nur aus den Initialen der Personen) repräsentiert.
   In der Datenbank wird für jede Person außer der ID das Geburtsdatum und ggf. das Sterbedatum gespeichert; außerdem, in welchem Kreis jemand zur Zeit lebt bzw. zuletzt gelebt hat. Daten über verstorbene Personen bleiben gespeichert.

   Elsa Müller (ID: EM) wurde am 4.3.1943 geboren, *Fritz Müller* (ID: FM) wurde am 3.4.1934 geboren. FM ist am 20.7.2021 gestorben. Beide leb(t)en im Kreis Göttingen.

   Markus Schmidt (ID: MS) wurde am 7.7.1977 geboren, er lebt im Kreis Göttingen.

   Andrea Meier (ID: AM) wurde am 9.9.1979 geboren, und lebt im Kreis Northeim.

   Karl Napf (ID: KN) wurde am 4.4.1944 geboren, ist am 30.1.2022 gestorben, und lebte zuletzt im Kreis Northeim.

2. Für jeden Kreis (= deutsche Verwaltungseinheit, entspricht district, county) ist der Name, die Bevölkerungszahl, und zu welchem Bundesland er gehört, gespeichert. Der Kreis *Göttingen* hat 330000 Einwohner, der Kreis *Northeim* hat 130000 Einwohner, beide gehören zum Bundesland *Niedersachsen*.

3. Es gibt verschiedene Impfstoffe. Jeder Impfstoff hat einen eindeutigen Namen. Außerdem ist der Name des Herstellers sowie das Datum der Zulassung in Deutschland gespeichert. Es kann mehrere/neue Impfstoffe desselben Herstellers geben (z.B. einen gegen die Omikron-Variante):

   Der Impfstoff *Comirnaty* des Herstellers *Biontech* wurde am 21.12.2020 zugelassen.

   Der Impfstoff *ComirOmi* (gegen die Omikron-Variante) des Herstellers *Biontech* befindet sich noch in der Studienphase, ist also noch nicht zugelassen.

   Der Impfstoff *Spikevax* des Herstellers *Moderna* wurde am 6.1.2021 zugelassen.

   Der Impfstoff *Vaxzevria* des Herstellers *AstraZeneca* wurde am 29.01.2021 zugelassen.

   Der Impfstoff *Nuvaxovid* des Herstellers *Novavax* wurde am 20.12.2021 zugelassen.

4. Die einzelnen Impfungen sind gespeichert:

   - Jede Person wird üblicherweise mehrmals, mit demselben oder unterschiedlichen Impfstoffen, geimpft. Die meisten Personen haben bisher zwei oder drei Impfungen erhalten, manche Senioren und immungeschwächte Personen können

jedoch bereits eine vierte Impfung bekommen haben.
In der Datenbank wird gespeichert, wer wann mit welchem Impfstoff geimpft wurde:

- *Elsa Müller* (ID: EM) und *Fritz Müller* (ID: FM) wurden beide am 28.12.2020 und am 8.2.2021 mit *Comirnaty* geimpft. EM wurde am 15.9.2021 ein weiteres Mal mit *Comirnaty* geimpft, und hat am 22.2.2022 bereits eine vierte Impfung im Rahmen einer Studie mit dem neuen noch nicht zugelassenen Impfstoff *ComirOmi* erhalten.
- *Markus Schmidt* (ID: MS) (Klausur WS2021: er ist Altenpfleger) wurde am 15.2.2021 sowie am 10.3.2021 mit *Vaxzevria* geimpft. Seine dritte Impfung erfolgte am 11.11.2021 mit *Spikevax*.
- *Andrea Meier* (ID: AM) wurde am 20.8.2021 sowie am 10.10.2021 mit *Spikevax* geimpft.

In dieser Klausur sollen die folgenden Daten hinzugenommen werden:

5. Personen werden auf Corona-Infektionen getestet. Hierbei wird das Datum und das Ergebnis, "positiv", wenn eine Infektion vorliegt, "negativ", wenn *keine* Infektion vorliegt, gespeichert. Personen können mehrmals, auch mehrmals positiv, getestet werden. Zu jeder Person wird maximal ein Text pro Tag gespeichert.

   - *Fritz Müller* (ID: FM) wurde am 10.7.2021 corona-positiv getestet, und ist –ohne Krankenhausaufenthalt– am 20.7.2021 im Pflegeheim verstorben.
   - *Andrea Meier* (ID: AM) wurde am 3.1.2022 corona-positiv getestet, und am 10.1.2022 corona-negativ.
   - *Karl Napf* (ID: KN) wurde am 10.10.2020 corona-positiv, am 22.10.2020 corona-negativ, sowie am 7.1.2022 ein weiteres Mal corona-positiv getestet.

