Technische Universität München
Fakultät für Informatik
Prof. R. Bayer, Ph.D.
Dipl.-Inform. Michael Bauer
Dr. Gabi Höfling

WS 2003/04,
Datenbanksysteme I
Lösungsblatt 5

24.11. 2003

Tupel- und Domänenkalkül, SQL - Beispiel: Kontoführung

Ausgangspunkt für Aufgabe 1 und Aufgabe 2 ist das aus den vorherigen Blättern bekannte Beispiel der Kontoführung mit folgendem relationalen Schema:

Kategorie:	(Bezeichnung:string, Klassifikationskriterium:string)
Kunde:	(Vorname:string, Name:string, Gebdat:date, Bezeichnung:string)
Konto:	(Kontonr: integer, Betrag:numeric, Gebühr:numeric)
Festgeldkonto:	(Kontonr: integer, Fälligkeit:date, Zinssatz: numeric)
Girokonto:	(Kontonr: integer, Dispolimit:numeric, Telebanking:boolean)
Kontobewegung:	(Datum:date, Vorname:string, Name:string, Gebdat:date, Kontonr:integer, Betrag: numeric, Bewegungsart: string)
Kontoauszug:	(Erstelldat:date, Kontonr:integer, Startdat:date, Gesamt:numeric)
besitzt:	(Vorname:string, Name:string, Gebdat:date, Kontonr:integer)
listet:	(Erstelldat:date, Kontonr:integer, Datum:date, Vorname:string, Name:string, Gebdat:date)

Aufgabe 1: Tupelkalkül

Formulieren Sie folgende Anfragen im Tupelkalkül:

1. Alle Kunden.

   $\{k \mid k \in \text{Kunde}\}$

   Die Tupelvariable s wird durch das Prädikat auf der rechten Seite bestimmt, d.h. die Ergebnismenge sind all die Tupel, bei denen die rechte Seite zu true ausgewertet wird. Da in unserem Fall die freie Variablen s nicht mehr weiter eingeschränkt wird, sind alle Tupel der Relation Kunde im Ergebnis enthalten

2. Alle Kategoriebezeichnungen

   $\{\mathbf{[kat.Bezeichnung]} \mid \text{kat} \in \text{Kategorie}\}$

   In unserer Abfrage sind wir nur an einer Spalte interessiert, aber allen Tupeln. Die rechte Seite bleibt gleich, links schränken wir die Ergebnistupel auf die Spalte Benennung der Ursprungstupel ein bzw. erstellen neue Tupel mit nur dieser einen Spalte

3. Alle Kontobewegungen des Kunden Heinz Schiller (geb. am 3.5.1942), bei denen mehr als 500 Euro bewegt wurden

   $\{kb \mid kb \in \text{Kontobewegung} \wedge$
   $\mathbf{kb.Vorname = Heinz \wedge kb.Name = Schiller \wedge kb.Gebdat = 19420503 \wedge}$
   $\mathbf{kb.Betrag > 500}\}$

   Hier kommt eine Einschränkung der Tupelmenge aus der Ursprungsrelation dazu.

   Es besteht auch die Möglichkeit, mehrere Bedingungen durch logische Operatoren zu verknüpfen, ensprechend den „normalen“ prädikatenlogischen Formeln.

4. Alle Kunden des Kontos 174266

   $\{k \mid k \in \text{Kunde} \wedge \exists\, \mathbf{b} \in \mathbf{besitzt}$
   $(k.\text{Vorname} = b.\text{Vorname} \wedge k.\text{Name} = b.\text{Name} \wedge k.\text{Gebdat} = b.\text{Gebdat} \wedge$
   $b.\text{Kontonr} = 174266)\}$

   Beziehungen zu anderen Relation können über Existenz- oder Allquantoren ausgedrückt werden. Diese Beziehungen können dann wieder durch aussagenlogische Ausdrücke beschränkt werden.

5. Alle Kunden mit mehr als 5000 Euro Guthaben

   $\{k \mid k \in \text{Kunde} \wedge \exists\, \mathbf{b} \in \mathbf{besitzt}$
   $(k.\text{Vorname} = b.\text{Vorname} \wedge k.\text{Name} = b.\text{Name} \wedge k.\text{Gebdat} = b.\text{Gebdat} \wedge$
   $\exists\, \mathbf{kn} \in \mathbf{Konto}\ (kn.\text{Kontonr} = b.\text{Kontonr} \wedge kn.\text{Betrag} > 5000))\}$

   Die Bedingungen zwischen Relationen können auch mehrfach geschachtelt werden.

