Universität Heidelberg
Institut für Informatik
Lehrstuhl für Datenbanksysteme
Prof. Dr. Michael Gertz

16.7.2019
Datenbanken
Sommersemester 2019
Lösungen zur Probeklausur

Vorlesung Datenbanken im SomSem 2019
— Lösungen zur Probeklausur —

Aufgabe 1: ER Modellierung

a. Geben Sie ein ER-Schema an, dass die obigen Informationen modelliert. Geben Sie Primärschlüsselattribute zu Entitytypen sowie Kardinalitäten zu den Beziehungstypen an.

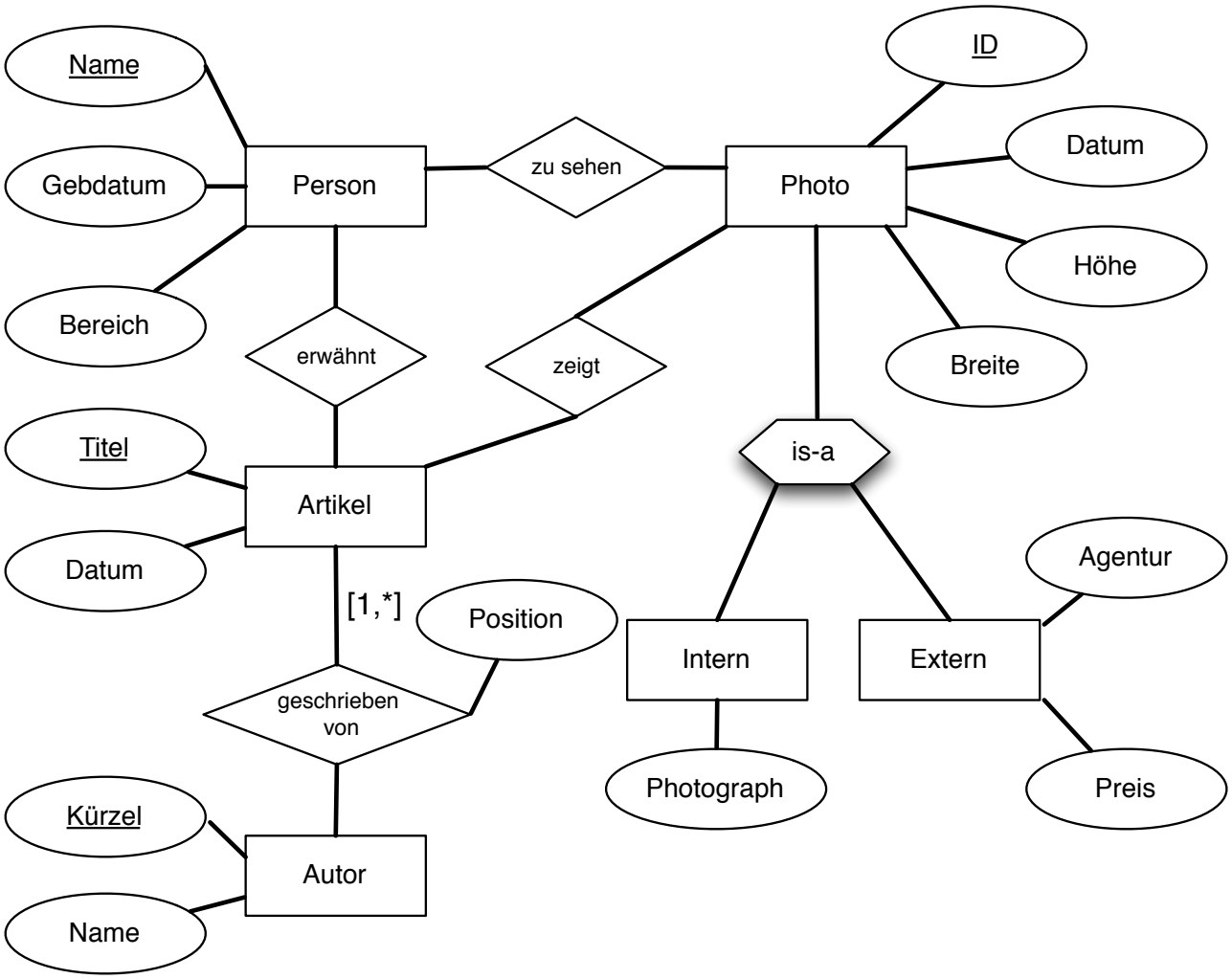


Figure 1: ER-Schema zu Aufgabe 1

Kommentare: Die verschiedenen Arten von Photos sind hier mit Hilfe einer Spezialisierung modelliert worden. Eine alternative (aber wesentlich schlechtere) Modellierung wäre, wenn man die Attribute der Unter-Typen einfach einem Entitytyp `Foto` hinzugefügt hätte und somit

keine Untertypen modelliert hätte. Alle nicht spezifierten Kardinalitäten sind vom Typ (0,*), z.B. kann ein Photo viele Personen zeigen, muss aber nicht unbedingt eine Person zeigen.

b. Geben Sie zwei Integritätsbeziehungen an (umgangssprachlich), die sich nicht in einem ER-Schema modellieren lassen.

- Das Datum eines Photos muss vor dem Datum des Artikels liegen, in dem dieses Photo erscheint.
- Beschreibt ein Artikel eine Person und enthält dieser Artikel auch ein Photo, so soll das Photo auch diese Person zeigen.
- Ein Photo kann nicht sowohl extern als auch Intern sein.
- ...

c. Übersetzen Sie Ihr ER-Schema in eine Menge von Relationenschemata. Geben Sie alle Primär- und Fremdschlüssel zu den Relationen an.