6. Für jedes Krankenhaus ist der Name und in welchem Kreis es liegt, abgespeichert.

   - Das *Uni-Klinikum Göttingen* und das *Weender Krankenhaus* liegen im Kreis Göttingen, das *Kreiskrankenhaus Northeim* liegt im Kreis Northeim.

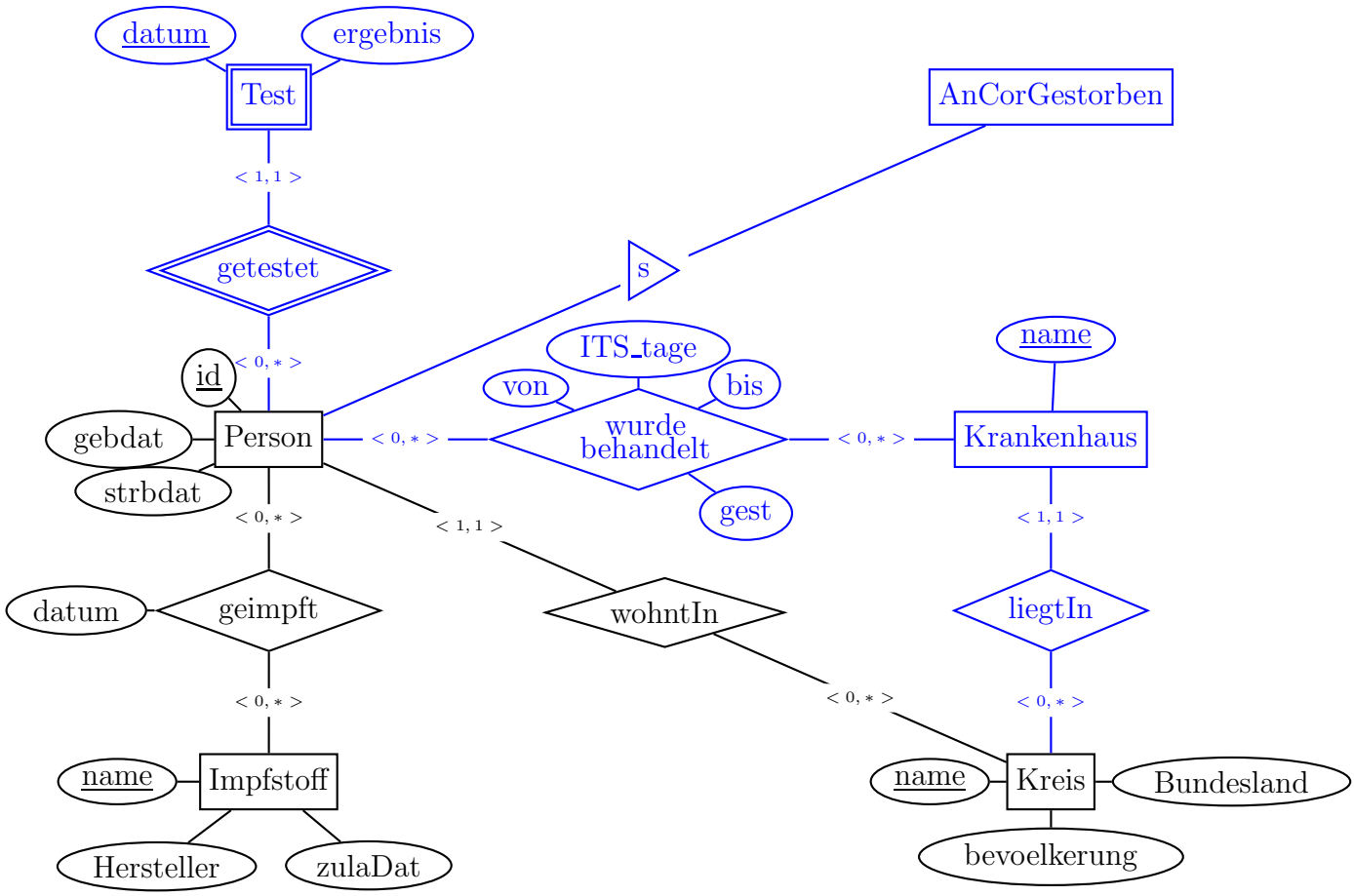
7. Es wird abgespeichert, wenn Personen mit Corona-Diagnose in ein Krankenhaus eingeliefert werden: Dort wird der Tag der Aufnahme, und der Tag der Entlassung oder der Sterbetag gespeichert, und ggf. wieviele Tage jemand auf der Intensivstation verbracht hat. Personen können auch mehrmals mit Corona in ein Krankenhaus eingeliefert werden.

