Klausur Datenbanken
Wintersemester 2009/2010
Prof. Dr. Wolfgang May
2. Februar 2010, 14-16 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

Studiengang:

Bei der Klausur sind **keine Hilfsmittel** (Skripten, Taschenrechner, etc.) erlaubt. Handies müssen ausgeschaltet sein. Papier wird gestellt. Benutzen Sie nur die **ausgeteilten**, zusammengehefteten **Blätter** für Ihre Antworten. Schreiben Sie mit blauem/schwarzem Kugelschreiber, Füller, etc.; Bleistift ist nicht erlaubt.

Zum **Bestehen** der Klausur sind **45** Punkte hinreichend.

☐ meine Note soll mit Matrikelnummer so bald wie möglich auf der Vorlesungs-Webseite veröffentlicht werden.

☐ meine Note soll nicht veröffentlicht werden; ich erfahre sie dann aus FlexNever oder beim zuständigen Prüfungsamt.

	Max. Punkte	Schätzung für "4"
Aufgabe 1 (ER-Modell)	20	15
Aufgabe 2 (Transformation in das Relationale Modell)	20	13
Aufgabe 3 (SQL und Relationale Algebra)	38	21
Aufgabe 4 (Indexierung und Auswertung)	12	6
Summe	90	55

Note:

Themenstellung: Internet-Auktionsplattform

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll eine Datenbank einer Internet-Auktionsplattform, wie z.B. *ebay*, entworfen werden:

- Über alle Personen, die als Verkäufer oder als Käufer auftreten, werden einige Daten abgelegt: Name (Vor+Nachname; z.B. *"Max Müller"*; es kann mehrere Personen mit demselben Namen geben), eindeutige Benutzer-ID (z.B. *mmueller123*), Geburtsdatum, Ort, Strasse+Hausnummer, Land.

- Um das Angebot etwas zu strukturieren, wird jeder zu verkaufende Gegenstand einer Kategorie zugeordnet: *Computer*, *Bücher*, *CDs*, *Weine*, *Souvenirs*, etc.

- Aus rechtlichen Gründen muss jeder Benutzer für jede Kategorie, in der er als Verkäufer auftritt, angeben, ob er kommerziell oder privat verkauft. *Max Müller verkauft kommerziell Computer, sowie privat Geschirr und Bücher.*

Die zu verkaufenden Gegenstände und die Auktionen sind folgendermaßen beschrieben:

- Wenn ein Benutzer einen Gegenstand verkaufen möchte, erhält er eine ID. Dafür wird eine neue Auktion eröffnet. Der Benutzer muss den Gegenstand einer Kategorie, z.B. *"Bücher"* zuordnen, und eine kurze Beschreibung, z.B. *"Sizilianisch Kochen"*, angeben.

  Zusätzlich zu der Bezeichnung des Objektes kann eine textuelle Beschreibung, z.B. *"guter Zustand, einzelne Fettflecken"* angegeben werden. Das Ende des Auktionszeitraumes wird automatisch auf 14 Tage nach der Eröffnung gesetzt.

  Max Müller hat seit am 26.1.2010 das o.g. Buch zum Verkauf angeboten.

- Die Auktion verläuft dann wie üblich: Bieter geben Gebote ab, die jeweils höher als das aktuelle Höchstgebot sein müssen. Wenn der Zeitraum einer Auktion beendet ist, erwirbt der letzte Bieter (der also das höchste Gebot abgegeben hat), den Gegenstand. Nach Abschluss des Geschäfts können der Verkäufer und der Käufer jeweils Bewertungen (-3,...,+3) bezüglich der Fairness des Geschäftspartners abgeben.

- Zu den laufenden Auktionen wird jeweils nur das letzte Gebot mit den notwendigen Daten gespeichert. *Das bisher letzte Gebot für das o.g. Buch wurde am 1.2.2010 von Emma Schmidt abgegeben und es beträgt 8.50E.*

