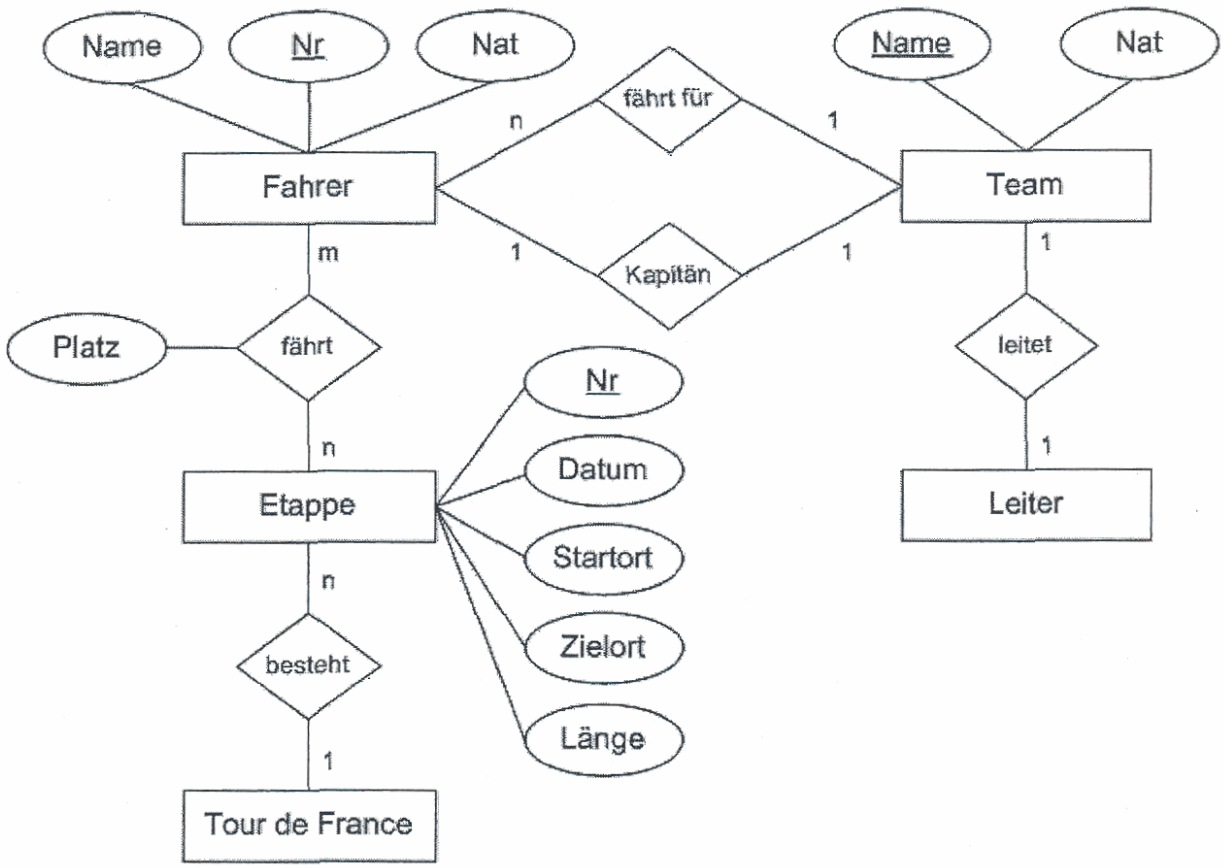
Aufgabe 1: Relationenalgebra - Relationenkalkül (4 Punkte)

In der Relationenalgebra werden die gewünschten Relationen durch Angabe einer Folge von Operationen, mit deren Hilfe diese Relationen aufgebaut werden sollen, angegeben. Der Benutzer wendet dabei spezielle Operationen auf Relationen an, um seine gewünschte Relation zu konstruieren. (2 Punkte)

Beim Relationenkalkül dagegen gibt der Benutzer die Definition der von ihm gewünschten Relation in den Begriffen schon gegebener Relationen an. Er sagt dabei nichts über die anzuwendenden Operationen aus. (2 Punkte)

Aufgabe 2: Entity-Relationship-Diagramm (10 Punkte)



