Klausur Datenbanken
Wintersemester 2023/2024
Prof. Dr. Wolfgang May
1. März 2024, 10:30-12:xx Uhr
Bearbeitungszeit (Papier): 90 Minuten (Ilias: 100 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))	16	13
Aufgabe 2 (ER-Modell (ASCII-Art-Modus))	0	0
Aufgabe 3 (Transformation in das Rel. Modell (Upload))	14	9
Aufgabe 4 (Transformation in das Rel. Modell (ASCII))	0	0
Aufgabe 5 (Relationales Modell: CREATE TABLE)	4	3
Aufgabe 6 (Anfragen (1))	5	4
Aufgabe 7 (Anfragen (2))	3	2
Aufgabe 8 (Anfragen (3))	4	1
Aufgabe 9 (Anfragen (4))	4	3
Aufgabe 10 (Anfragen (5))	5	1
Aufgabe 11 (Anfragen (6))	5	1
Aufgabe 12 (Anfragen (7))	5	1
Aufgabe 13 (Anfragen (8))	6	1
Aufgabe 14 (Anfragen (wahlweise: Bäume als Upload))	0	0
Aufgabe 15 (Update an der Datenbank)	4	2
Aufgabe 16 (Eine etwas kompliziertere SQL-Anfrage)	6	1
Aufgabe 17 (Entwurfsdiskussion: künstlich erzeugte IDs)	9	5
Aufgabe 18 (Anfragen)	0	0
Aufgabe 19 (Anfragen (XXXX))	0	0
Aufgabe 20 (Anfragen (XXXX))	0	0
Summe	90	47

Note:

Themenstellung: Schiffskreuzfahrt-Veranstalter

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll die Datenbank für ein *Schiffskreuzfahrt-Unternehmen* entwickelt werden.

Im ER-Diagramm und im relationalen Modell bereits vorgegebener Teil:

1. Jedes der Schiffe hat einen Namen. Für jedes Schiff ist gespeichert, wie lang es ist.

   Die "Neptun" ist 330m lang, die "Saturn" ist 220m lang.

2. Auf jedem Schiff gibt es viele Kabinen, von denen jede eine auf diesem Schiff eindeutige Nummer hat, einer Kabinenklasse zugeordnet ist (Innen-, Außen-, Balkon-, Familienkabine, ...) und eine Größe (in Quadratmetern) hat.
   Auf der "Neptun" sind die Kabinen 200-219 Innenkabinen (Größe: 14qm), 220-239 Außenkabinen (Größe: 12qm, 12 qm, 13 qm, 13 qm, der Rest 14 qm) ... und viele weitere ... , 500-509 sind Familienkabinen (20qm, 21qm, ..., 21qm, 20qm), 510-539 Innenkabinen, 530-559 Balkonkabinen, usw.
   Auf der kleineren "Saturn" sind die Kabinen 200-214 Innenkabinen, 216-235 Außenkabinen usw.

3. Das Unternehmen bietet *Reisen* auf diesen Schiffen an. Jede Reise findet mit einem bestimmten Schiff statt, beginnt an einem bestimmten Datum an einem bestimmten Ort, und endet an einem anderen Datum, an einem Ort (das kann derselbe sein, von dem sie auch gestartet ist, oder ein anderer).

4. Jeder Ort wird durch seinen Namen und das Land, zu dem er gehört, identifiziert.

   Die Reise N2412 mit der "Neptun" beginnt am 16.5.2024 in Hamburg (Deutschland) und endet am 30.5. in Malaga (Spanien, Landescode "E").

   Die Reise N2413 mit der "Neptun" beginnt am 30.5. in Malaga (Spanien) und endet am 13.6. in Venedig (Italien).

   Die Reise S2415 mit der "Saturn" beginnt am 18.5. in Kiel (Deutschland) und endet am 25.5. wieder in Kiel.

5. Die Reisen werden von Personen gebucht.
   Für jede *Person* ist der Name (Annahme: Namen sind eindeutig) gespeichert, die Anschrift, und das Geburtsdatum. Von allen mitreisenden Personen werden diese Daten ebenfalls gespeichert.

In der Klausur soll das ER-Diagram und das Relationale Modell um die folgenden Daten erweitert werden:

6. Für jede Reise ist für jede dabei angebotene Kabinenklasse der Preis festgelegt und in der Datenbank gespeichert.

   Für die Reise N2412 kostet eine Innenkabine 2699.00 EUR, eine Balkonkabine 4199.00 EUR, eine Familienkabine 4599.00 EUR.

   Für die Reise N2413 kostet eine Innenkabine 2599.00 EUR, eine Balkonkabine 3999.00 EUR, eine Familienkabine 4399.00 EUR.

7. Im Lauf der *Reise* legt das Schiff zu Besichtigungen etc. an verschiedenen Orten an. An jedem Tag legt es höchstens an einem Ort an (an manchen Tagen auch an keinem).

   Einige Orte der Reise N2412: 16.5.2023 Hamburg (D), 18.5. London (GB), 19.5. Brighton (GB), 20.5. Jersey (GB), 23.5. Porto (P), 24.5. Lissabon (P), 25.5. Lissabon (P), 26.5. Faro (P), 28.5. Gibraltar (GIB), 30.5. Malaga (E).

8. An den einzelnen Tagen werden außer den üblichen Stadtrundfahrten weitere (einzeln zu buchende und zu bezahlende) Ausflüge angeboten. Diese können bei verschiedenen Reisen an demselben Ort auch unterschiedlich sein.

   Für jeden Ausflug auf einer Reise ist die Bezeichnung und der Preis gespeichert (dieser kann auf verschiedenen Reisen unterschiedlich sein).

   Die Ausflugsangebote auf der Reise N2412 sind z.B. am 23.5. die Besichtigung einer Portweinkellerei (100 EUR), am 24.5. ein Konzert von Taylor Swift (250.00 EUR), am 24.5. und am 25.5. "Schlösser von Sintra" (50 EUR).

9. Für jede Reisebuchung wird eine interne ID vergeben. Es wird gespeichert, auf welchen Namen die Buchung läuft, und welche Kabinenklasse gebucht ist. Eine konkrete Kabine wird der Buchung erst später zugeordnet. Außerdem werden die Namen der in der gebuchten Kabine reisenden Personen gespeichert (man kann kann auf diese Weise auch mehrere Kabinen für weitere Personen buchen).