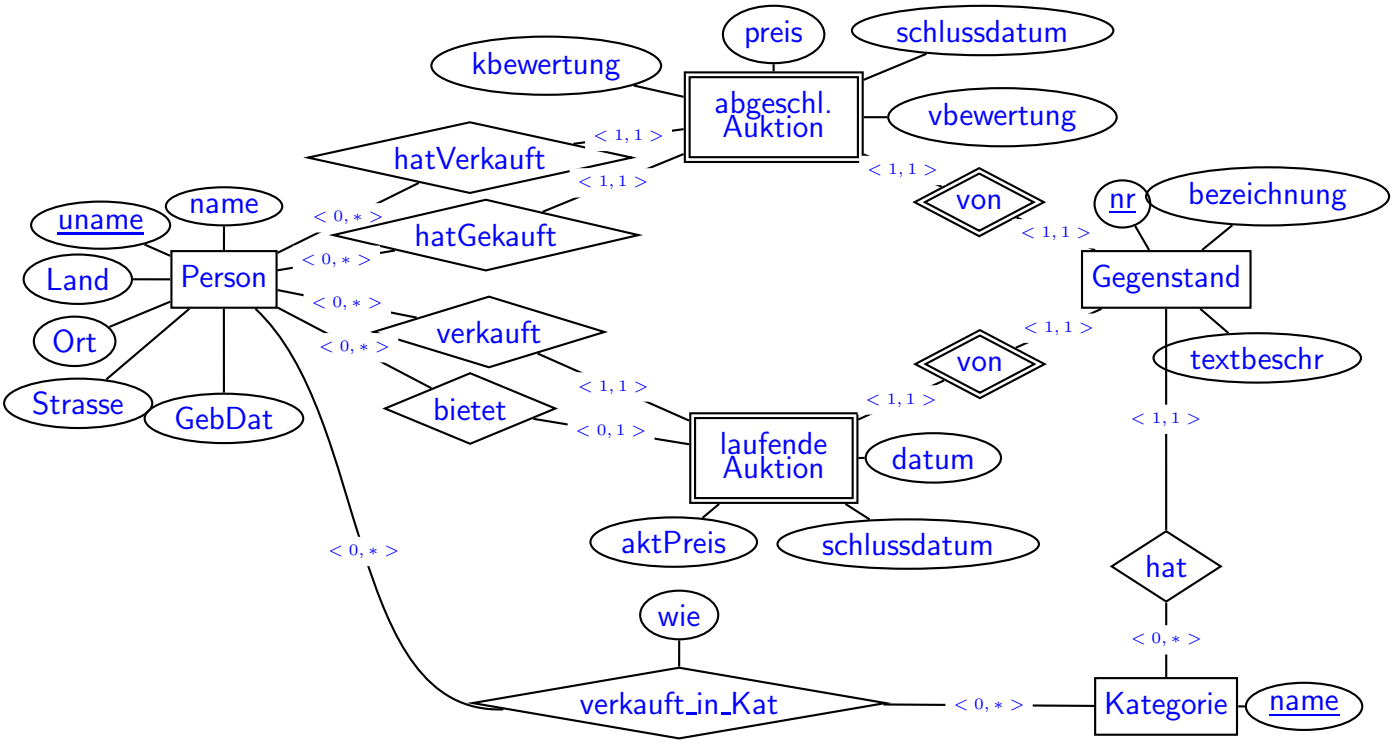
- Für alle beendeten Auktionen ist abgelegt, wer was von wem wann zu welchem Preis gekauft hat. Ausserdem werden die Bewertungen gespeichert.

  Max Müller hat in einer am 15.1.2010 abgelaufenen Auktion eine Flasche mit dem Bezeichnungstext "Flasche Château de Migraine, 1954" von Bernd Brot für 256E ersteigert. Er hat ihm dafür eine Bewertung von "-3" gegeben, da Bernd Brot nicht angegeben hatte, dass es sich um eine leere Flasche handelt. Umgekehrt hat Bernd Brot Max Müller mit "-3" bewertet, da er die Flasche nicht in der Kategorie "Wein", sondern unter "Souvenirs" eingeordnet hatte.

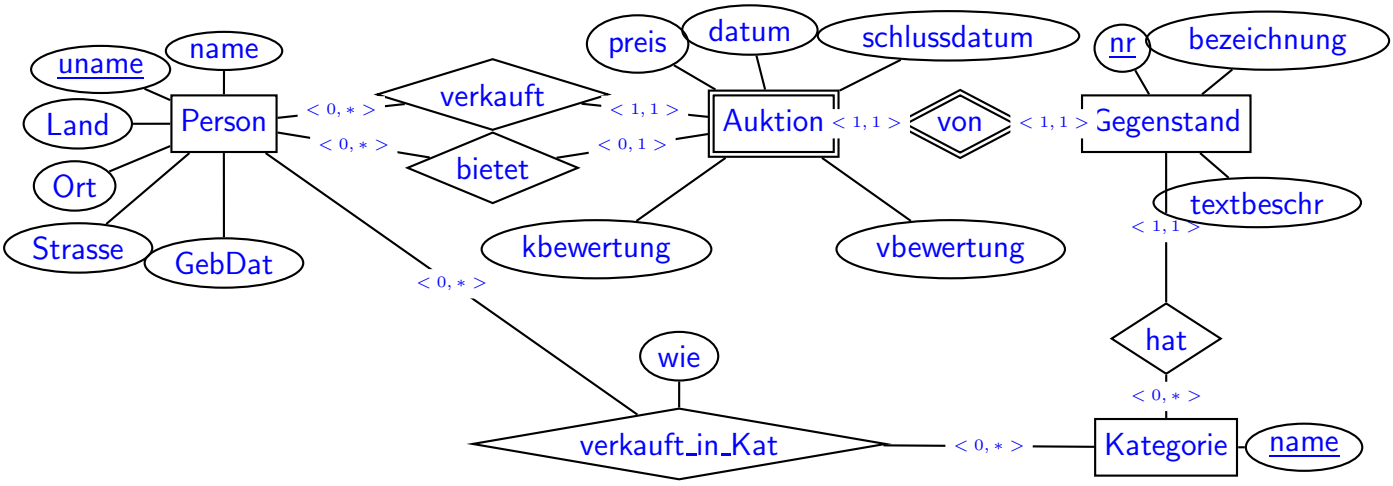
- Für Anfragen muss es möglich sein, zwischen abgeschlossenen und noch laufenden Auktionen zu unterscheiden.