Aufgabe 3: Datenmodellierung **(12 Punkte)**

a) Pfleger (Name, Adresse)
Tier (Name, Art, Pfleger_Name, GebäudeNr)
Futter (FNr, Name, Lieferant)
Gebäude (Nr, Größe)
Futterbedarf (Tiername, FNr, Menge)
Lagerung (FNr, GebäudeNr)

b)
```
CREATE TABLE Pfleger
    (Name CHAR(50) NOT NULL,
    Adresse CHAR(50),
    PRIMARY KEY (Name))

CREATE TABLE Tier
    (Name CHAR(50) NOT NULL,
    Art CHAR(50),
    Pfleger_Name CHAR(50)NOT NULL,
    GebäudeNr INTEGER NOT NULL,
    PRIMARY KEY (Name),
    FOREIGN KEY (Pfleger_Name) REFERENCES Pfleger (Name),
    FOREIGN KEY (GebäudeNr) REFERENCES Gebäude (Nr))

CREATE TABLE Futter
    (FNr INTEGER NOT NULL,
    Name CHAR(50),
    Lieferant CHAR(50),
    PRIMARY KEY (FNr))

CREATE TABLE Gebäude
    (Nr INTEGER NOT NULL,
    Größe INTEGER NOT NULL,
    PRIMARY KEY (Nr))

CREATE TABLE Futterbedarf
    (Tiername CHAR(50) NOT NULL,
    FNr INTEGER NOT NULL,
    Menge INTEGER,
    PRIMARY KEY (Tiername, FNr)
    FOREIGN KEY (Tiername) REFERENCES Tier (Name),
    FOREIGN KEY (FNr) REFERENCES Futter (FNr))

CREATE TABLE Lagerung
    (FNr INTEGER NOT NULL,
    GebäudeNr INTEGER NOT NULL,
    PRIMARY KEY (FNr, GebäudeNr)
    FOREIGN KEY (FNr) REFERENCES Futter (FNr),
    FOREIGN KEY (GebäudeNr) REFERENCES Gebäude (Nr))
```

Aufgabe 4: Queries (14 Punkte)

a) (1 Punkt)

```
SELECT MatrNr, Name
FROM Student
WHERE Semester = 1
```

b) (1 Punkt)

```
SELECT COUNT(*)
FROM Student
WHERE Semester >= 7 and Semester <= 10
```

alternativ:

```
WHERE Semester between 7 and 10
```

alternativ:

```
WHERE Semester in (7, 8, 9, 10)
```

c) (2 Punkte)

```
SELECT *
FROM Vorlesung
WHERE UPPER(Titel) LIKE '%DATENBANK'
```

d) (3 Punkte)

```
SELECT s.Name, v.Titel
FROM Student s, hoeren h, Vorlesung v
WHERE s.MatrNr = h.MatrNr
    AND h.VorlNr = v.VorlNr
ORDER BY s.Name ASC, v.VorlNr ASC
```

e) (3 Punkte)

```
SELECT p.Name, SUM(SWS)
FROM Vorlesung, Professor p, lesen l
WHERE VorlNr = l.VorlNr
    AND l.ProfNr = p.ProfNr
    AND p.Rang = 'C4'
GROUP BY l.ProfNr, p.Name
```

f) (4 Punkte)

```
SELECT s.*
FROM Student s
WHERE NOT EXISTS
    (SELECT *
    FROM Vorlesung v
    WHERE v.SWS = 4 AND NOT EXISTS
            (SELECT *
            FROM hoeren h
            WHERE h.VorlNr = v.VorlNr
            AND h.MatrNr = s.MatrNr))
```

Aufgabe 5: Normalformen (10 Punkte)

a) Die Relation befindet sich in der 1. NF, da alle Attribute atomar sind.

Der einzige Schlüssel der Relation ist TIER_NR, und alle Attribute sind voll funktional abhängig von TIER_NR. Damit befindet sich die Relation auch in 2. NF.

Es gibt die Abhängigkeit ABT_NAME → PFLEGER. Damit haben wir eine Abhängigkeit, die von einem Nichtschlüsselattribut ausgeht. Damit ist die Relation nicht in 3NF und damit auch nicht in BCNF.

b) Die Relation kann zerlegt werden, indem die Information über Abteilungen und Pfleger und ihre Zusammenhänge in eine eigene Relation ausgelagert wird:

Wir zerlegen die Relation in eine Relation über Tiere und ihre Zuordnung zu Abteilungen und eine zweite Relation über Abteilungen und ihre Pfleger:

TIER_NR	NAME	ABT_NAME
14	Bongo	Gehege1
17	Clara	Haus3
25	Admiral	Gehege1
29	Streik	Gehege1
40	Herbert	Gehege2
67	Anton	Haus3
81	RedBull	Haus3
89	Lenzen	Gehege2