   Ausserdem wird gespeichert, welche Ausflüge zu dieser Buchung gebucht sind, und wieviele Tickets (für ihre mitreisenden Personen). Dies kann bei der Buchung geschehen, oder -soweit verfügbar- auch noch vor oder während der Reise.

10. Markus Schmidt, geboren am 1.7.1990 wohnt in der Langen Straße 42 in Göttingen und hat mit der Buchungs-ID 123456789 für die Reise N2412 eine Familienkabine für sich und seine Frau Andrea Schmidt und die Kinder Lisa und Leon gebucht. Ihnen wurde bereits die Kabine 505 zugeordnet. Er hat zu dieser Buchung auch zwei Tickets für die Portweinkellerei am 23.5. gebucht.

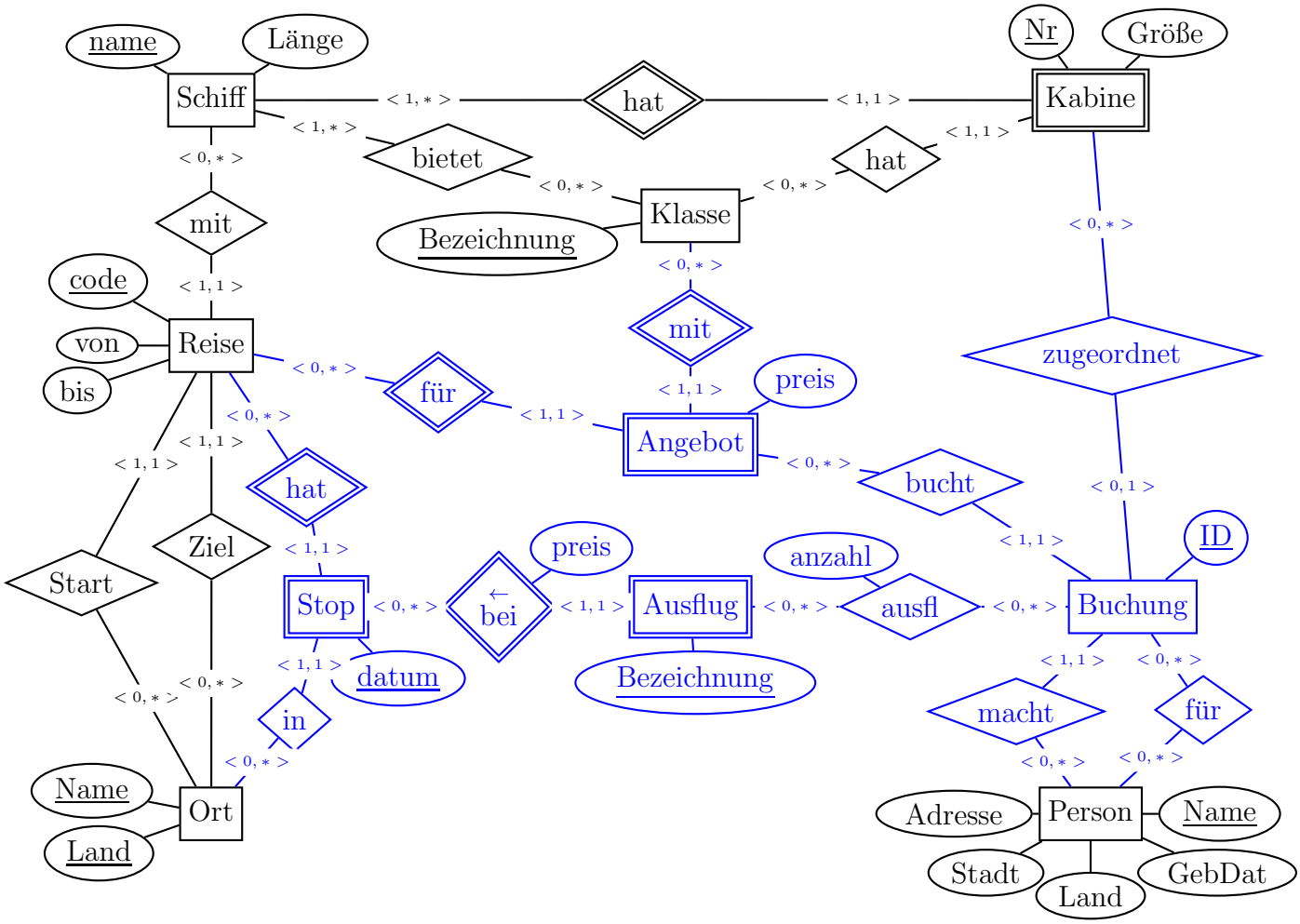
    Mit der Buchungs-ID 213456789 hat Markus Schmidt für dieselbe Reise (d.h. für N2412) eine Balkonkabine für die Großeltern Ernst und Erna Schmidt gebucht. Ihnen wurde noch keine Kabine fest zugeordnet.

11. Man kann auch zwei oder mehr direkt aufeinanderfolgende Reisen buchen: Fritz Müller (geboren am 9.9.1949, wohnt in der Hauptstraße 123, Hamburg) hat für sich und seine Frau Frieda Müller (4.4.1954) mit der Buchungs-ID 333333333 bzw 333333339 für die Reisen N2412 und N2413 jeweils eine Balkonkabine gebucht. Ihnen wurde noch keine Kabine fest zugeordnet.

    Er hat zu der Buchung 333333333 auch zwei Tickets für die "Schlösser von Sintra" am 25.5. gebucht.

Aufgabe 1 (ER-Modell (Foto oder PDF-Upload) [16 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.)