```
 Person(Name, GebDatum, Bereich)
Artikel(Titel, Datum)
Autor(Kuerzel, Name)
Foto(ID, Datum, Hoehe, Breite)
geschriebenVon(Titel→Artikel, Kuerzel→Autor)
beschreibt(Titel→Artikel, Name→Person)
zeigt(Name→Person, ID→ Foto)
enthaelt(Titel→Artikel, ID→ Foto)
extern(ID→ Foto, Fotograph)
intern(ID→ Foto, Agentur, Preise)
```

Zur Übersetzung der Aggregation gibt es noch eine (weniger gute) Alternative (eine Relation `FOTO` mit allen Attributen).

Aufgabe 2: Integritätsbedingungen

a. Welche Attribute in den obigen Relationen sind Fremdschlüsselattribute und welche Relationen referenzieren diese Attribute? Sie können hierzu folgende Notation verwenden, um zu beschreiben, dass ein Fremdschlüssel der Relation 1 auf den Primärschlüssel einer Relation 2 verweist: $\langle$Relation1$\rangle.\langle$Attribut(e)$\rangle \longrightarrow \langle$Relation2$\rangle$,

Zweigstelle.plz $\rightarrow$ Stadt
Route.start-plz $\rightarrow$ Stadt
Route.end-plz $\rightarrow$ Stadt
Spedition.plz $\rightarrow$ Stadt
Route.sname $\rightarrow$ Spedition
Zweigstelle.sname $\rightarrow$ Spedition

b. Gegeben sei folgende Integritätsbedingung: *Die Gesamtzahl der LKWs einer Spedition muss mindestens gleich der Summe der LKWs an den Zweigstellen der Spedition sein.* Geben Sie alle Operationen an, die diese Bedingung verletzen können.

update lkws on SPEDITION (new.lkws $<$ old.lkws)
insert into ZWEIGSTELLE
update lkws on ZWEIGSTELLE (new.lkws $>$ old.lkws)