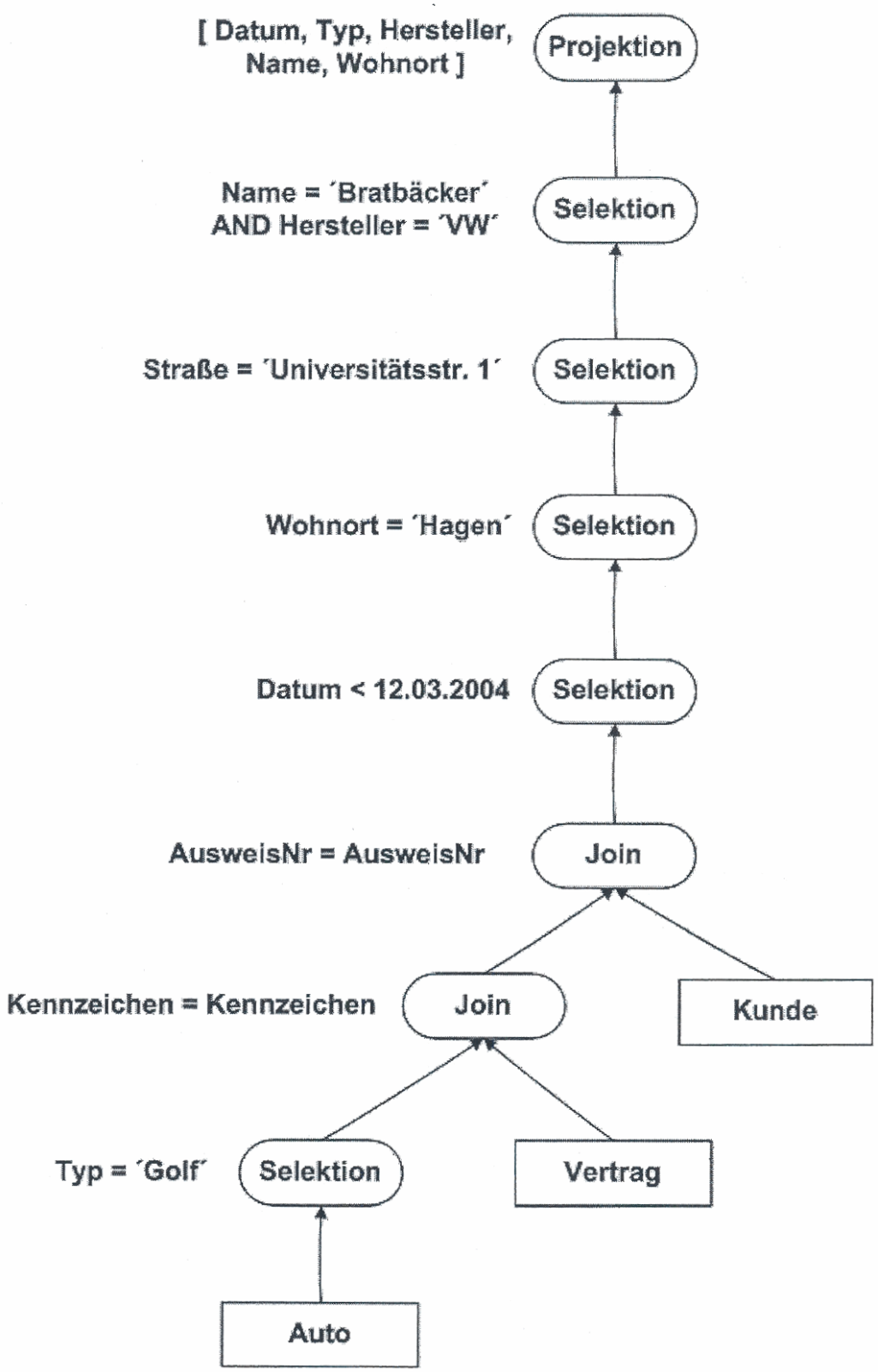
ABT_NAME	PFLEGER
Gehege1	Krause
Haus3	Müller
Gehege2	Meyer

Aufgabe 6: Funktionale Abhängigkeiten (10 Punkte)

(i) A → B, kann nicht gelten wegen Zeilen #1 und #2
(ii) B → C, kann gelten, da zu einem B-Wert immer derselbe C-Wert gehört
(iii) C → B, kann nicht gelten wegen Zeilen #1 und #3
(iv) B → A, kann nicht gelten wegen Zeilen #1 und #5, #4 und #6
(v) C → A, kann nicht gelten wegen Zeilen #1 und #3 und #5, #4 und #6
(vi) A → C, kann nicht gelten wegen Zeilen #1 und #2