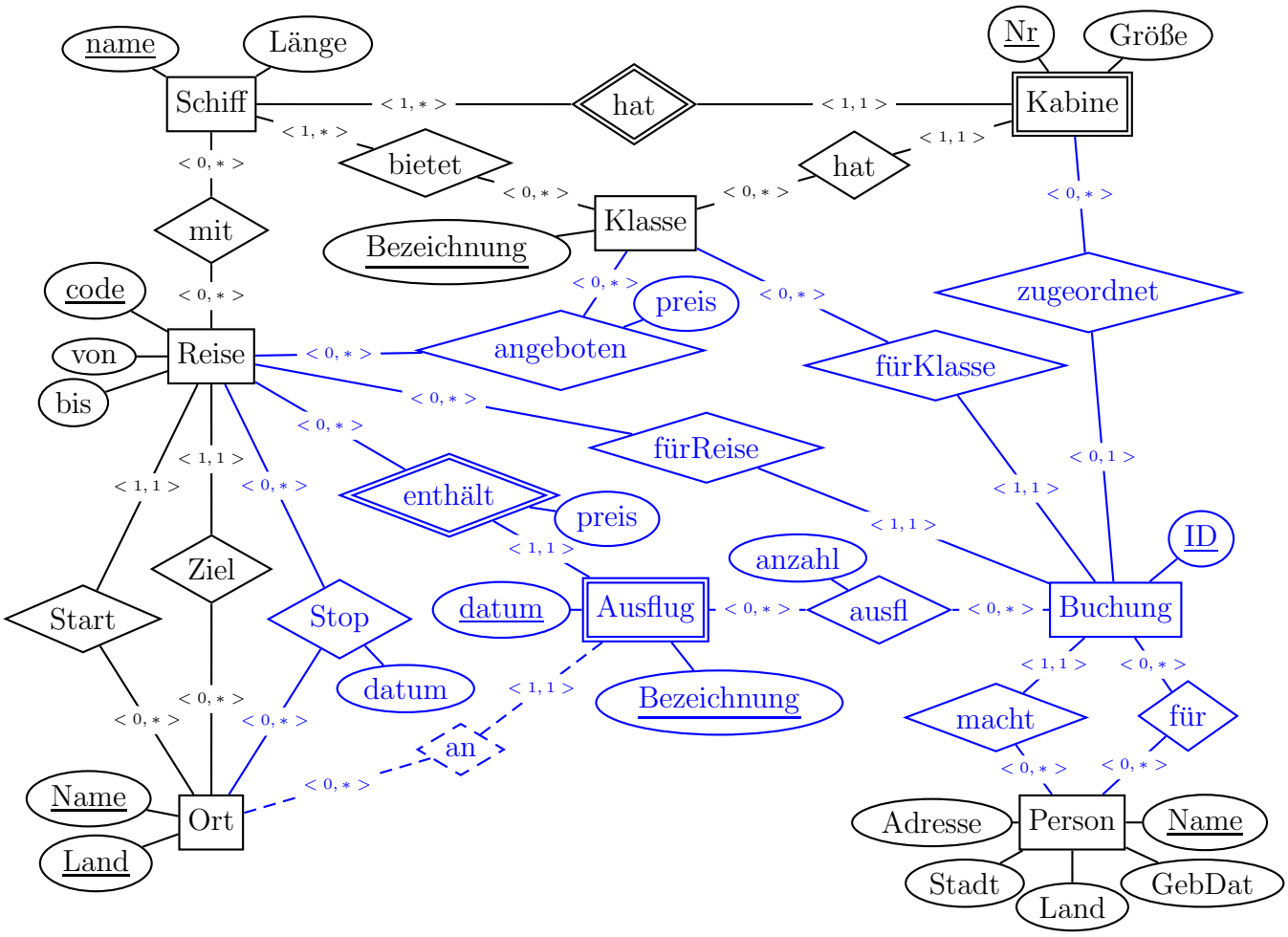
- **Wenn Sie die ER-Modell-Aufgabe als pdf/png/jpg- oder Foto-Upload bearbeiten wollen, machen Sie dies HIER,**
- **wenn Sie die Aufgabe stattdessen als ASCII-Art im Editor bearbeiten wollen, machen Sie dies in Aufgabe 2.**



Lösung Die beste Lösung ist obenstehend eingezeichnet. "Angebot" ist ein reifizierter Entitätstyp, mit dem jede "Buchung" verbunden wird.
"Stop" ist auch ein reifizierter Entitätstyp, mit den die optionalen Ausflüge verbunden werden. Hinweis: der identifizierende Entitätstyp des Stops ist dabei nur "Reise", und "Datum" ist lokaler Schlüssel.
(nicht weak zu "Ort": eine Reise kann an jedem Datum nur an einem Ort sein, aber sie kann mehrere Tage an einem Ort bleiben, oder später noch einmal an einem Ort stoppen, wo sie vorher schon einmal war)

Mit "Ausflug" als schwachem Entitätstyp garantiert man, dass er an eine bestimmte Reise und das Datum eines bestimmten Stops gebunden wird. Das Attribut "Preis" kann man dann wahlweise an die Beziehung "bei" oder direkt an den Ausflug (der ja über "bei" identifiziert wird) hängen.

Ebenfalls korrekte, aber nicht optimale Variante ohne die reifizierten Entitätstypen "Angebot" und "Stop":



Lösung

- Es gibt in dieser Variante nur eine binäre Beziehung "angeboten" mit einem Attribut "Preis" zwischen "Reise" und "Klasse". Dafür gibt es zwei Kanten "fürReise" bzw. "fürKlasse" von "Buchung" zu "Reise" bzw. "Klasse".

  Damit wäre aber als Instanz von "Buchung" eine Kombination (Reise, Klasse) möglich, die garnicht angeboten wird – die Integritätsbedingung würde fehlen.

  Beides führt dennoch schlussendlich zur selben relationalen Modellierung, wenn man daran denkt, bei "Angebot" den Primary Key dann richtig zu machen, und Buchung darauf zu beziehen.

- Analog ist "Stop" nur eine binäre Beziehung zwischen "Reise" und "Ort". Die Ausflüge können dann aber nur mit der entsprechenden Reise assoziiert werden. Dabei fällt auf,

dass das Attribut "Datum" schon irgendwie doppelt ist, und obwohl ein Ausflug zu einem Stop gehört, dieser Zusammenhang nicht wirklich da ist.

- Die Verbindung zwischen "Ausflug" und "Ort" ist optional und etwas redundant, da sich über den Zusammenhang mit Reise und Datum der Ort sowieso zwangsläufig ergibt. Allerdings könnte man damit schon Ausflüge planen, bevor die konkreten Reisen dazu entworfen werden.

- Wichtig ist, dass "datum" ein Attribut zu "Ausflug" ist, sonst wäre bei einer Buchung des Ausflugs "Schlösser von Sintra" zur Reise N2412 ggf. nicht klar, zu welchem Datum (24.5. oder 25.5.) sie gehört.

