UNIVERSITÄT DES SAARLANDES
Fachbereich 14 – Informatik
Lehrstuhl Prof. Weikum

Datenbanksysteme WS 1999/2000

Übung 13 — Beispiellösung

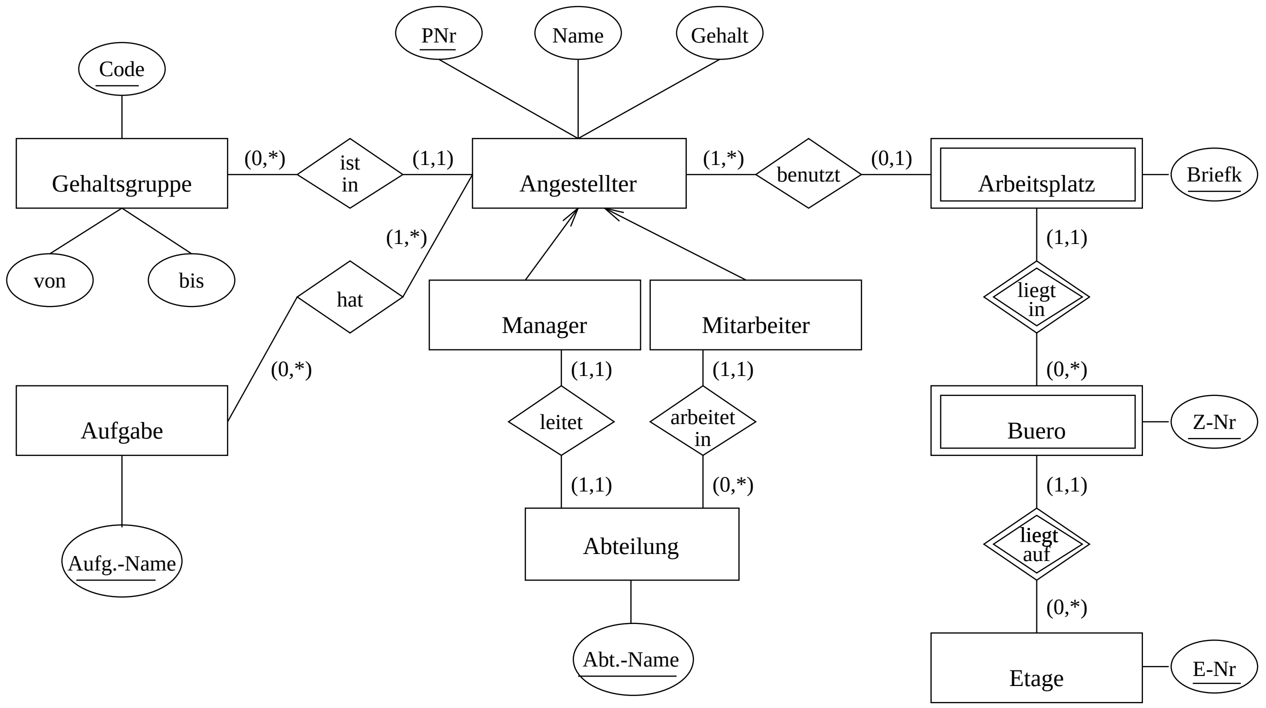
Aufgabe 1: Transformation ERM nach Relationenmodell

```
KUNDE          (Knr, Adresse, Rabattt, Name)
HAENDLER       (Name, Adresse, Provision)
PRODUKT        (PNr, Bez, Preis, Gewicht)
LAGER          (Ort)
BESTELLUNGEN   (BestNr, Datum, KUNDE.Knr, HAENDLER.Name, HAENDLER.Adresse)
LIEFERUNG      (LAGER.Ort, BESTELLUNGEN.BestNr, PRODUKT.PNr, Datum)
BESTELLPOSTEN  (BESTELLUNGEN.BestNr, PRODUKT.PNr, Menge)
LAGERUNG       (LAGER.Ort, PRODUKT.PNr, Vorrat)
```

Bemerkung An der im ERM modellierten Beziehung *Vertrieb* ist die Entität *Bestellungen* mit Kardinalität (1,1) beteiligt. Diese Beziehung kann daher durch die Hinzunahme der Primärschlüssel von Kunden und Händer in die Relation `BESTELLUNGEN` ausgedrückt werden.