Aufgabe 1 (ER-Modell [20 Punkte])
Entwickeln Sie ein ER-Modell für das Szenario. Geben Sie darin die Schlüsselattribute sowie die Beziehungskardinalitäten an.

Lösung



Alternative: laufende und abgeschlossene Auktionen nicht unterscheiden, evtl. ein Attribut laufend/abgeschlossen. Man kann dann aber auch nicht angeben, dass abgeschlossene Auktionen einen Käufer haben müssen (während noch nicht abgeschlossene evtl. noch keinen Bieter haben), und dass nur abgeschlossene Auktionen mit Bewertungen versehen sind.



Weitere Alternativen:

- Gegenstand und Aktion zusammenziehen,
- "datum" und "preis" als Attribute zu "bietet".

Aufgabe 2 (Transformation in das Relationale Modell [20 Punkte])

a) **Lösen Sie diesen Aufgabenteil auf dem *letzten* Blatt und trennen dieses ab (und geben es am Ende mit ab!).** Dann haben Sie dieses Blatt separat zugreifbar um später damit die Aufgaben 2b, 3 und 4 (SQL, Relationale Algebra+SQL, interne Auswertung) zu lösen.

Geben Sie an, welche Tabellen (mit Attributen, Schlüsseln etc.) Ihre Datenbank enthält (keine SQL `CREATE TABLE`-Statements, sondern einfach grafisch). (12 P)

Markieren Sie dabei auch Schlüssel (durch unterstreichen) und Fremdschlüssel (durch überstreichen).

Geben Sie die Tabellen mit jeweils mindestens zwei Beispieltupeln (z.B. denen, die sich aus dem Aufgabentext ergeben, und weiteren erfundenen) an.

b) Beschreiben Sie kurz, was entsprechend Ihrer Modellierung gemacht werden muss, wenn der Zeitraum einer Auktion abläuft. (2 P)

Lösung

- eine Tabelle "Kategorie" ist nicht unbedingt notwendig, da sie nur eine Spalte "Name" enthalten würde.
- die Beziehung zwischen Gegenstand und Kategorie wird in die Tabelle "Gegenstand" mit hineingenommen,

Person

uname	Name	Strasse	Ort	Land	Geburtsdatum
mmueller123	Max Müller	Mühlenstraße 1	Mühlhausen	D	01.01.1970
emma	Emma Schmidt	Donaugasse 42	Wien	A	08.08.1988
bbrot	Bernd Brot	Kuhdamm 456	Berlin	D	24.12.1950
:	:	:	:	:	

verkauft

Person	Kategorie	wie
pmueller	Computer	k
pmueller	Geschirr	p
pmueller	Buecher	p
:	:	:

Gegenstand

Nummer	Kategorie	Bezeichnung	Beschreibung
914278	Buecher	Sizilianisch Kochen	guter Zustand, einzelne Fettflecken
893681	Souvenirs	Fl. Chât. de Migraine, 1954	seltene Flasche, Originalkorken
:	:	:	:

Laufend

Gegenstand	Verkaeufer	Bieter	Gebot	Datum	Schluss
914278	pmueller123	eschmidt	8.50	1.2.2010	9.2.2010
:	:	:	:	:	:

Abgeschl						
Gegenstand	Verkaeufer	Kaeufer	Preis	SchlussDatum	VBewertung	KBewertung
893681	pmueller	bbrot	256.00	15.1.2010	-3	-3
:	:	:	:	:	:	:

Alternativen:

- Laufende und abgeschlossene Auktionen zusammen in einer Tabelle (das hängt von dem gewählten ER-Modell ab; Details siehe Teil b)),
- Die Attribute des Gegenstandes können (insbesondere wenn laufende und abgeschlossene Auktionen in derselben Tabelle stehen) mit in die Tabelle für Auktionen aufgenommen werden.

Ablaufen eines Auktionszeitraumes:

Hierzu verwendet man sinnvollerweise hier einen zeitgesteuerten Trigger:

- Getrennte Tabellen: Tupel aus *laufend* nach *abgeschl* rüberkopieren, und in *laufend* löschen.
- Gemeinsame Tabelle: ggf. *beendet* auf true setzen.

c) Geben Sie das `CREATE TABLE`-Statement für diejenige Tabelle (bzw. die Tabellen), in der bei Ihnen die Daten über die in der Aufgabenstellung beschriebene abgeschlossene Auktion der Flasche "Château de Migraine" abgespeichert sind, so vollständig wie möglich an (6 P).