Aufgabe 3: SQL-Anfragen

Formulieren Sie folgende Anfragen an die Beispieldatenbank in SQL:

a. Geben Sie die Postleitzahl all der Zweigstellen an, die mindestens 50% aller LKWs einer Spedition haben.

```
select z.plz
from zweigstelle z, spedition s
where z.sname = s.sname and z.lkws * 2 >= s.lkws;
```

b. Welche Speditionen haben ihre Zweigstellen nur in einer Stadt (aber eventuell unter verschiedenen Postleitzahlen)? Formulieren Sie diese Anfrage ohne Aggregationen oder Gruppierung. Auszugeben sind Name der Spedition und Name der Stadt.

```
select s.*, t.name
from SPEDITION s, STADT t
where s.plz = t.plz
  and not exists (select *
                  from ZWEIGSTELLE z, STADT t2
                  where z.sname = s.sname
                    and z.plz = t2.plz
                    and t2.name <> t.name);
```

Beachte: Da in der Relation STADT das Attribut Plz der Schlüssel ist, können bei dieser Anfrage inkorrekte Ergebnisse auftauchen (zwei Städte können den gleichen Namen haben).

c. Wie viele Zweigstellen hat die Spedition mit den meisten LKWs? Auszugeben sind der Name der Spedition und die Anzahl der Zweigstellen.

```
SELECT  s.name, s.zweigst
FROM    spedition s
WHERE   s.lkws >= all ( SELECT s2.lkws FROM spedition s2)
```

d. Welche Speditionen haben ihren Hauptsitz in Karlsruhe und bedienen mehr als 50 verschiedene Routen?

```
select s.sname
from SPEDITION s, STADT t, ROUTE r
where s.sname = r.sname
  and s.plz = t.plz and t.name = 'Karlsruhe'
group by s.sname
having count(*) > 50;

oder einfacher

select s.sname
from SPEDITION s, STADT t
where s.plz = t.plz and t.name = 'Karlsruhe'
  and 50 < (select count(*) from ROUTE where sname = s.sname);
```

e. Gibt es Routen, die die gleichen Start- und Endpostleitzahlen aber verschiedene Entfernungen haben? Auszugeben sind hierzu komplette Tupel der Relation ROUTE.

```
select *
from ROUTE r1
where exists (select *
              from ROUTE r2
              where r1.start_plz = r2.start_plz
                and r1.end_plz = r2.end_plz
                and r1.km <> r2.km);
```

f. Geben Sie für jede Route mit mehr als 1000 Km die Anzahl der Speditionen an, die diese Route bedienen. Gesucht sind nur Routen, die von mehr als 5 Speditionen bedient werden. Auszugeben sind neben der Anzahl der Speditionen die Start- und Endpostleitzahl.

```
select start_plz, end_plz, count(*)
from ROUTE
where km > 1000
group by start_plz, end_plz
having count(*) > 5;
```

g. Die Spedition “HotWheels” hat zwei neue Zweigstellen in den Städten mit den Postleitzahlen 69120 und 76646 aufgemacht. Insgesamt hat die Spedition 20 neue LKWs, die gleich auf die beiden neuen Zweigstellen verteilt werden. Geben Sie alle Update-Operationen auf entsprechende Relationen an, um diese Änderungen durchzuführen.

```
update SPEDITION
set lkws = lkws + 20
where sname  ='HotWheels';

insert into ZWEIGSTELLE values ('HotWheels', 69120, 10);
insert into ZWEIGSTELLE values ('HotWheels', 76646, 10);
```

Aufgabe 4: Anfragebearbeitung und -optimierung

a. Was wird durch die folgende SQL-Anfrage bestimmt?

```
select S.sname, T.name, R.km
from SPEDITION S, ROUTE R, STADT T
where S.sname = r.sname and
      R.end-plz = 69120 and R.start-plz = T.plz
```

Für jede Route, die in der Stadt mit der Postleitzahl 69120 endet, gebe den Namen der Speditionen aus, die eine Route zu dieser Stadt bedienen, inklusive dem Namen der "Start-Stadt" und der Länge der Route.