Aufgabe 2: Modellierung mit ERM

a) ERM-Diagramm



Um der Semantik der Aufgabenstellung gerecht zu werden, muß für die beiden ISA-Beziehungen (Manager resp. Mitarbeiter) gefordert werden, daß sie disjunkt und vollständig–überdeckend sind.

b) Nach Regel 1 (Skript S.279) muß für jedes Entity eine eigene Relation erstellt werden:

```
Abteilung      (Abt.-Name)
Arbeitsplatz   (Briefk, Z-Nr, E-Nr, PNr)
Aufgabe        (Aufg.-Name)
Büro           (Z-Nr, E-Nr)
Etage          (E-Nr)
Gehaltsgruppe  (Code, von, bis)
Mitarbeiter    (Pnr, Name, Gehalt, Code, Abt.-Name)
Manager        (Pnr, Name, Gehalt, Code, Abt.-Name)
```

Nach Regel 2 können Relationships, die in einer N:1-, 1:N- oder einer 1:1–Beziehung stehen, in eine Entity–Relation integriert werden. Dies wurde für die folgenden Relationships durchgeführt:

„`ist in`“ wurde in die Relationen `Manager` bzw. `Mitarbeiter`,
„`benutzt`“ wurde in `Arbeitsplatz`,
„`leitet`“ wurde in `Manager` und
„`arbeitet in`“ wurde in `Mitarbeiter` integriert.

Die Relationships „`liegt in`“ und „`liegt auf`“ können aufgrund dessen, daß sie *weak-relationships* sind, in die Relationen `Etage` bzw. `Büro` integriert werden.
Daher brauchen wir nur noch die Relationship `hat` separat als Relation zu beschreiben:
`hat (Aufg.-Name, PNr)`

Aufgabe 3: Ausbeute (Recall) und Präzision (Precision)

a) Es seien $C_1 > 0$ die Kosten, welche entstehen, wenn ein Benutzer ein vom Suchsystem gefundenes Dokument inspiziert, das irrelevant ist. Außerdem seien $C_2 > 0$ die Kosten, die dadurch entstehen, wenn das Suchsystem ein relevantes Dokument nicht unter den ersten r Dokumenten rangiert, die der Benutzer inspiziert. Ermitteln Sie, ob in folgenden Situationen gilt $C_1 \ll C_2$, $C_1 \approx C_2$ oder $C_1 \gg C_2$.

- Ein Patentanwalt überprüft, ob eine Erfindung bereits patentiert ist.
- Ein Journalist möchte herausfinden, wann die Swissair das erste Mal über den Atlantik flog.

Lösung:
Im ersten Fall (Patentanwalt) gilt $C_1 \ll C_2$. Die Kosten C_1 (Kosten für das Inspizieren eines irrelevanten Dokuments) entsprechen dabei den Kosten für etwa 5 sec Arbeitszeit. Falls jedoch ein relevantes Dokument übersehen wird, können sich die Kosten C_2 (Kosten, die durch das Nicht-Inspizieren eines relevanten Dokuments entstehen) hingegen auf mehrere Tausend Euro (Gerichtskosten, etc.) belaufen.

Im Falle der journalistischen Recherche gilt gilt $C_1 \gg C_2$. C_1 entspricht dabei wieder den Kosten für etwa 5 sec Arbeitszeit. Allerdings ist es in diesem Beispiel egal, ob noch ein zusätzliches relevantes Dokument existiert, das nicht inspiziert wird.
Daher gilt $C_2 = 0$.

Im ersten Beispiel (Patentanwalt) ist also eine gute Ausbeute (möglichst **alle** relevanten Dokumente) gefragt, während im zweiten Beispiel (journalistische Recherche) eine gute Präzision (möglichst **keine** irrelevanten Dokumente) vorliegen sollte.

b) Eine Dokumentenkollektion enthalte $n = 1$ Mio. Dokumente, wovon der Benutzer $r = 10$ nach einer Anfrage inspiziert. Dabei wird eine Ausbeute von 20% und eine Präspizierzision von 50% erzielt. Der „Fallout“einer Anfrage ist definiert als das Verhältnis

$$\frac{\text{Anzahl inspizierter irrelevanter Dokumente}}{\text{Anzahl aller irrelevanten Dokumente}}.$$

- Wie hoch ist der Fallout?
- Wie verhält sich der Fallout für $r \to n$?