- Exakt gelesen verlangt die Aufgabenstellung, dass der Preis eines Ausflugs für jede Reise unterschiedlich sein kann. Dies kann man in dieser Variante auf zwei Arten erreichen:

  - (wie eingezeichnet) "Ausflug" weak bzgl. Reise mit <1,1>-Kardinalität (wie an sich auch in der Modellierung mit reifizierten Stops). Dann kann das "Preis"-Attribut wahlweise dem Entittatstyp "Ausflug" oder dem Beziehungstyp "enthält" zugeordnet sein,
  - oder Ausflug nicht weak, mit <1,*>-Kardinalität bzgl. "enthält" zu "Reise" und dem Attribut "Preis" an dem Beziehungstyp "enthält".
  - (keine gute Lösung): Ausflug *nicht* weak, mit <1,1>-Kardinalität bzgl. "enthält" zu "Reise" (und dem Attribut "Preis" egal wo): wenn dann ein Ausflug an einem Tag für mehrere an diesem Tag an diesem Ort seiende Reisen angeboten wird - dann müssen daraus mehrere Ausflüge mit verschiedenen Namen werden. (Nach diesem ER-Design darf im Relationalen Entwurf "Reise" nicht im Schlüssel von "Ausflug" sein!)

- Eine weitere (etwas falsche) Variante wäre, die Verbindung zwischen Ausflug und Reise ganz wegzulassen, und stattdessen den Ausflug an einen Ort zu binden, und (in der Praxis dann durch eine Anfrage) festzulegen, dass alle Ausflüge an einem Tag für alle Reisen, die an diesem Tag in diesem Ort sind, buchbar sind.
  Dagegen spricht das Beispieltupel mit dem Konzert von Taylor Swift, das am 24.5.2024 *abends* stattfindet, und für Reisen, die nur am 24. tagsüber in Lissabon sind, und abends schon weiterfahren, nicht buchbar sein sollte.

  Diese Variante würde außerdem das Benutzerinterface verlangsamen (jedesmal raussuchen, welche Ausflüge man einem Buchenden einer Reise anbietet).

- An dem Beispiel sieht man, dass man durch die fehlende Reifikation und den fehlenden Bezug (unnötig) einen *zyklischen* Zusammenhang Ort-Datum-Reise-Datum-Ausflug-an-Ort einhandelt, der ggf. zu Inkonsistenzen führen kann.

  Reifikation kann häufig helfen, solche Zusammenhänge eben in einem Entitätstyp zu reifizieren und damit in den Griff zu bekommen.

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) [14 Punkte])
Vervollständigen Sie in dieser Aufgabe das relationale Modell. Gegeben sind die folgenden Relationen:

Schiff

Name	Länge
Neptun	330
Saturn	220
:	:

Klasse

Bezeichnung
Innen
Außen
Balkon
Familie
:

Kabine

Nr	Schiff	Klasse	qm
200	Neptun	Innen	14
201	Neptun	Innen	14
220	Neptun	Außen	12
500	Neptun	Familie	20
200	Saturn	Innen	12
:	:	:	:

Ort

Name	Land
Hamburg	D
Kiel	D
London	GB
Lissabon	P
Malaga	E
:	:

Reise

Code	Schiff	von	bis	Start	StartL	Ziel	ZielL
N2412	Neptun	16.5.2024	30.5.2024	Hamburg	D	Malaga	E
N2413	Neptun	30.5.2024	13.6.2024	Malaga	E	Venedig	I
S2415	Saturn	30.5.2024	13.6.2024	Kiel	D	Kiel	D
:	:	:	:	:	:	:	:

Person

Name	GebDat	Adresse	Stadt	Land
Markus Schmidt	1.7.1990	Lange Straße 42	Göttingen	D
Andrea Schmidt	...	Lange Straße 42	Göttingen	D
Lisa Schmidt	...	Lange Straße 42	Göttingen	D
Leon Schmidt	...	Lange Straße 42	Göttingen	D
Ernst Schmidt	...	...	...	D
Erna Schmidt	...	...	...	D
Fritz Müller	9.9.1949	Hauptstraße 123	Hamburg	D
Frieda Müller	4.4.1954	Hauptstraße 123	Hamburg	D
:	:	:	:	:

Geben Sie die noch fehlenden Tabellen (mit Attributen, Schlüsseln, Fremdschlüsseln etc.) für die Preise, Zwischenstops, Ausflüge sowie alles, was Buchungen betrifft, 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

Angebote

Reise	Klasse	Preis
N2412	Innen	2699
N2412	Balkon	4199
N2412	Familie	4599
N2413	Innen	2599
N2413	Balkon	3999
N2413	Familie	4399
:	:	:

Stops

Reise	Datum	Ort	Land
N2412	16.5.2024	Hamburg	D
N2412	18.5.2024	London	GB
N2412	23.5.2024	Porto	P
N2412	24.5.2024	Lissabon	P
N2412	25.5.2024	Lissabon	P
N2412	26.5.2024	Faro	P
N2412	28.5.2024	Gibraltar	GIB
N2412	30.5.2024	Malaga	E
N2413	30.5.2024	Malaga	E
N2413	31.5.2024	Almeria	E
N2413	13.6.2024	Venedig	I
S2415	18.5.2024	Kiel	D
:	:	:	:
S2415	30.5.2024	Kiel	D
:	:	:	:

Ausflug

Reise	Datum	Bezeichnung	Preis
N2412	23.5.2024	Portweinkellerei	100
N2412	24.5.2024	Taylor Swift	250
N2412	24.5.2024	Sintra	50
N2412	25.5.2024	Sintra	50
:	:	:	:

Buchung

ID	Person	Reise	Klasse	Schiff	Kabine
123456789	Markus Schmidt	N2412	Familie	Neptun	505
213456789	Markus Schmidt	N2412	Balkon	Neptun	null
333333333	Fritz Müller	N2412	Balkon	Neptun	null
333333339	Frieda Müller	N2413	Balkon	Neptun	null
:	:	:	:	:	:

Reisende

Buchung	Person
123456789	Markus Schmidt
123456789	Andrea Schmidt
123456789	Lisa Schmidt
123456789	Leon Schmidt
213456789	Ernst Schmidt
213456789	Erna Schmidt
333333333	Fritz Müller
333333333	Frieda Müller
333333339	Fritz Müller
333333339	Frieda Müller
:	:

Ausflugsbuchung

Buchung	Reise	Datum	Ausflug	Anzahl
123456789	N2412	23.5.2024	Portweinkellerei	2
333333333	N2412	24.5.2024	Sintra	2
:	:	:	:	:

Hinweise:

- Bei "Stop" muss der Primary Key (Reise, Datum) und nicht (Reise, Ort, Land) sein: eine Reise kann an jedem Datum nur an einem Ort sein, aber sie kann mehrere Tage an einem Ort bleiben, oder später noch einmal an einem Ort stoppen, wo sie vorher schon einmal war.