   - Markus Schmidt (MS) wurde am 27.2.2022 Corona-positiv getestet, und in das *Weender Krankenhaus* eingeliefert. Dort befindet er sich momentan.
   - Karl Napf (KN) wurde am 11.10.2020 mit Corona-Diagnose in das *Kreiskrankenhaus Northeim* eingeliefert, und ist am 24.10.2020 gesund entlassen worden; dabei hatte er 3 Tage auf der Intensivstation gelegen.
   - Karl Napf (KN) wurde am 9.1.2022 ein weiteres Mal mit Corona-Diagnose in das *Kreiskrankenhaus Northeim* eingeliefert, und ist dort an Corona am 30.1.2022 gestorben, nachdem er 20 Tage auf der Intensivstation behandelt wurde.

8. Es wird gespeichert, welche Personen wann an Corona gestorben sind. Dabei ist ggf. zu berücksichtigen, dass nicht alle vorher in einem Krankenhaus waren (siehe Fritz Müller (FM)).

Aufgabe 1 (ER-Modell (Foto oder PDF-Upload) [14 Punkte])
Vervollständigen Sie das angegebene ER-Modell (einschl. Kardinalitäten).
(d.h. machen Sie eine eigene Grafik (z.B. mit draw.io) in der Sie von dem vorhandenen *nur die benötigten Entitätstypen (ohne deren Attribute)* "neu" zeichnen und das Gesuchte dazwischenfügen, oder kopieren Sie das untenstehende ER-Diagramm in ein Zeichenprogramm und malen rein ... oder verwenden ein Blatt Papier+Smartphone.)

- **Wenn Sie die ER-Modell-Aufgabe als pdf/png/jpg- oder Foto-Upload bearbeiten wollen, machen Sie dies HIER,**
- **wenn Sie die Aufgabe stattdessen als ASCII-Art im Editor bearbeiten wollen, machen Sie dies in Aufgabe 2.**



Lösung
Eine naheliegende Lösung ist obenstehend eingezeichnet.

Alternativen und Hinweise:

- CoronaTodesfall kann auf verschiedene Arten modelliert werden.: Für Corona-Todesfälle, die nicht im Krankenhaus aufgetreten sind, muss man die Modelleierung geeignet wählen:

  - die obige Lösung modelliert "AnCorGestorben" als Subklasse von Person.
    Damit resultiert die Umsetzung in das relationale Modell in einer einspaltigen Tabelle (=Menge), die nur die ID der Personen enthält.
    Das Sterbedatum ist ja sowieso bei Person dabei, damit hat man keine Redundanz, muss aber ggf. einmal mehr joinen.
  - "gestorbenAnCorona" als Attribut zu Person (boolean oder Datum).
    Nachteil: ist aber dann nur sehr selten non-null; besser separat ablegen (das ist aber schon eher die nächste Aufgabe).
  - "Todesursache" als Attribut zu Person.

  - (Schwacher) Entitätstyp CoronaTodesfall mit und identifizierender Beziehung "verstorbenAn" zu Person. Man kann ein Attribut "datum" dazumodellieren, was aber etwas redundant (zu Person.Sterbedatum) ist.
    In dem Fall resultiert die Umsetzung in das relationale Modell in einer ein- oder zweispaltigen Tabelle (id, datum).
  - eine Beziehung "verstorben" zwischen Person und Test (mit oder ohne Datum-Attribut).

- wurdeBehandelt mit Attributen entlassen und gestorben (als exclusive-or gedacht) anstelle (bis, gestorben).
  (wenn "gestorben" bei "wurdeBehandelt" nicht dabei ist, ist nicht sicher, ob die Person im Krankenhaus oder ggf nach Entlassung gestorben ist). -1/2P

- Einen (reifizierten, schwachen) Entitätstyp "Krankenhausaufenthalt" anstelle der Beziehung "wurdeBehandelt". Er steht mit <1,1>-Kardinalität in zwei (identifizierenden) Beziehungen zu Person bzw. Krankenhaus. Das relationale Modell ist –bis auf den Tabellennamen– dann das gleiche.