Lösung:
Bie einer Präzision $\pi = 0.5$ gilt, daß von 10 inspizierten Dokumenten 5 relevant sind. Die Ausbeute $\rho = 0.2$ (Anteil inspizierter relevanter Dokumente bzgl. aller relevanten) gibt an, daß es insgesamt 25 relevante Dokumente gibt.

Im folgenden bezeichne n die Gesamtzahl der Dokumente, r die Anzahl der inspizierten Dokumente, $\left|D_r^{rel}\right|$ die Anzahl der relevanten Dokumente, die inspiziert wurden und $\left|D^{rel}\right|$ die Gesamtzahl der relevanten Dokumente.
Dann gilt für den Fallout: $Fallout = \frac{r-\left|D_r^{rel}\right|}{n-\left|D^{rel}\right|}$, in unserem Beispiel also: $Fallout = \frac{10-5}{10^6-25}$
Damit ergibt sich für $r \to n$ folgendes Verhalten des Fallouts:

$$\lim_{r\to n} Fallout = \lim_{r\to n} \frac{r-\left|D_r^{rel}\right|}{n-\left|D^{rel}\right|} = \frac{10^6-25}{10^6-25} = 1$$

Aufgabe 4: ff*-idf-Gewichtung

Gegeben seien die Features (bzw. Terme) $F = \{a, b, c, d, e\}$ und die Dokumente

$$\begin{array}{ll} d_1 : a\ b\ b\ e\ a\ a & d_2 : c\ b\ b\ c\ a\ c\ d \\ d_3 : b\ b\ b\ b\ e\ b & d_4 : c\ b\ e \\ d_5 : b\ a\ c\ b & d_6 : e\ e\ e\ c\ e \end{array}$$

.

Berechnen Sie die ff*idf-Gewichte der Indexfeatures sowie die Ergebnisse der Anfragen unter Verwendung des Cosinusmaßes und der ff*idf-Gewichtung.

$$\begin{array}{ll} q_1 : d & q_2 : c \\ q_3 : c\ d & q_4 : a\ b\ e \\ q_5 : a\ b\ c\ d\ e & \end{array}$$

.

Lösung:
Zur Berechnung der ff*idf-Gewichtung werden zunächst die Featurehäufigkeiten und die relativen Dokumenthäufigkeiten bestimmt.

Die Featurehäufigkeit (feature frequency) $ff(f_j, d_i)$ ist die Häufigkeit des Feature f_j im Dokument d_i, hier für $1 \leq i \leq 6$ und $1 \leq j \leq 5$.

$ff(f_j, d_i)$	$j = 1$ $(f_1 = a)$	$j = 2$ $(f_2 = b)$	$j = 3$ $(f_3 = c)$	$j = 4$ $(f_4 = d)$	$j = 5$ $(f_5 = e)$
$i = 1$	3	2	0	0	1
$i = 2$	1	2	3	1	0
$i = 3$	0	5	0	0	1
$i = 4$	0	1	1	0	1
$i = 5$	1	2	1	0	0
$i = 6$	0	0	1	0	4

Die inverse Dokumenthäufigkeit (inverse document frequency) ist definiert als:
$idf(f_j) = log \frac{\text{Anzahl aller Dokumente}}{\text{Anzahl der Dokumente mit } f_i}$
Damit ergeben sich folgende Werte:

j	f_j	$idf(f_j)$		
1	a	$log\frac{6}{3}$	$= log\, 2$	$\approx 0,3010$
2	b	$log\frac{6}{5}$	$= log\, 1,2$	$\approx 0,0792$
3	c	$log\frac{6}{4}$	$= log\, 1,5$	$\approx 0,1761$
4	d	$log\frac{6}{1}$	$= log\, 6$	$\approx 0,7782$
5	e	$log\frac{6}{4}$	$= log\, 1,5$	$\approx 0,1761$