- Buchung: Um den Entwurf "korrekt" zu machen, muss als FK für die Kabine die Spalte "Schiff" dazugenommen werden.
  Damit hat die Tabelle eine nicht auf dem Schlüssel basierende funktionale Abhängigkeit (=Zusammenhang, der schon in einer anderen Tabelle, nämlich in "Reise", gespeichert ist) Reise→Schiff innerhalb der Tabelle, was man eigentlich nicht haben möchte.

- Ausflugsbuchung: Um den Entwurf "korrekt" zu machen, muss als FK für den Ausflug die Spalte "Reise" dazugenommen werden. Damit hat die Tabelle eine nicht auf dem Schlüssel basierende funktionale Abhängigkeit Buchung→Reise (die schon in einer anderen Tabelle, nämlich in "Buchung", gespeichert ist) innerhalb der Tabelle, was man eigentlich nicht haben möchte.
  Dies hat auch die Folge, dass "Reise" hier nicht Teil des Primary Key ist.

- Beides sind Nachteile von künstlich erzeugten Keys, siehe Aufgabe 17 (Entwurfsdiskussion)).

FKs:
Angebote(Reise)→Reise(Code)
Angebote(Klasse)→Klasse(Bezeichnung)
Stop(Reise)→Reise(Code)
Stop(Ort,Land)→Ort(Name,Land)
Ausflug(Reise,Datum)→Stop(Reise,Datum) (*)
Buchung(Person)→Person(Name)
Buchung(Reise)→Reise(Code)
Buchung(Klasse)→Klasse(Bezeichnung)
Buchung(Reise, Klasse)→Angebot(Reise, Klasse) (*)
Buchung(Kabine,Schiff)→Kabine(Nr,Schiff)
Ausflugsbuchung(Buchung)→Buchung(ID)
Ausflugsbuchung(Ausflug,Reise,Datum)→Ausflug(Bezeichnung,Reise,Datum)
Reisende(Buchung)→Buchung(ID)
Reisende(Person)→Person(Name)
Die beiden mit (*) bezeichneten FKs werden vom ER-Modell explizit impliziert, wenn "Angebot" und "Zwischenstop" reifiziert sind. Wenn nicht, muss man beim Entwurf des Relationalen Modells eben aufpassen, dass es auch FKs auf aus Beziehungen entstandenen Tabellen geben kann.

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 [4 Punkte])
Geben Sie das `CREATE TABLE`-Statement für Ihre Tabelle, in der abgespeichert wird, wer welche Reisen gebucht hat, so vollständig wie möglich an.

Lösung

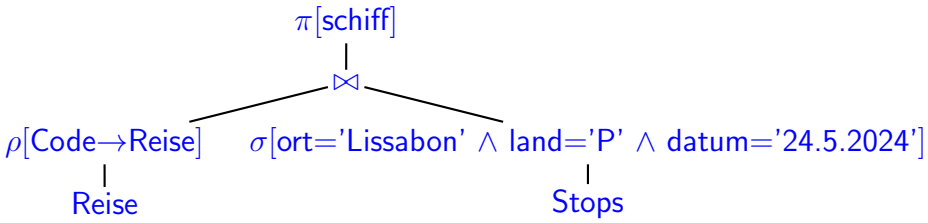
```
CREATE TABLE Buchung     1.5P Basis
(ID      NUMBER PRIMARY KEY,       1/2P
 Person  VARCHAR2(50) NOT NULL REFERENCES Person(Name),  Ref 1/2P
 Reise   VARCHAR2(5)  NOT NULL ,   not null: 1/2P
 Klasse  VARCHAR2(10) NOT NULL ,
 Schiff  VARCHAR2(10),
 Kabine  NUMBER,
 FOREIGN KEY (Reise, Klasse) REFERENCES Angebote(Reise, Klasse),
     1P; 1/2P wenn die beiden einspaltigen FKs da sind
 FOREIGN KEY (Schiff, Kabine) REFERENCES Kabine(Schiff, Nummer));
     1/2P (der auch bei Folgefehler bzgl Aufg. 3 wegen Vereinfachung abgezogen wi
```

Da die ID als (halb-künstlicher) PK eingeführt wurde, muss man die "notwendigen" Spalten explizit auf NOT NULL setzen.
Der kombinierte Fremdschlüssel Buchung(Reise, Klasse) folgt bei dem ER-Modell der Referenzlösung mit reifiziertem Entitätstyp "Angebote" "von selbst", während man sonst daran denken muß, dass man ihn (dann auf den Schlüssel der aus der Beziehung "wird angeboten" entstandene Tabelle) als Integritätsbedingung hinzufügen sollte.
Einspaltige REFERENCES für "Reise" und "Klasse" benötigt man dann nicht.

Aufgabe 6 (Anfragen (1) [5 Punkte])
Geben Sie eine SQL-Anfrage *und* einen Ausdruck oder Baum der relationalen Algebra an, die die Namen derjenigen Schiffe ausgeben, die am 24.5.2024 in Lissabon (Portugal) sind. (Algebra-Baum entweder hier oder als Grafik in Aufg. 14)

Lösung

```
SELECT Schiff  -- kein DISTINCT notwendig, da keine Duplikate entstehen können
FROM Stops, Reise
WHERE Ort = 'Lissabon' AND land='P' AND datum='24.5.2024'
  AND reise=code
```



Aufgabe 7 (Anfragen (2) [3 Punkte])
Geben Sie eine SQL-Anfrage an, die für jede Person, die mindestens einmal eine Reise gebucht hat, ausgibt, wieviel Umsatz das Unternehmen mit von dieser Person durchgeführten *Reisebuchungen* (*ohne die Ausflüge*) insgesamt gemacht hat (einschließlich schon bestehender Buchungen, deren Reisedatum in der Zukunft liegt).

Lösung

```
SELECT Person, SUM (Preis)
FROM Buchung b, Angebote a
WHERE b.reise = a.reise AND b.klasse = a.klasse
GROUP BY Person
```

Aufgabe 8 (Anfragen (3) [4 Punkte])
Geben Sie eine SQL-Anfrage (und in der *nächsten Aufgabe* einen Algebra-Ausdruck oder -Baum) an, die die Namen derjenigen Schiffe ausgibt, mit denen noch nie eine Reise durchgeführt wurde, die einen Stop in Großbritannien gemacht hat.