Lösung

```
CREATE TABLE abgeschl                                    Basis: 2P
( Gegenstand NUMBER PRIMARY KEY REFERENCES Gegenstand.Nummer, 1/2 + 1/2
  Verkaeufer VARCHAR2(30) NOT NULL REFERENCES Person.uname,   1/2 + 1/2
  Kaeufer    VARCHAR2(30) NOT NULL REFERENCES Person.uname,   1/2 + 1/2
  Preis      NUMBER NOT NULL,                                 1/2 NOT NULL
  SchlussDatum DATE NOT NULL,
  VBewertung NUMBER CHECK (-3 <= VBewertung <= 3),            1/2 CHECK
  KBewertung NUMBER CHECK (-3 <= KBewertung <= 3))
```

Wenn man abgeschlossene und nicht abgeschlossene Auktionen in derselben Tabelle ablegt, muss hat man sich für die späteren Anfragen überlegen, wie man abgeschlossene von noch laufenden Auktionen unterscheidet. Dies kann man einfach ein Attribut *beendet*:{*yes*/*no*} machen.

```
CREATE TABLE Auktion                                     Basis: 2P
( Gegenstand NUMBER PRIMARY KEY REFERENCES Gegenstand.Nummer, 1/2 + 1/2
  Verkaeufer VARCHAR2(30) NOT NULL REFERENCES Person.uname,   1/2 + 1/2
  Kaeufer    VARCHAR2(30) REFERENCES Person.uname,            1/2 + 1/2
  Preis      NUMBER,
  Datum      DATE,
  SchlussDatum DATE NOT NULL,                                 1/2 NOT NULL
```

```
beendet     VARCHAR2(3) NOT NULL, CHECK ,
VBewertung NUMBER CHECK (-3 <= VBewertung <= 3),          1/2 CHECK
KBewertung NUMBER CHECK (-3 <= VBewertung <= 3))
```

Wenn man *beendet* nicht vorsieht, muss man in Anfragen *Schlussdatum* $<$ *SYSDATE* vergleichen.

Wenn man die Tabellen "Gegenstand" und "Auktion" (insbesondere wenn laufende und abgeschlossene Auktionen in derselben Tabelle stehen) zusammenfasst, kann man eine zusätzliche Integritätsbedingung

```
(Verkaeufer,Kategorie) REFERENCES verkauft.(Person,Kategorie)
```

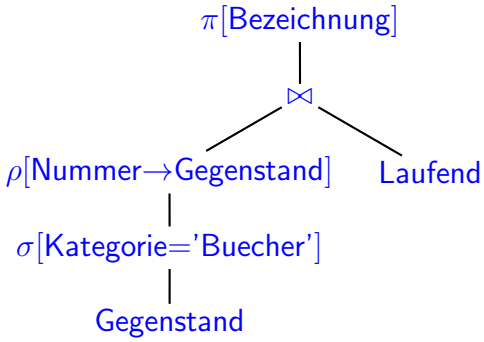
hinzunehmen, mit der auch gleich überprüft wird, ob der Verkäufer sich entsprechend registriert hat.

Aufgabe 3 (SQL und Relationale Algebra [38 Punkte])
Verwenden Sie für diese Aufgabe die von Ihnen entworfene relationale Datenbasis. Keine der Antworten soll Duplikate enthalten.

a) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die die Beschreibungen aller Bücher ausgibt, die momentan ersteigert werden können (2+2 P)

Lösung

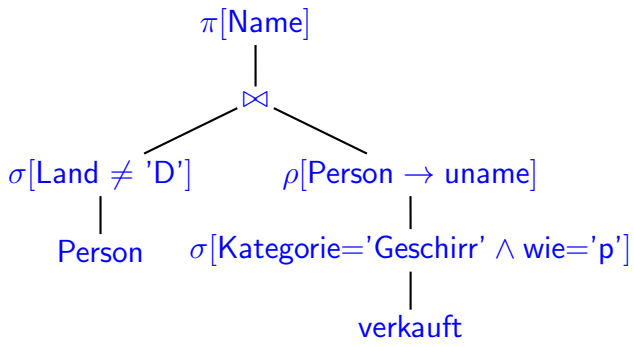
```
select distinct beschreibung
  -- distinct falls eine Beschreibung mehrfach auftritt
from gegenstand, laufend
where gegenstand.nummer = laufend.gegenstand
  and gegenstand.Kategorie='Buecher';
```



b) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die die Namen aller Personen ergeben, die privat Geschirr verkaufen und nicht in Deutschland wohnen. (3+3 P)