6. Alle Kunden, die nur Girokonten haben (d.h. alle Kunden, die kein Festgeldkonto haben)
   Die beiden Aussagen sind nicht äquivalent, da im zweiten Fall auch Kunden ohne Konto gemeint sein können.
   Alle Kunden, die kein Festgeldkonto oder gar kein Konto haben:

   $\{k \mid k \in \text{Kunde} \wedge \neg\, \exists\, \mathbf{b} \in \mathbf{besitzt}$
   $(k.\text{Vorname} = b.\text{Vorname} \wedge k.\text{Name} = b.\text{Name} \wedge k.\text{Gebdat} = b.\text{Gebdat} \wedge$
   $\exists\, f \in \text{Festgeldkonto}\ (f.\text{Kontonr} = b.\text{Kontonr}))\}$

   Existenztests können auch negativer Art sein, also überprüfen, ob es kein Tupel einer bestimmten Beschreibung gibt.

   Der Ausdruck liefert ein endliches Ergebnis, da er sicher ist, d.h. das Ergebnis ist Teilmenge der Domäne der Formel. Durch die Angabe von $k \in \textit{Kunde}$ enthält die Domäne dieser Formel alle Attributwerte der Relation *Kunde* und damit auch die Ergebnismenge.

   Alle Kunden, die kein Festgeldkonto aber ein Girokonto haben:

   $\{k \mid k \in \text{Kunde} \wedge \exists\, \mathbf{b} \in \mathbf{besitzt}$
   $(k.\text{Vorname} = b.\text{Vorname} \wedge k.\text{Name} = b.\text{Name} \wedge k.\text{Gebdat} = b.\text{Gebdat} \wedge$
   $\neg\, \exists\, \mathbf{f} \in \mathbf{Festgeldkonto}\ (f.\text{Kontonr} = b.\text{Kontonr}))\}$

Aufgabe 2: Domänenkalkül

Formulieren Sie die Anfragen aus Aufgabe 1 entsprechend im Domänenkalkül.

1. Alle Kunden.

   $\{[v,n,g,b] \mid [v,n,g,b] \in \text{Kunde}\}$

   Beim Domänenkalkül werden einzelne Domänen für die jeweiligen Spalten angegeben, die dann weiter bestimmt werden, z.B. als Teil einer Relation.

2. Alle Kategoriebezeichnungen

   $\{\mathbf{[b]} \mid \exists\, \mathbf{x}\ ([b,x] \in \text{Kategorie})\}$

   Falls nur auf einen Teil einer Relation zugegriffen wird, muss der Rest durch gebundene Variablen ausgedrückt werden. Die Bedeutung der einzelnen Variablen ist durch ihre Position in der Relation bestimmt, nicht durch explizite Benennung.

3. Alle Kontobewegungen des Kunden Heinz Schiller (geb. am 3.5.1942), bei denen mehr als 500 Euro bewegt wurden

   {[d,v,n,g,k,b,a] | [d,v,n,g,k,b,a] ∈ Kontobewegung ∧
   v = Heinz ∧ n = Schiller ∧ g = 19420503 ∧ b > 500}

   Ähnlich wie beim Tupelkalkül können die Werte der Variablen eingeschränkt und diese Einschränkungen logisch verknüpft werden.

4. Alle Kunden des Kontos 174266

   {[v,n, g,b] | [v,n, g,b] ∈ Kunde ∧ **∃ k ([v,n,g,k] ∈ besitzt** ∧ k=174266) }

   Joins können im Domänenkalkül implizit durch die Verwendung derselben Domänenvariablen in mehreren verschiedenen Relationen ausgedrückt werden. Hier geschieht dies durch v,n,g, die in beiden Relationen verwendet werden.