b. Übersetzen Sie diese Anfrage **direkt** in einen Ausdruck der Relationenalgebra. Verwenden Sie nur die Grundoperationen der Algebra (beachten Sie hierbei, dass z.B. der Join hier als nur eine abgeleitete Operation betrachtet wird).

$$\pi_{\texttt{SPEDITION.sname,name,km}}\Big(\sigma_{\texttt{end\_plz=69120}\wedge\texttt{SPEDITION.sname=ROUTE.sname}\wedge\texttt{start\_plz=plz}}(\texttt{SPEDITION}\times\texttt{ROUTE}\times\texttt{STADT})\Big)$$

Man hätte hier auch Umbenennungen von SPEDITION, ROUTE und STADT verwenden können.

c. Führen Sie eine algebraische Optimierung für den unter b) bestimmten Ausdruck unter Verwendung der Transformationsregeln durch. Es reicht aus, wenn Sie hierzu nur das Endergebnis (Anfragebaum) angeben.

Beachte: Im folgenden ist nur der Ausdruck angegeben, welcher ja einfach in einen Operatorbaum umgewandelt werden kann (und hier auch als Lösung angegeben werden sollte).

$$\pi_{\texttt{sname,name,km}}\Big(\sigma_{\texttt{end\_plz=69120}}(\texttt{ROUTE})\underset{\texttt{plz=start\_plz}}{\bowtie}\texttt{STADT}\Big)$$

Beachte_2: Die Relation SPEDITION taucht in diesem Ausdruck gar nicht mehr auf, da das Attribute sname schon in der Relation ROUTE vorhanden ist und somit ein Join nicht notwendig ist (Ok, das ist nicht trivial, und eine Lösung mit der Relation SPEDITION wäre auch korrekt gewesen).

d. Angenommen für die Primärschlüssel in den obigen Relationen existiert jeweils ein Clustered Index. Es sei weiterhin angenommen, dass ein Unclustered Index für das Attribut end-plz existiert.

Geben Sie einen effizienten Auswertungsplan (annotierter Anfragebaum) für die von Ihnen unter c) beschriebene Anfrage an. Welche Algorithmen verwenden Sie für die Joins? Begründen Sie Ihre Wahl. Begründen Sie auch, warum der von Ihnen erstellte Auswertungsplan am effizientesten ist.

Beachte: Anzugeben ist hier natürlich ein Auswertungsplan in Form eines Baumes. Im Folgenden wird dieser Baum nur beschrieben:

Erst wird der Index (Index Scan) auf ROUTE.end_plz verwendet, um die TIDs mit der Eigenschaft end_plz = 69120 zu bekommen. Dies ist wesentlich effizienter als ein Scan der Relation ROUTE, um entsprechende ROUTE-Tupel zu bekommen. Unter Verwendung dieser TIDs hole dann die kompletten Tupel aus der Relation ROUTE (Realisierungs-Operator). Nun führe mit dem Zwischenergebniss ein Condition-Join (start_plz = plz, unter Verwedung eines Nested-Loops Join Algos) mit der Relation STADT durch. Zum Schluss projiziere auf die Attribute sname, name und km.

Aufgabe 5: Relationale Algebra

Formulieren Sie folgende Anfragen als Ausdrücke der relationalen Algebra. Wenn möglich, wenden Sie die Transformationsregeln an, um eine effiziente Anfrage zu erhalten.

a. Welche Spedition (sname) aus Mannheim hat die meisten LKWs?

Sei $\texttt{MS} = \sigma_{\texttt{name}='\texttt{Mannheim}'}(\texttt{STADT} \underset{\texttt{STADT.plz}=\texttt{SPEDITION.plz}}{\bowtie} \texttt{SPEDITION})$

$\pi_{\texttt{sname}}(\texttt{MS}) - \pi_{\texttt{sname}}(\sigma_{\texttt{lkws}<\texttt{lkws2}}(\texttt{MS} \times \beta_{\texttt{lkws2}\leftarrow\texttt{lkws}}(\pi_{\texttt{lkws}}(\texttt{MS}))))$

b. Geben Sie alle Routen an, die von solchen Speditionen bedient werden, die eine Zweigstelle in Sindelfingen haben.