Lösung

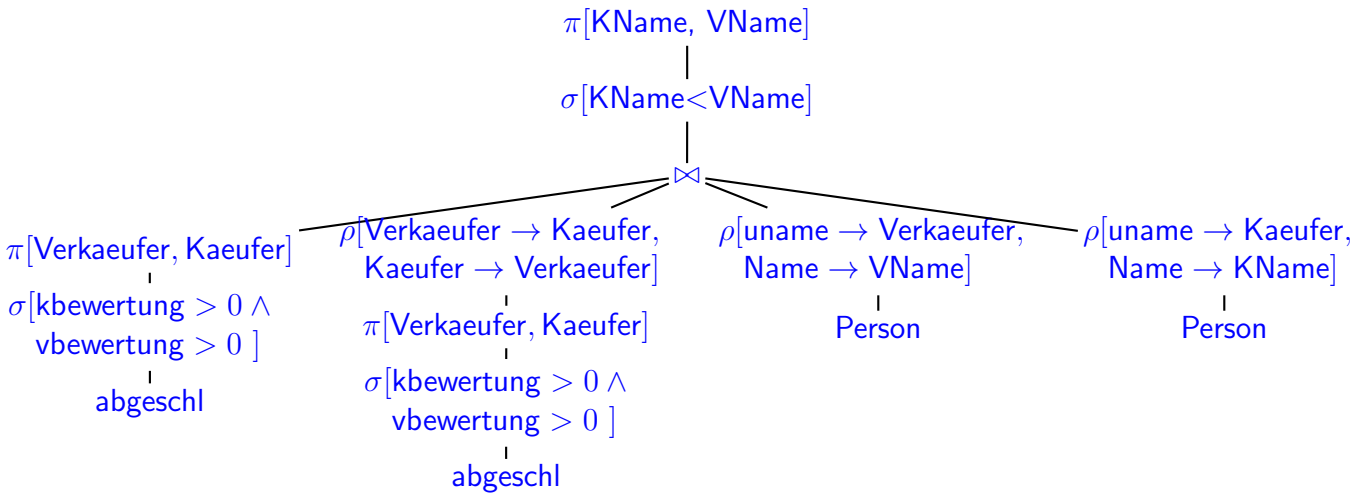
```
select person.name -- kein distinct notwendig da (Person,Kategorie)
                   -- in verkauft key ist
from person, verkauft
where person.uname = verkauft.Person
  and verkauft.Kategorie = 'Geschirr'
  and verkauft.wie = 'p'
  and not Person.Land = 'D';
```



c) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, die alle Paare von Namen von Personen (A, B) berechnen, so dass B und A sich bereits in beide Richtungen etwas verkauft haben, und sich dabei auch jeweils gegenseitig positiv bewertet haben: (4+4 P)

Lösung

```
select distinct p1.name, p2.name
from abgeschl a1, person p1, abgeschl a2, person p2
where a1.verkaeufer = a2.kaeufer
  and a2.verkaeufer = a1.kaeufer
  and p1.uname = a1.verkaeufer
  and p2.uname = a1.kaeufer
  and p1.uname > p2.uname   -- dann werden nicht (a,b) und (b,a) ausgegeben
  and a1.kbewertung > 0 and a1.vbewertung > 0
  and l1.kbewertung > 0 and a2.vbewertung > 0;
```

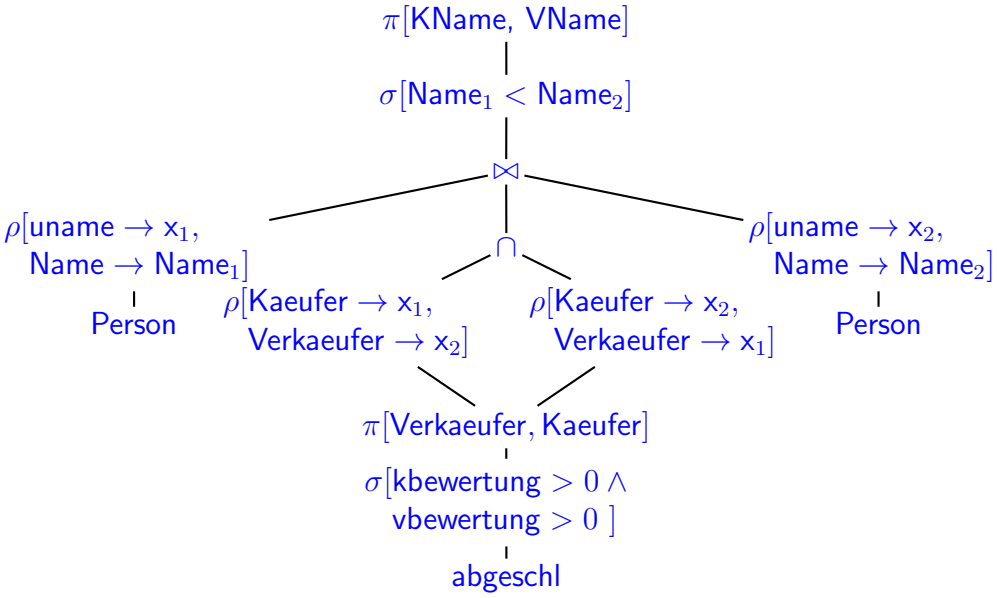


Eine andere Lösung verwendet kein Join, sondern zwei EXIST oder eine Schnittmenge um das gegenseitige Verkaufen zu testen:

```
select p1.name, p2.name  -- distinct dann nicht notwendig
from person p1, person p2
where p1.uname > p2.uname   -- dann werden nicht (a,b) und (b,a) ausgegeben
  and exists (select *
              from auktion
              where verkaeufer = p1.uname and kaeufer = p2.uname
                and kbewertung > 0 and vbewertung > 0)
  and exists (select *
              from auktion
              where verkaeufer = p1.uname and kaeufer = p2.uname
                and kbewertung > 0 and vbewertung > 0)

select p1.name, p2.name  -- distinct dann nicht notwendig
from person p1, person p2
where p1.uname > p2.uname   -- dann werden nicht (a,b) und (b,a) ausgegeben
  and (p1.uname, p2.uname)
      in ((select verkaeufer as x1, kaeufer as x2
           from auktion
           where kbewertung > 0 and vbewertung > 0)
          intersect
          (select verkaeufer as x2, kaeufer as x1
           from auktion
           where kbewertung > 0 and vbewertung > 0))
```

In der Algebra kann die Schnittmenge ausdrücken, das äußere IN entspricht einem Join:



d) Geben Sie eine **SQL-Anfrage** an, die die Namen aller Personen ausgibt, welche insgesamt für mehr als 100.000E Gegenstände kommerziell verkauft haben (4 P).