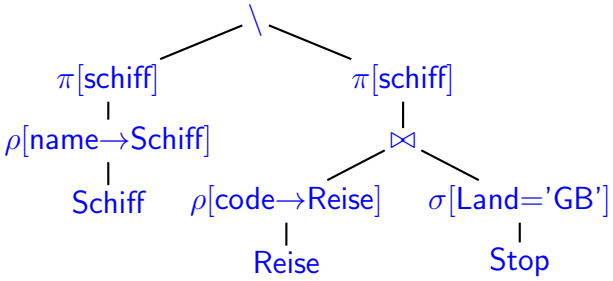
Lösung

```
SELECT name
FROM Schiff s
WHERE NOT EXISTS
 (SELECT *
  FROM Reise r, Stops st
  WHERE r.schiff = s.name and r.code = st.reise
    AND st.Land = 'GB')
```

```
(SELECT name
 FROM Schiff s)
MINUS
(SELECT Schiff
 FROM Reise r, Stops st
  WHERE r.code = st.reise AND st.Land = 'GB')
```

Aufgabe 9 (Anfragen (4) [4 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 14 als Grafik oder Foto hochladen.

Lösung



Aufgabe 10 (Anfragen (5) [5 Punkte])
Geben Sie eine SQL-Anfrage (und in der *nächsten Aufgabe* einen Algebra-Ausdruck oder -Baum) an, die die Namen derjenigen Personen ausgibt, die schon mit jedem der Schiffe, auf denen es Balkonkabinen gibt, (mindestens) einmal eine Reise gebucht haben.

Lösung
Hinweis: da nur wenige die Spalte "Schiff" zu "Buchung" dazugenommen haben, wird die Lösung für diesen etwas umständlicheren Fall, wo man "Reise" noch dazunehmen muss, gegeben:

```
-- mit Schiff<->Klasse aus Kabine
SELECT name
FROM Person p
WHERE NOT EXISTS
  (SELECT *  -- k.schiff
   FROM Kabine k
   WHERE k.Klasse = 'Balkon'
     AND NOT EXISTS
     (SELECT *
      FROM Buchung, Reise
      WHERE b.Reise = r.code
        AND r.schiff = k.schiff
        AND b.person = p.name))
```

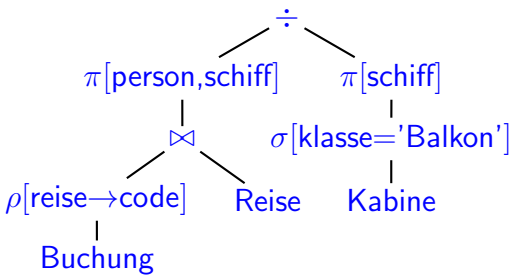
```
-- mit Schiff<->Klasse aus "Angebote"
SELECT name
FROM Person p
WHERE NOT EXISTS
  ((SELECT r1.schiff
    FROM Reise r1, Angebote a
    WHERE r1.code = a.reise
      AND a.Klasse = 'Balkon')
   MINUS
   (SELECT r2.schiff
    FROM Buchung, Reise
    WHERE b.Reise = r2.code
      AND b.person = p.name))
```

```
SELECT name
FROM Person p
WHERE (SELECT COUNT(DISTINCT Schiff)
       FROM Buchung, Reise
       WHERE b.Reise = r.code
         AND b.person = p.name
         AND Schiff IN
             (SELECT Schiff
              FROM Kabine
              WHERE Klasse = 'Balkon'))
    = (SELECT COUNT(DISTINCT Schiff)
       FROM Kabine k
       WHERE k.Klasse = 'Balkon')
```

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 14 als Grafik oder Foto hochladen.

Lösung



Aufgabe 12 (Anfragen (7) [5 Punkte])
Geben Sie eine SQL-Anfrage (und in der *nächsten Aufgabe* einen Algebra-Ausdruck oder -Baum) an, die die Namen derjenigen Personen ausgibt, die schon mindestens einmal eine Reise gebucht haben, und bei jeder der von der jeweiligen Person gebuchten Reisen mindestens ein Ort in Portugal (P) angelaufen wurde.

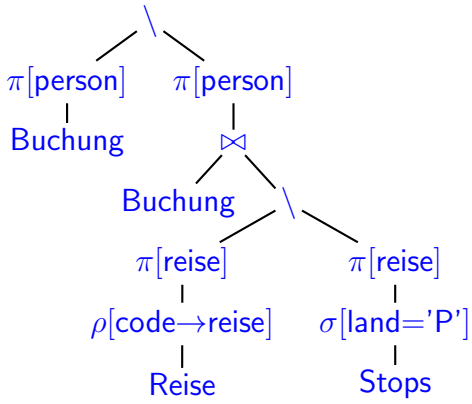
Lösung

```
SELECT DISTINCT Person
FROM Buchung b1
WHERE NOT EXISTS
  (SELECT *
   FROM BUCHUNG b2
   WHERE b1.person = b2.person
     AND NOT EXISTS
         (SELECT *
          FROM Stops s
          WHERE s.reise = b2.reise
            AND s.Land='P'))
```

Aufgabe 13 (Anfragen (8) [6 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 14 als Grafik oder Foto hochladen.

Lösung



Aufgabe 14 (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)") und Aufgabe 13 ("Anfragen (8)") als Grafik/Fotos hoch (alle in einer Datei oder einzeln).

Aufgabe 15 (Update an der Datenbank [4 Punkte])
Markus Schmidt erweitert seine Reisebuchung 123456789 um ein weiteres Ticket für die Besichtigung der Portweinkellerei am 23.5., und erweitert die Reisebuchung 213456789 ebenfalls um ein solches Ticket (Leon und Opa gehen auch mit, und Oma passt auf Lisa auf).
Geben Sie die Updates an, die an der Datenbank in der gegebenen Situation ausgeführt werden müssen.

Lösung

```
UPDATE Ausflugsbuchung
  SET Anz = Anz +1
  WHERE Buchung = 123456789 AND datum = "23.5.2024"
    AND Ausflug = "Portweinkellerei";                          2.5P
INSERT INTO Ausflugsbuchung
VALUES (213456789, "N2412", "23.5.2024", "Portweinkellerei", 1);     1.5P

Bzw., da in den entsprechenden Benutzerdialog nur Buchungs-Nummer,
Bezeichnung des Ausflugs (incl.\ Datum) und Anzahl Tickets vorkommen, muesste
die Anwendung das Statement

INSERT INTO Ausflugsbuchung
VALUES (213456789, (SELECT Reise
                    FROM Buchung
                    WHERE ID=213456789), "23.5.2024", "Portweinkellerei", 1);

ausfuehren.
```

(nur ein Insert reicht nicht, sonst bleibt das alte Tupel drin und das Insert würde den Primary Key verletzen und abgelehnt).