Aufgabe 2 (ER-Modell (ASCII-Art-Modus) [0 Punkte])
Alternativ zu Aufgabe 1 können Sie dasselbe hier als ASCII-Art im Editor bearbeiten. So etwa so:

```
[Entity]---<0,1>---<<ident-beziehung>>---<1,1>---[[weakEnt]]-(attr)
   \                                                    |
 (_keyattr_)                                       (_keyattr_)
```

Aufgabe 3 (Transformation in das Rel. Modell (Upload) [9 Punkte])
Vervollständigen Sie in dieser Aufgabe das relationale Modell. Gegeben sind die folgenden Relationen:

Person

id	GebDatum	SterbeDatum	Kreis
EM	4.3.1943	NULL	Göttingen
FM	3.4.1934	20.7.2021	Göttingen
MS	7.7.1977	NULL	Göttingen
AM	9.9.1979	NULL	Northeim
KN	4.4.1944	30.1.2022	Northeim
:	:	:	:

Kreis

Name	Bevölkerung	Bundesland
Göttingen	330000	Niedersachsen
Northeim	130000	Niedersachsen
:	:	:

Impfstoff

Name	Hersteller	Zulassungsdatum
Comirnaty	BionTech	21.12.2020
ComirOmi	BionTech	NULL
Spikevax	Moderna	06.01.2021
Vaxzevria	AstraZeneca	29.01.2021
Nuvaxovid	Novavax	20.12.2021

geimpft

Person	Datum	Impfstoff
EM	28.12.2020	Comirnaty
EM	8.2.2021	Comirnaty
EM	15.9.2021	Comirnaty
EM	22.2.2022	ComirOmi
FM	28.12.2020	Comirnaty
FM	8.2.2021	Comirnaty
MS	15.2.2021	Vaxzevria
MS	10.3.2021	Vaxzevria
MS	11.11.2021	Spikevax
AM	20.8.2021	Spikevax
AM	10.10.2021	Spikevax
:	:	:

Geben Sie die noch fehlenden Tabellen (mit Attributen, Schlüsseln, Fremdschlüsseln etc.) für die Tests, Krankenhäuser und -behandlungen mit jeweils mindestens zwei Beispieltupeln (z.B. welche, die sich aus dem Aufgabentext ergeben) an (kein SQL `CREATE TABLE`-Statement, sondern grafisch als Tabellen).
Geben Sie die Fremdschlüsselreferenzen in der Form

```
rel1(A,B) -> rel2(X,Y)
```

an.

- **Wenn Sie die Relationales-Modell-Aufgabe als pdf/png/jpg- oder Foto-Upload bearbeiten wollen, machen Sie dies HIER,**
- **wenn Sie die Aufgabe stattdessen als ASCII-Text im Editor bearbeiten wollen, machen Sie dies in Aufgabe 4.**

Lösung

AnCorGest

Person
KN
:

Krankenhaus

Name	Kreis
Uni-Kl. Göttingen	Göttingen
Weender Krhs.	Göttingen
Kreiskrhs. Northeim	Northeim
:	:

Test

Person	Datum	Ergebnis
FM	10.7.2021	pos
AM	3.1.2022	pos
AM	10.1.2022	neg
KN	10.10.2020	pos
KN	22.10.2020	neg
KN	7.1.2022	pos
MS	27.2.2022	pos

behandeltIn					
Person	Krankenhs	von	bis	its_tage	gestorben
KN	Kreiskrhs. Northeim	10.10.2020	24.1.2022	3	NULL
KN	Kreiskrhs. Northeim	9.1.2022	30.1.2022	20	30.1.2022
MS	Weender Krankenhaus	27.2.2022	NULL	NULL	NULL
:	:	:	:	:	:

FKs:
AnCorGest(Person)→Person(id)
Test(Person)→Person(id),
Krankenhaus(Kreis)→Kreis(name),
behandeltIn(Person)→Person(id), behandeltIn(Krankenhs)→Krankenhaus(Name)

Aufgabe 4 (Transformation in das Rel. Modell (ASCII) [0 Punkte])
Alternativ zu Aufgabe 3 können Sie dasselbe (Tabellen und Fremdschlüsselreferenzen) hier als Textfile eingeben
(Empfehlung: editieren Sie es in einer lokalen Datei und kopieren es dann ins Ilias, dann können Sie Ihre lokale Datei zur Bearbeitung der SQL-Aufgaben auch sehen).

- Die Tabellenskizze kann z.B. so aussehen:

```
tabname(_attr1_,_attr2_,attr3, attr4)
-------------------------------------
         bsp11   bsp12  bsp13  bsp14
         bsp21   bsp22  bsp23  bsp24

oder analog mit
tabname(attr1,attr2,attr3,attr4)  Primary Key: (attr1,attr2)
```

- Geben Sie die Fremdschlüsselreferenzen in der Form

```
rel1(A,B) -> rel2(X,Y)
```

an.

Aufgabe 5 (Relationales Modell: CREATE TABLE [5 Punkte])
Geben Sie das `CREATE TABLE`-Statement für Ihre Tabelle für die Krankenhausbehandlungen so vollständig wie möglich an.