Lösung

```
SELECT Name -- kein distinct notwendig wegen group by
FROM Person, verkauft, abgeschl, Gegenstand
WHERE Person.uname = verkauft.Person
  AND verkauft.wie = 'k'
  AND abgeschl.Verkaeufer = Person.uname
  AND abgeschl.Gegenstand = Gegenstand.nummer
  AND verkauft.Kategorie = Gegenstand.Kategorie
GROUP BY Person.uname
HAVING sum(abgeschl.preis) > 100000;
```

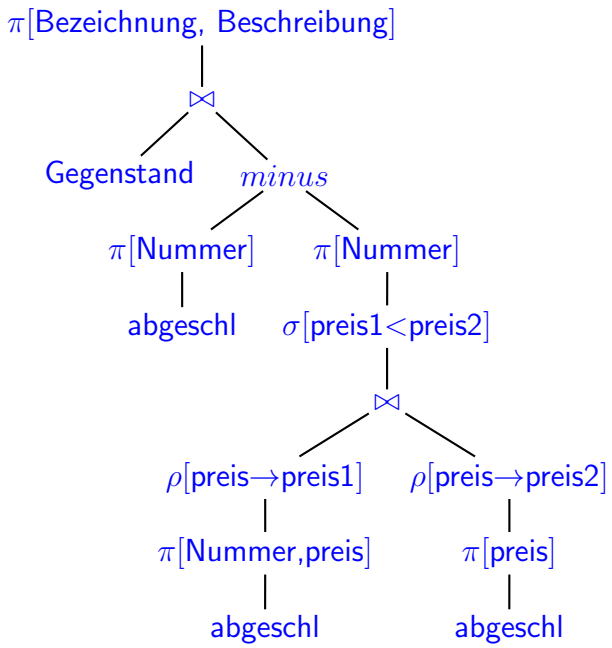
e) Geben Sie **eine SQL-Anfrage und einen Algebra-Ausdruck** an, der die Bezeichnung und Beschreibung des Gegenstandes (oder der Gegenstände, falls es mehrere solche gibt) ausgibt, die von allen bereits abgeschlossenen Auktionen den höchsten Preis erzielt haben (3+5 P).
(Sie dürfen `ROWNUM` **nicht** verwenden.)

Lösung

```
SELECT DISTINCT Bezeichnung, Beschreibung
FROM Gegenstand, abgeschl
WHERE Gegenstand.nr = abgeschl.Gegenstand
  AND abgeschl.preis =
   (SELECT MAX(preis)
    FROM abgeschl);

SELECT DISTINCT Bezeichnung, Beschreibung
FROM Gegenstand, abgeschl a1
WHERE Gegenstand.nr = abgeschl.Gegenstand
  AND NOT EXISTS
   (SELECT *
    FROM abgeschl a2
    WHERE a2.preis > a1.preis);
```

Algebra: innere Anfrage: die Nummern aller versteigerten Gegenstände, ohne die Nummern derjenigen, für die es einen Gegenstand gibt, der teurer ist. Das Ergebnis dann mit der Tabelle "Gegenstand" joinen um Bezeichnung und Beschreibung zu bekommen.



f) Ein bisschen Theorie: Gegeben seien zwei Relationen $R[\bar{X}]$ und $S[\bar{X}]$ mit demselben Format. R enthalte m Tupel, S enthalte n Tupel.

- Betrachten Sie $R \cup S$. Wie groß kann die Anzahl k der Tupel in $R \cup S$ minimal bzw. maximal (bzgl. m und n) sein (mit Begründung) (2+2 P)?
- Betrachten Sie $R \bowtie S$. Welche (genaue!) Aussage können Sie über die Anzahl der Tupel in $R \bowtie S$ machen, wenn Sie $k = |R \cup S|$ kennen (mit Begründung) (4 P)?

Lösung

- minimales k: wenn eine der Relationen in der anderen enthalten ist, z.B. wenn $R \subseteq S$, dann ist $R \cup S = S$ und damit $|R \cup S| = n$. Analog umgekehrt. Allgemein also $|R \cup S| \geq max(n, m)$.
- maximales k: wenn R und S disjunkt sind. Also $max(n, m) \leq k \leq n + m$.
- Join: da beide Relationen dasselbe Format $\bar{X}$ haben, ist $R \bowtie S = R \cap S$.
  Mengentheoretisch gilt $|R \cup S| = |R| + |S| - |R \cap S|$, und damit durch Umformung auch $|R \cap S| = |R| + |S| - |R \cup S|$.
  Man kann es sich aber auch von Grund auf überlegen: Wenn in obigem Fall $k < n + m$ ist, gab es einige Tupel, die in beiden Relationen vorhanden sind, nämlich genau $n + m - k$ viele. Also ist $|R \bowtie S| = n + m - k$.