Innerhalb SQL direkt gibt es keine Möglichkeit, in dem Statement direkt zu prüfen, ob schon ein Tupel vorhanden ist, oder nicht. Das muss im umgebenden Programm (z.B. in Java, mit JDBC) oder innerhalb einer DB-spezifischen SQL-Erweiterung (Oracle: PL/SQL) geschehen.

Aufgabe 16 (Eine etwas kompliziertere SQL-Anfrage [6 Punkte])
Geben Sie eine SQL-Anfrage an, die folgendes löst:

Jemand möchte eine Rundreise buchen, d.h. eine Folge von *direkt zeitlich aneinander anschliessenden* Reisen (d.h., es geht nach dem Ende der einen Reise gleich am selben Tag weiter) auf einem Schiff, die in Hamburg beginnt, und nach (einer oder) mehreren Reisen auch wieder in Hamburg endet.
Die Anfrage soll dann für jede solche Lösung das Schiff, das Abreisedatum und das Rückkehrdatum ausgeben.

Lösung

```
select r1.schiff, r1.von, r2.bis
from  reise r1, reise r2
where r1.schiff = r2.schiff
  and r1.start = 'Hamburg' and r2.ziel = 'Hamburg'
  and r1.von <= r2.von
  and
  (r1.von = r2.von    --- es ist dieselbe Reise von HH nach HH
   OR not exists  -- Reise mit diesem Schiff in diesem Zeitraum,
                     die nicht unmittelbar weitergeht
    ( SELECT *
      FROM reise r3
      WHERE r3.schiff = r1.schiff
        AND r3.von >= r1.von AND r3.von < r2.von
        AND NOT EXISTS
          (SELECT *
           FROM reise r4
           WHERE r4.schiff = r3.schiff
             AND r4.von = r3.bis)))
```

Es geht auch mit dem in der Vorlesung nicht behandelten `WITH RECURSIVE` (bzw. in Oracle ähnlich mit "CONNECT BY" angebotenen) Konstrukt für rekursive Relationen bzw. transitive Hülle (7 Teilnehmer (von 98) haben dieses Konstrukt verwendet, aber nur zwei Lösungen waren strukturell richtig).

```
WITH RECURSIVE KettenreiseHHNachIrgendwo (Schiff, von, bis, Ziel, ZielL ) AS (
 SELECT Schiff, von, bis, Ziel, ZielL
   FROM Reise
   WHERE Start = 'Hamburg' AND StartL='D'
 UNION ALL
 SELECT k.Schiff, k.von, r.bis, r.Ziel, r.ZielL
  FROM KettenreiseHHNachIrgendwo k, Reise r
  WHERE k.Schiff = r.Schiff AND k.bis = r.von  -- derselbe Ort ist es dann auch
)
```

```
SELECT Schiff, von, bis
FROM KettenreiseHHNachIrgendwo
WHERE Ziel = 'Hamburg' AND ZielL='D';
```

In der Klausur war die Verwendung von ChatGPT erlaubt (musste dann angegeben werden). Von denjenigen, die bei dieser Aufgabe angegeben haben, ChatGPT verwendet zu haben, hat ChatGPT offensichtlich manchen Teilnehmenden einen Lösungsversuch wie die erste Variante mit 2 Reisen anfangend angeboten (mit verschiedenen Subquery-Bedingungen, die aber alle mehr oder weniger kaputt, wirr oder unpassend (z.B. "Zwischenreisen" komplett verbieten) waren), und manchen einen Ansatz mit WITH RECURSIVE vorgeschlagen hat, mit dem es aber die Bedingungen dann auch nicht hinbekommen hat, der aber wohl leichter zu einer richtigen Lösung zu überarbeiten gewesen wäre.

Aufgabe 17 (Entwurfsdiskussion: künstlich erzeugte IDs [9 Punkte])
Im vorgegebenen ER-Modell wird basierend auf dem Text der Aufgabenstellung der künstlich erzeugte ID-Code der Reisen als Schlüssel eingeführt.

a) Wenn es diesen Code nicht gäbe, was wäre im ER-Diagramm und im relationalen Entwurf anders, was wäre der Schlüssel des Entitätstyps "Reise", und der Tabelle "Reise"? (3P)

b) Wo hat diese Designentscheidung Auswirkungen auf Ihren Teil des Tabellen-Entwurfs? Beschreiben Sie (kurz) Vorteile sowie Nachteile. (6P)

Lösung Anmerkung: Künstlich erzeigte Schlüssel wie Reise.Code und Buchung.Nr (das man durch (Person, Buchungszeitpunkt) ersetzen könnte) - werden auch als "Surrogate Keys" (Ersatzschlüssel) bezeichnet. Insbesondere die Buchungsnummer ist auch nicht nur ein interner Identifier, sondern wird z.B. auch im Web-Interface benutzt, wenn jemand später noch Ausflüge buchen möchte oder Reklamationen hat.

a) "Reise" wäre im ER-Diagramm ein schwacher Entitätstyp, der sich über die Beziehung "mit" und das genutzte Schiff identifiziert, und die Datums-Spalte "von" als lokalen Schlüsselanteil hat.
In der Tabelle "Reise" würden damit anstatt "Code" entsprechend die Spalten "Schiff" und "von" zusammen den Primärschlüssel bilden. (3P)

b) oder a) Auswirkungen: man muss in allen Relationen, die eine Referenz auf "Reise" enthalten, den entsprechenden ggf. mehrteiligen Schlüssel als Fremdschlüssel verwenden ("Zwischenstop" und "Buchung", "Angebot", "Ausflugsbuchung"). (1P, kann auch unter (a) schon beschrieben sein)

b1) Vorteil des gewählten Entwurf mit "Code":
Überall, wo er als Foreign Key verwendet wird, hat man nur eine Spalte anstatt zwei, (1P)
daher kürzere Join-Bedingungen (die Auswertungsperformance ist ziemlich gleich, da die Datenbank automatisch Indexe anlegt) (1P)