Lösung

```
CREATE TABLE behandeltIn                          1P Basis
(Person VARCHAR2(20) REFERENCES Person(id),   1/2 P
 Krankenh VARCHAR2(20) REFERENCES Krankenhaus(name),  1/2 P
 von DATE CHECK (von BETWEEN 01.02.2020 AND SYSDATE),
 bis DATE CHECK (bis BETWEEN 01.02.2020 AND SYSDATE),  1P für die vier Zeilen
 its_tage NUMBER CHECK (its_tage >= 0),
 gestorben DATE CHECK (bis BETWEEN 01.02.2020 AND SYSDATE),
   -- diese als Tabellenconstraintsm, da sie mehrere Attribute betreffen:
 CHECK (von <= bis),          1/2
```

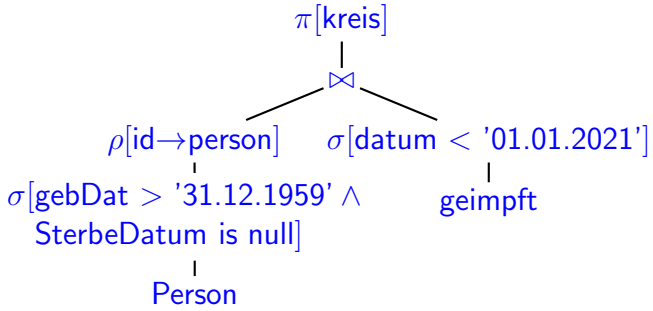
```
 CHECK (von <= gestorben),   1/2
 CHECK (intensiv <= (bis - von)),   +1/2 wer dran denkt
 PRIMARY KEY (Person, von)   1 P
)
Hinweis: intensiv kann damit NULL oder 0 sein, wenn jemand dort nicht war;
 man könnte NULL aber auch verbieten
```

Aufgabe 6 (Anfragen (1) [6 Punkte])
Verwenden Sie für diese und die folgenden Aufgaben die von Ihnen vervollständigte relationale Datenbasis. Keine der Antworten soll Duplikate enthalten.

Geben Sie eine SQL-Anfrage *und* einen Ausdruck oder Baum der relationalen Algebra an, die die Namen aller Kreise zurückgeben, in denen mindestens eine Person lebt, die nach dem 31.12.1959 geboren wurde, und vor dem 1.1.2021 geimpft wurde.
Sie können wahlweise beides zusammen im untenstehenden Texteingabefeld eingeben, oder den Algebra-Baum in Aufgabe 12 als Grafik oder Foto hochladen.
Falls Sie den Algebra-Ausdruck/Baum hier eingeben wollen, schreiben Sie Pseudocode mit pi[...](...), sigma[...](...), join(...,...), etc.

Lösung

```
SELECT DISTINCT kreis
FROM Person p, geimpft i
WHERE p.id = i.person
  AND p.SterbeDatum IS NULL
  AND i.Datum < '01.01.2021'
  AND p.gebDatum > '31.12.1959'
```



Aufgabe 7 (Anfragen (2) [4 Punkte])
Geben Sie eine SQL-Anfrage an, die für jedes Krankenhaus ausgibt, wieviele Corona-Fälle im Dezember 2021 eingeliefert wurden, die zuvor mindestens eine Impfung erhalten hatten.

Lösung

```
SELECT krankenhs, COUNT(distinct Person) -- falls jemand rein-raus-wieder rein +1,
FROM behandeltIn b, geimpft i
WHERE b.Person = i.Person
  AND von BETWEEN '01.12.2021' AND '01.01.2022'
```

```
  AND i.datum < von
GROUP BY krankenhs

SELECT krankenhs, COUNT(distinct Person)
FROM behandeltIn
WHERE von BETWEEN '01.12.2021' AND '01.01.2022'
  AND person in (SELECT person
                 FROM geimpft
                 WHERE Datum < von)
GROUP BY krankenhs
```

Hinweis: BETWEEN komplettiert Datumswerte mit 00:00:00h;
30.11.2021 bzw 31.12.2021 wurden auch als korrekt gewertet.
Es ist nicht notwendig, explizit noch die Tabelle "Test" anzufragen, da davon ausgegangen wird, dass in der DB nur Corona-Fälle gespeichert werden.

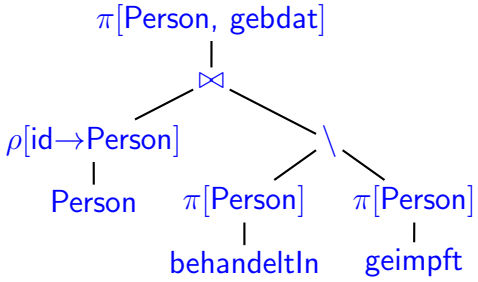
Aufgabe 8 (Anfragen (3) [4 Punkte])
Geben Sie eine SQL-Anfrage (und in der *nächsten Aufgabe* einen Algebra-Ausdruck oder Baum) an, die die ID und das Geburtsdatum aller Personen ausgibt, die nie geimpft wurden, und mindestens einmal mit Corona in einem Krankenhaus behandelt wurden.