Aufgabe 4 (Indexierung und Auswertung [12 Punkte])

a) Geben Sie eine SQL-Anfrage an, die die Benutzernamen aller Personen ergibt, die in *Hamburg* wohnen und einen Gegenstand für mehr als 100.000 Euro gekauft haben (d.h., es werden nur abgeschlossene Auktionen betrachtet!). (3 P)

b) Welche Art Index unterstützt eine entsprechende Suche auf der Spalte *Preis*? (1 P)

c) Beschreiben Sie, wie die o.g. SQL-Anfrage möglichst effizient ausgewertet wird. Sie dürfen dabei weitere Maßnahmen aus dem Kapitel "Interne Organisation und Implementierung" annehmen, die Sie für vernünftig halten.

- Beschreiben Sie grob Ihr gewähltes Vorgehen (2 P).
- Geben Sie eine Skizze an, wie Ihre Tabellen mit den Indexen etc., die Sie bei Ihrem Vorgehen nutzen, aussehen (2 P).
- Geben Sie die Anzahl der Seitenzugriffe (grobe Überschlagsrechnungen) an. Rechnen Sie nur da, wo es wirklich notwendig ist (4 P).

Gehen Sie bei Ihren Größenabschätzungen von folgenden Daten aus:

- Eine Speicherseite umfasst 4096 Bytes,
- Es sind 1.000.000 Kunden, 30.000.000 abgeschlossene Auktionen und 20.000 noch laufende Auktionen gespeichert.
- 10.000 Kunden wohnen in Hamburg, und 600 Auktionen hatten einen Schlusspreis von mehr als 100.000E.
- Nehmen Sie an, dass im Hauptspeichercache Indexseiten behalten werden, während normale Tabellen-Speicherungs-Seiten, wenn man sie nicht tupelweise iterativ durchgeht, gleich wieder ausgelagert werden.

Lösung

a)

```
SELECT p.uname
FROM Person p
WHERE p.ort = 'Hamburg'
  AND EXISTS
  (SELECT *
   FROM Abgeschl a
   WHERE a.kaeufer = p.uname
     AND a.preis > 100000)

(SELECT p.uname
 FROM Person p
 WHERE p.ort = 'Hamburg')
INTERSECT
(SELECT kaeufer
 FROM Abgeschl a
 WHERE a.preis > 100000)

SELECT DISTINCT p.uname
FROM Person p,  Abgeschl a
WHERE p.ort = 'Hamburg'
```

```
AND p.uname = a.kaeufer
AND a.preis > 100000
```

b) Index auf Preis: Um Bereichssuche zu ermöglichen, ist ein B*-Baum die richtige Wahl.

Die Größenberechnung wird hier nicht in der Antwort erwartet (ob es nun 3 oder 4 Ebenen sind, ist spielt beim Überschlag keine Rolle):
Blätter: Einträge sind Paare (Preis, Referenz auf Tabelle (Tabellenseitennummer, Tupelnummer)), jeweils 4+4+1 Byte. Damit ist $\ell = 450$ (Anzahl Einträge pro Blatt). Bei durchschnittlichem Füllgrad von 66% also ca. 300 Einträge. Also benötigt man etwa 100.000 Speicherseiten für die Blätter.
Innere Knoten: Für (Preis, Referenz auf B-Baum-Kindknoten, linkes und rechtes internes Kind) benötigt man 4+4+1+1 Byte, also kann man 400 Einträge speichern, Bei durchschnittlichem Füllgrad von 75% also auch ca. 300 Einträge. 100.000/300 = 330 Knoten auf unterer Ebene, also noch eine weitere innere Ebene und eine Wurzel.

c) 3 Möglichkeiten: Bedingungen separat auswerten, und dann joinen/Schnittmenge berechnen (siehe (c1)), oder basierend auf einer der Bedingungen gezielt zugreifen ((c2) und (c3)). Prinzipiell fängt man dabei besser mit der selektivsten Bedingung an. Hier ist das also die 600 Aktionen über 1.000.000E, und nicht die 10.000 Hamburger Kunden (die an wahrscheinlich 300.000 Auktionen beteiligt waren).

c1) Man fängt mit der Bedingung `preis > 1000000` an. Nach Annahme bekommt man 600 Auktionen.

Um diese (bzw. Referenzen auf diese) zu bekommen kann man im B*-Baum nach 100.000 suchen (z.B. 4 Schritte), und von dort aus die Blattknoten linear bis ans Ende durchgehen (also z.B. 3 Seiten); zusammen 7 Zugriffe. (man kann auch einfach die Blätter von hinten linear bis hinunter zur 100.000 durchgehen! - dann sind es nur 3 Zugriffe)

Ob es nun 3 oder 7 sind, ist ziemlich egal: Man bekommt 600 Auktionen, die mit hoher Wahrscheinlichkeit auf verschiedenen Seiten liegen. Also ca. 600 Zugriffe, um die 600 Tupel und damit die Benutzernamen (vielleicht 598 verschiedene – man rechnet wieder mit 600 weiter) zu bekommen. (Falls abgeschlossene und laufende Auktionen in einer gemeinsamen Tabelle gespeichert sind: jedes Auktions-Tupel noch auf *beendet* prüfen.)