$\pi_{\texttt{ROUTE}.*}(\sigma_{\texttt{name}=\texttt{Sindelfingen}}(\texttt{STADT}) \bowtie \texttt{ZWEIGSTELLE}) \bowtie (\texttt{ROUTE})$

Formulieren Sie die folgende Anfrage umgangssprachlich:

c. $\pi_{\texttt{sname}}(\sigma_{\texttt{lkws}>100 \wedge \texttt{blkws}>90}(\texttt{SPEDITION} \bowtie (\beta_{\texttt{blkws}\leftarrow\texttt{lkws}}(\pi_{\texttt{sname,lkws}}(\texttt{ZWEIGSTELLE})))))$

Namen der Speditionen, die insgesamt mehr als 100 LKWs haben und mind. eine Zweigstelle haben, die mehr als 90 LKSw hat.

Aufgabe 6: Transaktionsverwaltung

a. Gegeben seien die beiden folgenden Schedules $\texttt{S}_1$ und $\texttt{S}_2$:

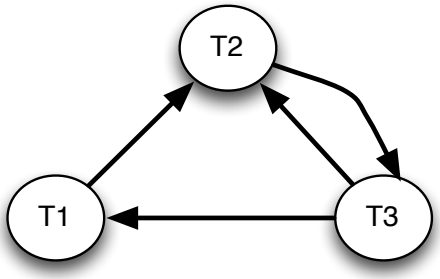
$\texttt{S}_1$:

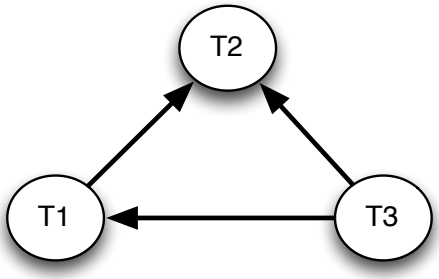
T_1	T_2	T_3
read(X)		
	read(Z)	
		read(X)
read(Z)		
	read(Y)	
		read(Y)
write(X)		
	write(Z)	
		write(Y)
	write(Y)	

$\texttt{S}_2$:

T_1	T_2	T_3
read(X)		
	read(Z)	
read(Z)		
		read(X)
		read(Y)
write(X)		
		write(Y)
	read(Y)	
	write(Z)	
	write(Y)	

Welcher der beiden Schedules ist serialisierbar? Zeichnen Sie jeweils den zugehörigen Konfliktgraphen und geben Sie gegebenenfalls jeweils einen seriellen Schedule an (hier reicht die Reihenfolge der Transaktionen).





S_1 ist nicht serialisierbar (Zyklus zw. T2 und T3); S_2 ist serialisierbar: T3, T1, T2

b. Angenommen sei die folgende Transaktion T. Zu Beginn der Transaktion hat das Objekt A den Wert 100.

Zeit	Operation
1	read(A,a)
3	a:=a + 100
5	write(A,a)
7	read(B,b)
9	write(B,b)
11	commit

(i) Angenommen sei eine weitere Transaktion T', die das Objekt A zum Zeitpunkt 4 lesen will. Welchen Wert für A sollte T' zu lesen bekommen? Woher nimmt das DBMS den entsprechenden Wert für A?

Falls T kein xlock auf A gesetzt hat, sieht T' den Wert 100 für A, anderenfalls hat T' kein Zugriff auf A. Wenn T' für A den Wert 100 sieht, so wird dieser Wert aus dem Redo-Log Buffer genommen (siehe auch Abschnitt 10.3 im Textbuch).

(ii) Angenommen, das System stürzt zum Zeitpunkt 9 ab. Welchen Wert hat A nach einem Neustart des Systems? Begründen Sie Ihre Antwort.

A hat den Wert 100, da die Transaktion T kein commit durchgeführt hat.