Aufgabe 7: Anfrageoptimierung (14 Punkte)

a) Operatorbaum der Anfrage (2 Punkte):

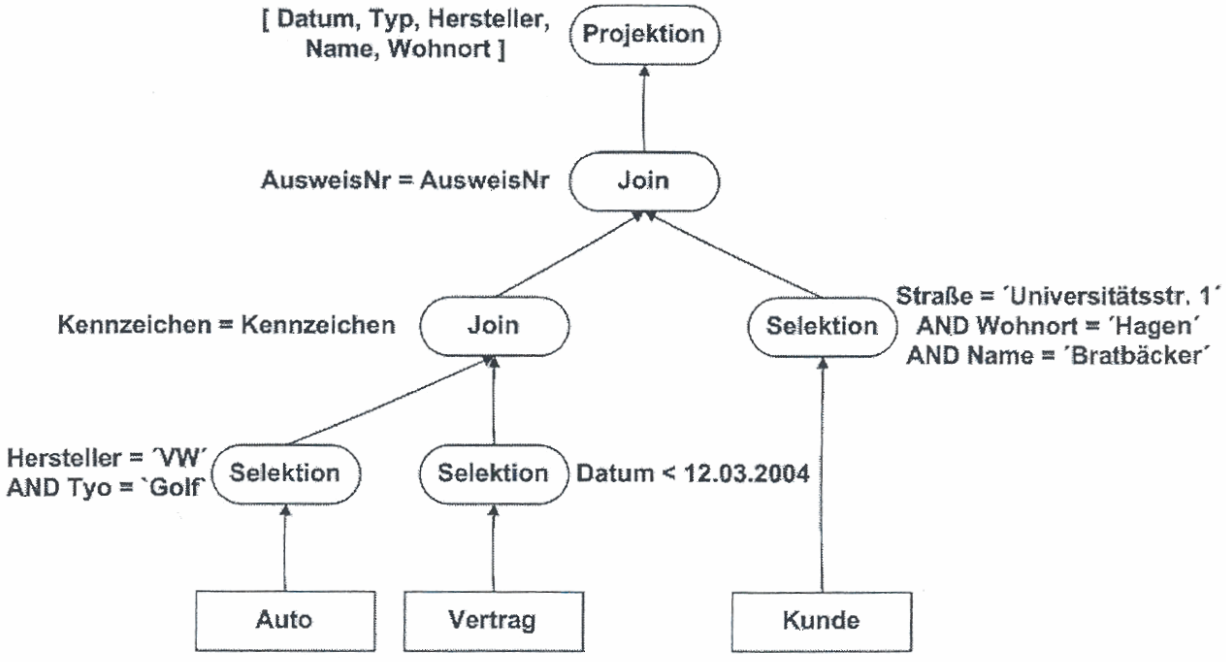


b) Optimieren nach heuristischen Regeln (6 Punkte):
Die Selektion **[Datum < 12.03.2004]** bezieht sich nur auf die Relation **Vertrag** und kann deshalb vor dem Verbund **Auto [Kennzeichen = Kennzeichen] Vertrag** ausgeführt werden.
[Wohnort = ´Hagen´] [Straße = ´Universitätsstr. 1´] sind Selektionen, die sich beide auf die Relation **Kunde** beziehen. Deshalb ist es möglich, diese zu der Selektion **[Wohnort = ´Hagen´ AND Straße = ´Universitätsstr. 1´]** zusammenzufassen und vor der Verbundoperation auszuführen.
Die Selektion **[Name = ´Bratbäcker´ AND Hersteller = ´VW´]** kann aufgespalten werden, wobei sich die Teilselektion **[Name = ´Bratbäcker´]** ausschließlich auf die Relation **Kunde** und die Teilrelation **[Hersteller = ´VW´]** ausschließlich auf die Relation **Auto** bezieht. Diese beiden Selektionen können dann vor den Verbünden durchgeführt werden.