Man legt diese ab (600 · 10 Zeichen sind 6K, die kann man als Zwischenergebnis behalten).

Als nächstes selektiert man alle Benutzernamen von Kunden in Hamburg. Wenn man keinen Index auf "Ort" hat, hat man auf den ersten Blick ziemlich verloren, weil man die Personen-Tabelle linear durchgehen muss: Ein Person-Tupel hat ca. 60 Byte, also 70 Tupel pro Seite. Bei 1.000.000 Kunden sind das dann 14.000 Seiten. Man muss alle durchgehen, um die Benutzernamen der 10.000 Hamburger Kunden zu finden.

Dabei stellt man aber fest, dass wenn man einen (Hash-)Index hätte, man die Kachel(n) für Hamburg (das sind auch ca. 300 Stück) zugreifen muss, und dann auch noch ca. 6000 Speicherseiten (die Hamburger Kunden liegen verteilt).

Einen realen Vorteil hat man, wenn man annimmt, dass *Person* nach Ort geordnet gespeichert ist (was aber an sich für die Anwendung keinen tieferen Sinn hat). Dann kommt man mit ca. 140 Seitenzugriffen aus.

Für jeden Benutzer prüft man sofort, ob er in dem obigen Zwischenergebnis enthalten ist (keine weiteren Zugriffe).

Anzahl der Zugriffe: 600 + 10000 = 10600.

c2) Man fängt wie in (c1) mit der inneren Subquery an: alle Auktionen, die einen Schlusspreis von über 100.000E hatten, nach Annahme sind das 600 Stück, die man mit 600 Seitenzugriffen bekommt.

Als nächstes muss man anhand von *Käufer (= Person.uname)* auf das Person-Tupel zugreifen. Hierzu setzt man einen Index auf Person.uname (Key!) voraus. Da man hier normalerweise keine Bereichssuche macht, kann man einen Hash-Index und Zugriff in $O(1)$ annehmen, und hat mit weiteren $2 \cdot 600$ Zugriffen (jeweils in die Hash-Kachel und dann auf das Tupel) die Orte und kann selektieren.

Wenn man hier statt Hash einen B*-Baum annimmt, hätte dieser 1.000.000 Einträge, z.B. 200 pro Blatt, also 5000 Blätter; bei 200 Einträgen in den inneren Knoten 25 Knoten auf unterer Ebene und eine Wurzel. Man würde einmal die inneren Knoten und die Wurzel in den Cache laden (26 Zugriffe), und dann allenfalls das jeweilige Blatt und auch wieder das Tupel aus dem Hintergrundspeicher holen müssen, also auch $2 \cdot 600$)

Nachdem 10.000 von 1.000.000 Kunden (also 1/100) in Hamburg wohnen, ist anzunehmen, dass 6 Ergebnisse ausgegeben werden.

Das war aber nicht gefragt, sondern die Anzahl der Zugriffe: 3(7) + 600 + (26) + $2 \cdot 600$ = 1803, 1807, 1829, 1836

c3) Man fängt mit den Kunden in Hamburg an. Wie in (c1) benötigt man 14000, 6000, oder 140 Seitenzugriffe, um die 10.000 Benutzernamen zu holen.

Um deren Käufe zu finden nimmt man einen Hash-Index auf *Abgeschl.Käufer* an. In diesem greift man erst auf die Hash-Kacheln der 10.000 Hamburger Personen zu (die durchschnittlich 30 Auktionen pro Person liegen dann in derselben Kachel, aber nur selten hat man die Kachel einer Person zufällig gerade auch schon von einer vorhergehenden Person im Cache - ca. 9000 Zugriffe).

Die 10.000 Hamburger Kunden (1/100 des Bestandes) sind statistisch bei 300.000 Auktionen Käufer.

* Man kann jetzt auf die Auktionstupel zugreifen, und jedes auf seinen Preis überprüfen. Die Tupel liegen verteilt auf ca. 300.000 Speicherseiten. Dort kann man jetzt nicht annehmen, dass die Tupel nach den Käufern geordnet/gruppiert sind (eher schon nach den Verkäufern). Statistisch muss man davon ausgehen, dass man auf 150.000-250.000 Seiten zugreifen muss, um alle gesuchten Auktionen zu testen.
  Summe: 10.000 (oder 6.300 oder 140) + 9.000 + 150.000 → 170.000 Zugriffe.
* Man kann aber auch einfach mal nur die Referenzen auf die 300.000 Auktionen temporär in einer Menge (als HashSet ca. 4+1 Byte pro Eintrag also 800 pro Seite, sind 400 Speicherseiten) ablegen (das ist 1.6MB, die passen in den Cache).
  Als zweites greift man (wie in Fall (c1)) über den Index auf Abgeschl.Preis zu, und bekommt mit 3 (oder 7) Zugriffen die Referenzen der 600 Auktionen, die mit mehr als 100.000E endeten. Diese kann man gleich mit dem HashSet von oben vergleichen, und ggf. ausgeben.
  Summe: 10.000 (oder 6.300 oder 140) + 9.000 = 19000 (oder 9140)