Lösung

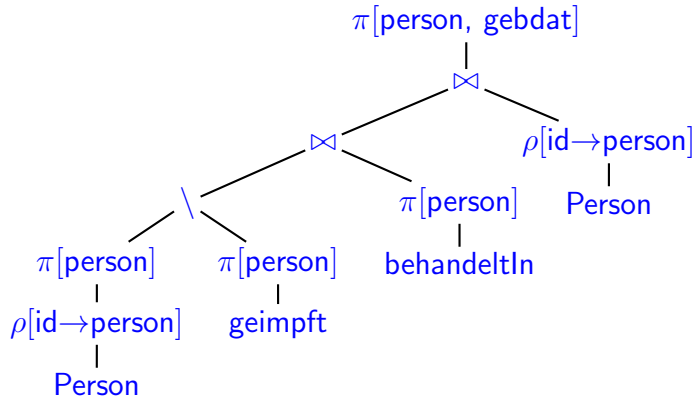
```
SELECT DISTINCT id, gebDat
FROM Person, behandeltIn
WHERE id = Person
AND id NOT IN  (SELECT person FROM geimpft)

SELECT id, gebDat
FROM Person
WHERE id IN (SELECT person FROM behandeltIn)
AND id NOT IN  (SELECT person FROM geimpft)
```

Aufgabe 9 (Anfragen (4) [5 Punkte])
Geben Sie einen Algebra-Ausdruck oder -Baum an, der die Anfrage aus der vorhergehenden Aufgabe beantwortet.
Sie können den Audruck oder Baum wahlweise hier (als ASCII-Art im Editor) abgeben, oder in Aufgabe 12 als Grafik oder Foto hochladen.

Lösung



Wenn man mit (person \ geimpft) ⋈ behandeltIn anfängt, muss man die Person am Ende nochmal dranjoinen, um auch das Geburtsdatum zu bekommen:



Aufgabe 10 (Anfragen (5) [6 Punkte])
Geben Sie eine SQL-Anfrage (und in der *nächsten Aufgabe* einen Algebra-Ausdruck oder Baum) an, der die Namen derjenigen Impfstoffe i ausgibt, für die gilt: in jedem Kreis lebt mindestens eine Person, die mindestens einmal mit dem Impfstoff i geimpft worden ist.

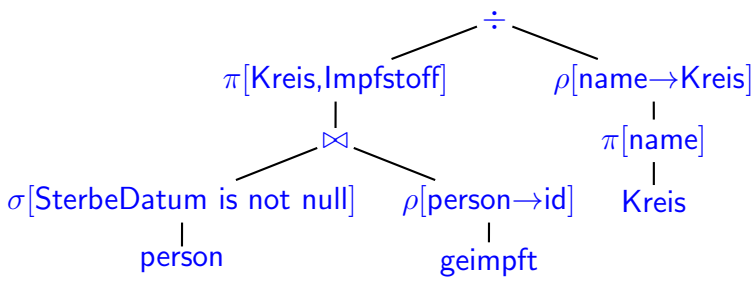
Lösung

```
SELECT name
FROM Impfstoff i
WHERE NOT EXISTS
  (SELECT *
   FROM Kreis k
   WHERE NOT EXISTS
     (SELECT *
      FROM Person p, geimpft g
      WHERE p.id=g.person
        AND p.SterbeDatum IS NULL
        AND p.kreis = k.name
        AND g.Impfstoff = i.name))
```

```
SELECT name
FROM Impfstoff i
WHERE NOT EXISTS
  (SELECT *
   FROM Kreis k
   WHERE (i.name, k.name) NOT IN
     (SELECT g.Impfstoff, p.Kreis
      FROM Person p, geimpft g
      WHERE p.id=g.person
        AND p.SterbeDatum IS NULL))
```

Aufgabe 11 (Anfragen (6) [5 Punkte])
Geben Sie einen Algebra-Ausdruck oder Baum an, der die Anfrage aus der vorhergehenden Aufgabe beantwortet.
Sie können den Ausdruck oder Baum wahlweise hier (als ASCII-Art im Editor) abgeben, oder in Aufgabe 12 als Grafik oder Foto hochladen.

Lösung



Aufgabe 12 (Anfragen (wahlweise: Bäume als Upload) [0 Punkte])
Laden Sie hier wahlweise die Algebra-Bäume für Aufgabe 6 ("Anfragen (1)"), Aufgabe 9 ("Anfragen (4)") und Aufgabe 11 ("Anfragen (6)") als Grafik/Fotos hoch (alle in einer Datei oder einzeln).