Optimierte Anfrage:

(Vertrag [Datum < 12.03.2004]) [Kennzeichen = Kennzeichen] (Auto [Hersteller = ´VW´ AND Typ = ´Golf´]) [AusweisNr = AusweisNr] (Kunde [Wohnort = ´Hagen´ AND Straße = ´Universitätsstr. 1´ AND Name = ´Bratbäcker´]) [Datum, Typ, Hersteller, Name, Wohnort]

Resultierender Operatorbaum (2 Punkte):



c) Die notwendigen Verbund-Operationen (z.B. R [A = B] S) lassen sich auf physischer Ebene noch optimieren durch:

- Ausnutzung von Sekundärindexen (2 Punkte):
  Existiert ein Index für B, dann ist für jedes Tupel in R sofort die zugehörige Tupelmenge in S feststellbar – und damit die Tupelmenge des Verbundes konstruierbar. Die Zahl der notwendigen Zugriffe auf Tupel wird damit linear zur Größe der beiden Relationen
- Sortieren von R nach A und von S nach B (2 Punkte):

Bei effizienten Sortierverfahren werden O(n log n) Schritte benötigt. Das anschließende Erstellen des Verbundes ist dann sehr schnell: R und S werden parallel durchlaufen, wobei jeweils auf gleiche Werte für A und B zu prüfen ist.

Aufgabe 8: Recovery (10 Punkte)

a)

⇒ T1 muss nicht wiederholt werden, da sie bereits vor dem Checkpoint beendet wurde. (1 Punkt)

⇒ Für T2 müssen die After-Images aus dem Logfile in die Datenbank geschrieben werden. Diese brauchen nur bis zum letzten Checkpoint zurückreichen, da alle davorliegenden "After-Images" bereits in die Datenbank eingetragen worden sind. (1 Punkt)

⇒ Für T3 muss ein Rollback ausgeführt werden. (1 Punkt)

⇒ Für T4 müssen die After-Images aus dem Logfile in die Datenbank geschrieben werden. (1 Punkt)

⇒ Für T5 muss ein Rollback ausgeführt werden. (1 Punkt)

b)

⇒ T1 muss nicht wiederholt werden. (1 Punkt)

⇒ T2 muss nicht wiederholt werden, da das Herausschreiben der veränderten Daten bei EOT geschah. (1 Punkt)

⇒ Für T3 muss ein Rollback ausgeführt werden. (1 Punkt)

⇒ T4 muss nicht wiederholt werden, da das Herausschreiben der veränderten Daten bei EOT geschah. (1 Punkt)

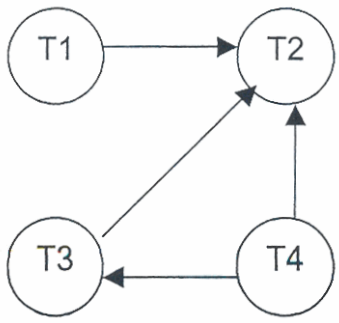
⇒ Für T5 muss ein Rollback ausgeführt werden. (1 Punkt)

Aufgabe 9: Serialisierbarkeit (6 Punkte)

a) (3 Punkte)

Konflikte in L_1:

$w_4(a) \rightarrow r_3(a)$,
$r_3(b) \rightarrow w_2(b)$,
$w_1(c) \rightarrow r_2(c)$,
$w_4(d) \rightarrow r_2(d)$

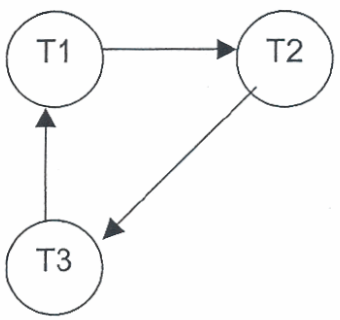


Der Abhängigkeitsgraph enthält keinen Zyklus. Somit ist die Schedule serialisierbar. Serielle Schedule: T4, T3, T1, T2

b) (3 Punkte)

Konflikte in L_2:

$w_2(a) \rightarrow r_3(a)$,
$r_3(b) \rightarrow w_1(b)$,
$w_1(c) \rightarrow r_2(c)$



Der Abhängigkeitsgraph enthält einen Zyklus. Die Schedule ist nicht serialisierbar.

Aufgabe 10: Zugriffsschutz **(14 Punkte)**

a) (je 3 Punkte)

```
CREATE VIEW V_Angestelltenname (Name) AS
    SELECT Name
    FROM Angestellter

CREATE VIEW V_Abteilungsinfo (Abteilung, Durchschnittsgehalt) AS
    SELECT DISTINCT Abteilung, AVG (Gehalt)
    FROM Angestellter
    GROUP BY Abteilung
```

b) (je 2 Punkte)

```
GRANT DELETE ON Angestellter TO Meier
GRANT SELECT ON V_Angestelltenname TO Meier
GRANT SELECT ON V_Abteilungsinfo TO Meier
```

c) (2 Punkte)

```
GRANT SELECT ON Angestellter TO Meier WITH GRANT OPTION
```