Aus den beiden Tabellen ergibt sich dann die ff*idf-Gewichtung mittels $d_{ij} = ff(f_j, d_i) * idf(f_j)$.

d_{ij}	$j=1$ $(f_1=a)$	$j=2$ $(f_2=b)$	$j=3$ $(f_3=c)$	$j=4$ $(f_4=d)$	$j=5$ $(f_5=e)$
$i=1$	$3*\log 2$ $\approx 0,9031$	$2*\log 1,2$ $\approx 0,1584$	$0*\log 1,5$ $=0$	$0*\log 6$ $=0$	$1*\log 1,5$ $\approx 0,1761$
$i=2$	$1*\log 2$ $\approx 0,3010$	$2*\log 1,2$ $\approx 0,1584$	$3*\log 1,5$ $\approx 0,5283$	$1*\log 6$ $\approx 0,7782$	$0*\log 1,5$ $=0$
$i=3$	$0*\log 2$ $=0$	$5*\log 1,2$ $\approx 0,3959$	$0*\log 1,5$ $=0$	$0*\log 6$ $=0$	$1*\log 1,5$ $\approx 0,1761$
$i=4$	$0*\log 2$ $=0$	$1*\log 1,2$ $\approx 0,0792$	$1*\log 1,5$ $\approx 0,1761$	$0*\log 6$ $=0$	$1*\log 1,5$ $\approx 0,1761$
$i=5$	$1*\log 2$ $\approx 0,3010$	$2*\log 1,2$ $\approx 0,1584$	$1*\log 1,5$ $\approx 0,1761$	$0*\log 6$ $=0$	$0*\log 1,5$ $=0$
$i=6$	$0*\log 2$ $=0$	$0*\log 1,2$ $=0$	$1*\log 1,5$ $\approx 0,1761$	$0*\log 6$ $=0$	$4*\log 1,5$ $\approx 0,7044$

Um schließlich die Ähnlichkeit der Anfragen zu den jeweiligen Dokumenten unter Verwendung des Cosinusmaßes zu bestimmen, benötigt man zunächst noch die Gewichtung der Features in den Queries. Dazu wird die einfachste Variante benutzt: Die vorhandenen Features werden mit 1 gewichtet und nicht vorhandene Features mit 0 (-1 wäre auch denkbar, je nach Semantik der Notation aus der Aufgabestellung). Dadurch erhält man für jede Query einen Vektor aus $[0,1]^5$:

q_k	q_{k1} (a)	q_{k2} (b)	q_{k3} (c)	q_{k4} (d)	q_{k5} (e)
$q_1 : d$	0	0	0	1	0
$q_2 : c$	0	0	1	0	0
$q_3 : cd$	0	0	1	1	0
$q_4 : abe$	1	1	0	0	1
$q_5 : abcde$	1	1	1	1	1

Die Änlichkeit zwischen einem Dokument d_i und einer Query q_k (vgl. Skript S.294) ist hier:

$$similarity(d_i, q_k) = \frac{d_{i1}q_{k1} + d_{i2}q_{k2} + d_{i3}q_{k3} + d_{i4}q_{k4} + d_{i5}q_{k5}}{(d_{i1}^2 + d_{i2}^2 + d_{i3}^2 + d_{i4}^2 + d_{i5}^2) * (q_{k1}^2 + q_{k2}^2 + q_{k3}^2 + q_{k4}^2 + q_{k5}^2)}$$

Folgende Tabelle enthält für jede Query q_k die (gerundeten) Ähnlichkeit zum Dokument d_i. Die Rangfolge der Dokumente im Ergebnis einer Query erhält man durch Sortierung der Dokumente nach absteigender Ähnlichkeit:

$similarity(d_i, q_k)$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$i=1$	0	0	0	0,47	0,28
$i=2$	0,78	0,53	0,65	0,15	0,35
$i=3$	0	0	0	1,02	0,61
$i=4$	0	2,58	1,29	1,25	1,26
$i=5$	0	1,20	0,60	1,04	0,87
$i=6$	0	0,33	0,17	0,45	0,33