Aufgabe 13 (Anfragen (7) [6 Punkte])
Betrachten Sie die folgende Frage:
Gesucht werden die IDs aller noch lebenden Personen, die *vollständig geimpft* sind, d.h., die mindestens drei Impfungen bekommen haben, und seit der dritten Impfung mindestens 14 Tage vergangen sind.

Warum ergibt die untenstehende SQL-Anfrage eine *falsche* Ergebnismenge (2P), und wie muss sie korrekt lauten (4P)?
(Hinweis: mit SYSDATE erhält man das heutige Datum, und Minus angewandt auf Datumswerte ergibt die Anzahl Tage dazwischen, d.h., der SQL-Ausdruck in der 5. Zeile ist in sich richtig)

```
SELECT person
FROM geimpft
WHERE person in (SELECT id FROM person WHERE sterbedatum IS NULL)
GROUP BY person
HAVING count (*) >= 3
   AND SYSDATE - MAX(datum) >= 14
```

Lösung Wenn eine Person bereits vier Impfungen erhalten hat (im Beispiel Elsa Müller), wird bei `max(datum)` das Datum der 4. Impfung verwendet. Im Fall von Elsa Müller liegt dieses zum Klausurzeitpunkt noch keine 14 Tage zurück, und sie würde im Ergebnis fehlen.
Am besten wählt man gleich nur diejenigen Impfungen aus, die mindestens 14 Tage zurückliegen, und schaut, ob es mindestens dreie sind:

```
SELECT person
FROM geimpft
WHERE person in (SELECT id FROM person WHERE sterbedatum IS NULL)
  AND SYSDATE - datum >= 14
GROUP BY person
HAVING count (*) >= 3
```

Es geht auch mit einer Aufspaltung des HAVING, wenn man annimmt, dass bei Personen, die 4x geimpft sind, die dritte Impfung ja mindestens 14 Tage vor der vierten Impfung erfolgt sein muss:

```
SELECT person
FROM geimpft
WHERE person in (SELECT id FROM person WHERE sterbedatum IS NULL)
GROUP BY person
HAVING count (*) >= 4
   OR  (count (*) >= 3 AND SYSDATE - MAX(datum) >= 14 )
```

Aufgabe 14 (Update an der Datenbank [4 Punkte])
Markus Schmidt (MS) befindet sich laut Aufgabentext zur Zeit mit Corona im Weender Krankenhaus. Geben Sie (hier) das entsprechende Beispieltupel in Ihrer Tabelle aus Aufgabe 3 bzw. 4 an (falls Sie es dort nicht schon angegeben haben) (1P).
Der Corona-Test bei Markus Schmidt fiel heute negativ –also gesund– aus, und es geht ihm auch wieder einigermaßen gut, so dass er aus dem Krankenhaus entlassen wird.
Geben Sie die beiden SQL-Statements an, mit denen (i) der Test in die Datenbank eingetragen wird (1P), und (ii) seine Entlassung aus dem Krankenhaus gespeichert wird (2P).

Lösung formal in der Tupelschreibweise:
{Person → 'MS', Krankenhaus → 'Weender Krhs.', von → '27.2.2022'};
(die Spalten, die Nullwerte haben, muss man dann ja nicht angeben),
oder
behandeltIn('MS','Weender Krankenhaus','27.2.2022', NULL, NULL, NULL).
oder
behandeltIn('MS','Weender Krankenhaus','27.2.2022', NULL, 0, NULL).

```
INSERT INTO Test VALUES('MS', '02.03.2022', 'neg');
UPDATE behandeltIn
  SET bis = '02.03.2022'
  WHERE person='MS' AND von='27.2.2022';
  (AND krankenhaus='Weender Krankenhaus' ist nicht notwendig, da er ja
   nur in einem liegen kann)
fuer die komplexere Variante, die das Datum selber herausfindet:
UPDATE behandeltIn
  SET bis = '02.03.2022'
  WHERE person = 'MS' AND von = (SELECT max(von)
                                 FROM behandeltIn
                                 WHERE person='MS');
oder
UPDATE behandeltIn
  SET bis = '02.03.2022'
  WHERE person = 'MS' AND bis IS NULL;
```

Aufgabe 15 (Interne Auswertungsstrategie [6 Punkte])
Betrachten Sie die Anfrage aus Aufgabe 7 (Anfragen (2)) *"Geben Sie eine SQL-Anfrage an, die für jedes Krankenhaus angibt, wieviele Corona-Fälle im Dezember 2021 eingeliefert wurden, die zuvor mindestens eine Impfung erhalten hatten."*
Nehmen Sie an, dass ein Baumindex auf "geimpft.Person" vorhanden ist.
Beschreiben Sie, wie die von Ihnen in Aufgabe 7 (Anfragen (2)) angegebene Anfrage effizient ausgewertet werden kann. Geben Sie einen O-Kalkül-Ausdruck an, der Ihre Überlegung beschreibt.