Wenn man natürliche Schlüssel verwendet, und ein Update einen dieser Werte ändert, muss man das Update überall mitziehen (Mondial: Landescode oder ein Stadtname ändert sich), da es ON UPDATE CASCADE nicht gibt. (+1P)
(Dieses Szenario kommt in der gegebenen Datenbank wohl selten vor, weil man das zugeordnete Schiff und das Abreisedatum einer Reise nicht mehr ändern sollte, wenn die ersten Kunden gebucht haben)

b2) Nachteil: wenn eine künstliche ID (Reisecode) irgendwo als FK verwendet wird, "verdeckt" sie die sonst als Schlüsselwerte verwendeten Spalten (hier "Schiff" und "von"):

Man braucht immer eine zusätzliche Join-Operation, um die sonst als Schlüsselwerte verwendeten Spalten herauszufinden (z.B. wenn man wissen will, welche Schiffe an einem bestimmten Tag an einem bestimmten Ort Zwischenstop machen). (1P)

Teile der natürlichen Schlüsselkombination werden oft auch für andere Zwecke als Konsistenzmerkmal benötigt: "Reisecode" verdeckt in der Tabelle "Buchung" den Bestandteil "Schiff", der als Teil des FK (KabineNr, Schiff) benötigt wird (siehe Aufgaben 3/4 + 5). Man muss also die Spalte Schiff beim Design der Tabelle "Buchung" dazunehmen, um Buchung(Kabine,Schiff) $\rightarrow$ Kabine(Nr,Schiff) als Integritätsbedingung zuzusichern. (1P)

Damit ist die Tabelle formal allerdings nicht in der 2. Normalform.

Ein analoges Problem gibt es bei der Tabelle "Ausflugsbuchungen": Buchungs-ID verdeckt potentiell "Reise". Man muss streng nach Kochrezept vorgehen, und den kompletten Primary Key von Ausflug (Reise, Datum, Bezeichnung) als Spalten übernehmen. "Reise(code)" ist dann allerdings nicht Teil des Primary Key, weil der Reisecode funktional von der Buchungs-ID abhängt. Diese Integritätsbedingung kann man dennoch nicht durch einen Foreign Key überprüfen. Auch bei dem INSERT in Aufgabe 15 zeigt sich das Verdeckungs-Problem. (1P)

Weitere Aspekte, die genannt wurden, und jeweils 1P gaben:

- Risiko manueller Fehleingaben (z.B. Zahlendreher) bei automatisch generierten (durchnummerierten) Schlüsseln: da die meisten Werte wirklich in der DB vorkommen gibt, führt ein Tippfehler nicht zu einer Ablehnung, sondern zu einer Referenz auf etwas anderes.

- Finden und Unterhalten (neuer) eindeutiger IDs: Das ist aber kein Problem, weil jedes Datenbanksystem die Möglichkeit, eine durchzählende ID als key zu vergeben, anbietet In Oracle z.B. im CREATE TABLE-Statement:
  `id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY`

- Natürliche Schlüssel drücken auch Eindeutigkeitsbedingungen an diese Kombinationen aus.
  In diesem Anwendungsfall würde die Eindeutigkeit von (Schiff,von) verhindern, dass man noch einer anderen Reise dasselbe Schiff am selben Abreisetag zuordnet. Wenn man einen künstlichen Schlüssel hat, müsste man deshalb bei der Tabellendefinition noch `UNIQUE (schiff, von)` dazunehmen.

  Andererseits kann es z.B. in der Planungsphase für eine Reisesaison durchaus sein, verschiedene Alternativen für ein Schiff an demselben Abreisetag anzulagen, schrittweise mit Ausflugsangeboten zu komplettieren, und später eine zu streichen oder ein anderes Schiff zuzuordnen.

- Finden und Unterhalten (neuer) eindeutiger IDs: Das ist aber kein Problem, weil jedes Datenbanksystem die Möglichkeit, eine durchzählende ID als key zu vergeben, anbietet In Oracle z.B. im CREATE TABLE-Statement:
  id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY

- Künstliche Schlüssel können zum Datenschutz/Anonymisierung sinnvoll sein. Man muss dann allerdings die Zugriffsrechte auf die einzelnen Tabellen so definieren, dass nicht einfach ein Join ausgeführt werden kann. Meistens würde man eher ein View über der entsprechenden Tabelle/Anfrage erzeugen, das die betroffenen Spalten ausblendet.

1/2P: Natürlichen Schlüsseln wird oft eine höhere Intuitivität und Lesbarkeit zugeschrieben. Aber ist Reise "N2412" wirklich weniger intuitiv, als im Aufgabentext jedes Mal "die Reise mit der Neptun ab dem 16.5.2024" zu schreiben?

Weitere Anmerkungen:
Bei der oben beschriebenen Ergänzung von "Buchung" um "Schiff" könnte damit wiederum beim Namen des Schiffes ein Fehler passieren (nicht das Schiff, das zu der gebuchten Reise.code passt), den man nicht mit einer Integritätsbedingung prüfen kann, da (Reise(code),Schiff) nirgends Primary Key ist.
Weitergehend wäre eine Integritätsbedingung Buchung(Kabine, Schiff, Klasse) $\rightarrow$ Kabine(Nr, Schiff, Klasse) wünschenswert, das geht aber -egal wie man es macht- nicht, da es dort nicht Primary Key ist.

Man sieht, dass man sich in einer stark zwischen den Entitätstypen vernetzten Datenbank mit künstlichen IDs Integritätsprobleme einhandeln kann.

Weitere Übungsaufgaben zu dieser Datenbank

Aufgabe 18 (Anfragen [0 Punkte])
Erweiterung der vorhergehenden Aufgabe: Geben Sie eine SQL-Anfrage an, die für jede Person, die einmal eine Buchung durchgeführt hat, ausgibt, wieviel Umsatz das Unternehmen mit von dieser Person durchgeführten Buchungen (*einschließlich der Ausflüge*) insgesamt gemacht hat (einschließlich Buchungen, deren Reisedatum in der Zukunft liegt).

Lösung

```
SELECT Person, SUM (Preis)
FROM  ((SELECT Person, Preis
        FROM  Buchung b, Angebote a
        WHERE b.reise = a.reise AND b.klasse = a.klasse
        )
       UNION
        (SELECT Person, Preis * Anzahl as Preis
         FROM Ausflugsbuchung a, Buchung b, ausflug x
         WHERE a.buchung = b.buchung
           AND b.reise = x.reise and a.datum = x.datum
           AND a.ausflug = x.bezeichung)
       )
GROUP BY Person
```