   Alternativ: explizite Ausformulierung der Joinbedingung:

   {[v,n, g,b] | [v,n, g,b] ∈ Kunde ∧

   ∃ v1,n1,g1,k ([v1,n1,g1,k] ∈ besitzt ∧ **v1=v ∧ n1=n ∧ g1=g** ∧ k=174266) }

5. Alle Kunden mit mehr als 5000 Euro Guthaben

   {[v,n, g,b] | [v,n, g,b] ∈ Kunde ∧ ∃ k ([v,n,g,k] ∈ besitzt ∧ **∃ x,y ([k,x,y]** ∈ **Konto** ∧ x>5000))}

6. Alle Kunden, die nur Girokonten haben (d.h. alle Kunden, die kein Festgeldkonto haben)

   Alle Kunden, die kein Festgeldkonto oder gar kein Konto haben:

   {[v,n, g,b] | [v,n, g,b] ∈ Kunde ∧ ¬ **∃ k** ([v,n,g,k] ∈ besitzt ∧
   ∃ x,y ([k,x,y] ∈ Festgeldkonto))}

   Alle Kunden, die ein Girokonto aber kein Festgeldkonto haben:

   {[v,n, g,b] | [v,n, g,b] ∈ Kunde ∧ **∃ k** ([v,n,g,k] ∈ besitzt ∧
   ¬ ∃ x,y ([k,x,y] ∈ Festgeldkonto))}

Aufgabe 3: SQL-Datentypen

Überlegen Sie, mit welchen SQL-Datentypen sie folgende Daten am besten darstellen:

1. Ergebnisse von wissenschaftlichen Berechnungen

   Ergebnisse von wissenschaftlichen Berechnungen Float, Real oder Double, da mit solchen Typen auch in wiss. Berechnungen gearbeitet wird. Man rechnet dabei mit der besten Genauigkeit, die man erreichen kann. Rundungsfehler können vorkommen, und müssen im Algorithmendesign berücksichtigt werden (siehe auch KoMa)

2. Kontostände, Umsätze und Bilanzen bei einer großen Bank

   Numeric mit einer festen Anzahl von Nachkommastellen.
   Banken rechnen normalerweise mit einer festen Anzahl von Stellen nach dem Komma, aber in diesem Bereich exakt (Rechnung auf 4 Stellen, beim Eintrag Rundung auf 2 oder 3 Stellen).

3. Studentenzahlen an den deutschen Universitäten

   (UNSIGNED) Smallint oder Integer.
   Studentenzahlen sind ganzzahlig, der Wertebereich muss über die erwartete Maximalzahl bestimmt werden.

4. Ortsnamen in Deutschland

   VARCHAR(n)
   Die Länge der Ortsnamen kann sehr stark schwanken, so dass CHAR(N) zu unnötiger Platzverschwendung führt. N muss gross genug sein, um auch den längsten Namen aufnehmen zu können, eine Schätzung liegt bei 60 Zeichen.

5. Raumbezeichnungen an der TU (MI 00.00.001)

   CHAR(N)
   Da die Raumbezeichnungen Buchstaben und Punkte enthalten, können keine Integer verwendet werden. Ein String fester Länge CHAR(N) ist hier wohl besser als ein VARCHAR, da die Länge der Raumbezeichnungen weitestgehend konstant ist, und sich damit der (kleine) Overhead von VARCHAR (Längenbezeichner) vermeiden lässt.

6. Satellitenbilder

   BLOB
   Satellitenbilder sind sehr groß (hunderte Megabyte bis mehrere Gigabyte), und ihre Daten und interne Struktur können nur schwer von einem relationalen DBMS ausgewertet werden.

7. Ein sehr langer Text (50MB) bestehend aus vielen einzelnen Wörtern und Sätzen.

   Die wohl beste Lösung ist es, die einzelnen Sätze als VARCHAR anzulegen und durchzunummerieren. Die nahe liegende Umsetzung in ein CLOB (wie BLOB, nur mit „normalen Zeichen“) hat den Nachteil, dass die Datenbank dann keine effektiven Zugriffe auf Teile des Textes zulässt. Wenn ein bestimmter Ausdruck gesucht wird, muss das Datenbanksystem den ganzen Text an ein externes Programm übergeben (z.B. grep), da es keine Information über den Aufbau des Textes hat. Bei einzelnen VARCHARs kann ein Index gebildet werden, und damit der Zugriff erheblich vereinfacht werden. Bei manchen Volltextdatenbanksystemen wird der Text sogar in einzelne Wörter zerlegt.Kontostände, Umsätze und Bilanzen bei einer großen Bank