Lösung

- Die Relation "behandeltIn" enthält weniger Tupel als "geimpft", also läuft die äußere Schleife über "behandeltIn", und macht einen Semijoin-Zugriff auf "geimpft", bei dem der Index benutzt wird.

- Diese ist am besten nach nach dem "von"-Datum geordnet gespeichert oder indexiert, oder nach (Krankenhaus,von) gruppiert indexiert/geordnet abgelegt, so dass man den Filter auf Dezember gleich benutzen kann, um garnicht alle Fälle durchgehen zu müssen.
- Wenn nach Krankenhaus gruppiert-geordnet: alle Tupel durchgehen, für jede Person-ID über den Baum-Index in "geimpft" zugreifen. Wenn man die ID auf der Blattseite nicht findet, weiter mit der nächsten ID, sonst die Tupels auf "früher" prüfen, wenn man das erste erfüllende gefunden hat, count++ und weiter.
  Dann Krankenhaus-Name und count ausgeben.
- Wenn nicht geordnet, eins nach dem anderen durchgehen, wenn das Krankenhaus "neu" ist, count_k anlegen; in jedem Fall wie oben nachschauen und ggf. count_k++. Am Ende alle Krankenhaus-Namen und counts ausgeben.
- $O(\text{behandeltIn}) \cdot O(\log(\text{geimpft}))$

Aufgabe 16 (Eine etwas kompliziertere SQL-Anfrage [6 Punkte])
Es soll untersucht werden, welcher Impfstoff als Booster (d.h. bei der dritten oder vierten Impfung) die beste Wirksamkeit hat.
Dazu soll für jeden Impfstoff berechnet werden, wieviele Prozent der mit diesem Impfstoff geboosterten Personen *später* mit einer Corona-Diagnose in ein Krankenhaus eingeliefert wurden.

(Je nachdem, wie Sie die Aufgabe lösen, können Sie, wenn eine Person mit unterschiedlichen Impfstoffen dritt- und viertgeimpft wurde, nur eine oder beide als Booster-Impfungen zählen, das macht es ggf. einfacher; geben Sie an, welche Variante Sie verwenden)

Lösung

```
SELECT dritt.impfstoff, beh.k/dritt.i * 100
FROM
  (SELECT i3.impfstoff, count(distinct person) as i
   -- falls jemand damit die 3. und 4. Impfung bekommen hat
   -- wenn 3. und 4. Impfung mit verschiedenen Impfstoffen,
   --   zählt es für beide als Booster
   FROM geimpft i1, geimpft i, geimpft i3
   WHERE i1.person=i2.person
     AND i2.person= i3.person
     AND i1.datum < i2.datum
     AND i2.datum < i3.datum
   GROUP BY i3.impfstoff
  ) dritt,
  (SELECT i3.impfstoff, count(distinct b.person) as k
    -- person könnte ausserdem mehrmals eingeliefert worden sein
   FROM geimpft i1, geimpft i, geimpft i3, behandeltIn b
   WHERE i1.person=i2.person
     AND i2.person= i3.person
     AND i3.person = b.person
     AND i1.datum < i2.datum
     AND i2.datum < i3.datum
```

```
    AND i3.datum < b.von
   GROUP BY i3.impfstoff
  ) beh
  WHERE dritt.impfstoff = beh.impfstoff

es geht auch mit einem outer join:
SELECT impfstoff, count(distinct i.person)/count(distinct b.person) * 100
FROM
  (SELECT i1.person, i3.impfstoff, i3.datum
   FROM geimpft i1, geimpft i, geimpft i3
   WHERE i1.person=i2.person
     AND i2.person= i3.person
     AND i1.datum < i2.datum
     AND i2.datum < i3.datum
  ) i
  LEFT OUTER JOIN
  behandeltIn b
  ON i.person = b.person
  WHERE datum < b.von
  GROUP BY impfstoff

Eine andere Möglichkeit ist, statt der dritten die späteste Impfung zu suchen,
wenn diese mindestens die dritte ist. Dann doppelt oder outer join wie oben:

SELECT impfstoff, count(distinct boo.person)/count(distinct b.person) * 100
FROM
(SELECT i1.impfstoff, i1.person
 FROM geimpft i1
 WHERE datum = (SELECT max(datum)
                FROM geimpft i2
                WHERE i2.person = i1.person
                GROUP BY i2.person
                HAVING count(*) > 2)
) boo
 LEFT OUTER JOIN
  behandeltIn b
  ON boo.person = b.person
  WHERE datum < b.von
  GROUP BY impfstoff
```