Klausur Datenbanken
Wintersemester 2016/2017
Prof. Dr. Wolfgang May
8. Februar 2017, 14-16 Uhr
Bearbeitungszeit: 90 Minuten

Vorname:

Nachname:

Matrikelnummer:

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)	16	13
Aufgabe 2 (Transformation in das Relationale Modell)	18	14
Aufgabe 3 (SQL und Relationale Algebra)	42	14
Aufgabe 4 (Verschiedenes)	14	9
Summe	90	50

Note:

Themenstellung: Formel-1

Alle Klausuraufgaben basieren auf einem gemeinsamen "Auftrag": In der Klausur soll eine Datenbank über die Formel-1 entworfen werden, die Daten seit deren Einführung im Jahr 1950 enthält:

1. Formel-1-*Teams* haben einen Namen, sind in einem Land zuhause (hier wird angenommen, dass sie nie in ein anderes Land wechseln), und außerdem ist gespeichert, wann sie gegründet wurden (was auch vor der Einführung der Formel-1 gewesen sein kann; einige davon sind nicht mehr aktiv, aber das spielt an dieser Stelle keine Rolle): *Ferrari* (seit 1929) und *Alfa Romeo* (seit 1913) sind Teams aus *Italien*. *Mercedes* (seit 1894) ist ein Team aus *Deutschland*. *Red Bull*[1] ist ein 2005 gegründetes Team aus *Österreich*. *McLaren* (gegründet 1965) ist ein Team aus *Großbritannien*.

2. Zu jedem Land ist außer dem Namen auch gespeichert, auf welchem (eindeutigen) Kontinent es liegt (Russland und die Türkei werden zu Europa gerechnet).

3. Zu jedem *Fahrer* ist der Name (es wird angenommen, dass es keine zwei Fahrer mit demselben Namen gibt), sein Geburtsdatum, und sein *Heimatland* gespeichert (auch wenn er schon lange in Monaco wohnt). *Sebastian Vettel* wurde am 3.7.1987 geboren, und ist aus *Deutschland*. *Nico Rosberg* wurde am 27.6.1985 geboren, und ist ebenfalls aus *Deutschland*. *Giuseppe Farina* wurde am 30.10.1906 geboren und war *Italiener*.

4. Zu jedem Fahrer ist gespeichert, bei welchen Teams er wann unter Vertrag stand bzw. steht. *Sebastian Vettel* stand vom 1.1.2007 bis 31.7.2007 bei *Sauber* unter Vertrag, dann vom 1.8.2007-31.12.2008 bei *Toro Rosso*[2], vom 1.1.2009-31.12.2014 bei *Red Bull*, und seit 1.1.2015 bei *Ferrari*. *Nico Rosberg* stand vom 1.1.2006 bis 31.12.2009 bei *Williams* unter Vertrag, und vom 1.1.2010 bis 31.12.2016 bei *Mercedes*. *Giuseppe Farina* stand vom 1.1.1950 bis 31.12.1951 bei *Alfa Romeo* unter Vertrag, und vom 1.1.1952 bis 31.12.1955 bei Ferrari.
   Es kann vorkommen, dass ein Fahrer erst bei einem Team T_1 unter Vertrag steht, dann bei einem oder mehreren anderen, und später nochmal bei T_1.

5. Eine Saison entspricht jeweils einem Kalenderjahr. In jeder Saison finden mehrere (5 bis 25) Rennen auf unterschiedlichen Rennstrecken statt; auf jeder Strecke maximal ein Rennen pro Saison:

6. Jede Strecke hat einen Namen (meistens der Name einer Stadt), ausserdem ist gespeichert, in welchem Land sie liegt. Der *Hockenheimring* und der *Nürburgring* liegen in *Deutschland*, *Monza* liegt in *Italien*, *Suzuka* liegt in *Japan*.

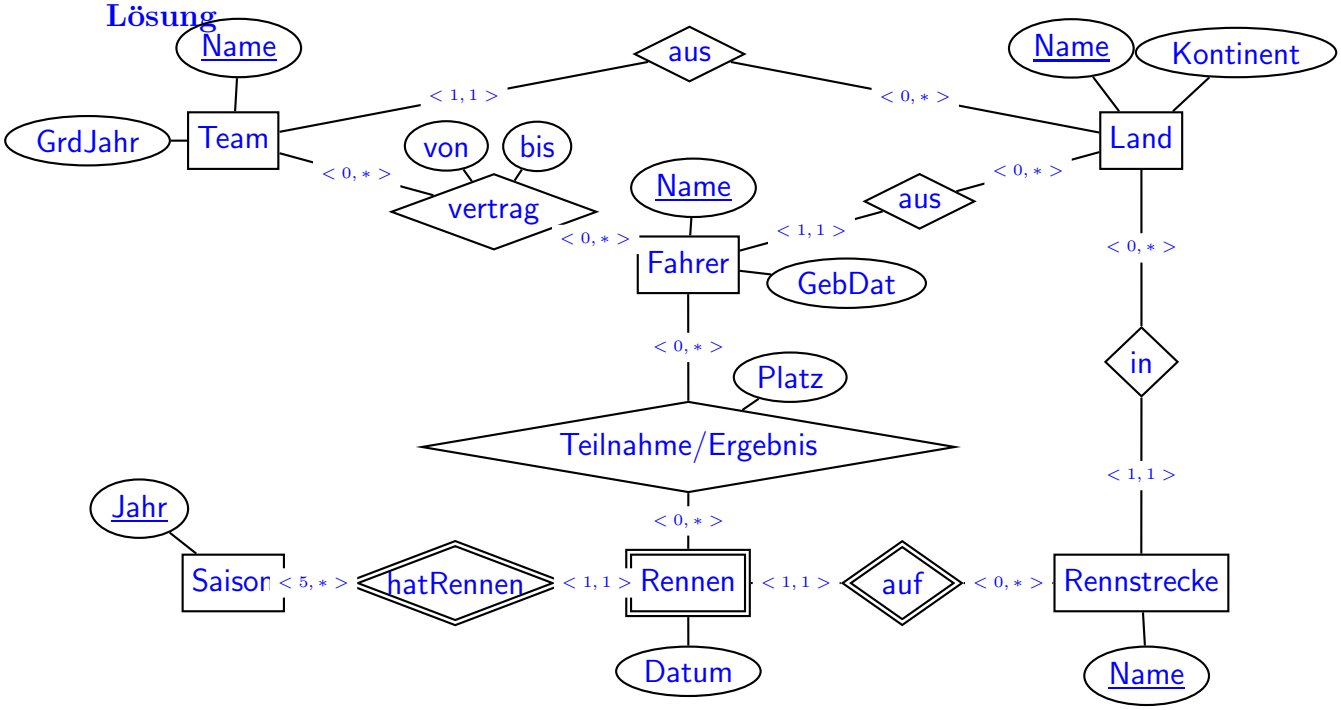
7. In der Saison 2016 fanden u.a. Rennen auf dem *Hockenheimring* am 31.7.2016, in *Monza* am 4.9.2016 und in *Suzuka* am 9.10.2016 statt. In der Saison 2015 fanden u.a. Rennen in *Monza* am 6.9.2015 und in *Suzuka* am 27.9.2015 (und kein Rennen auf dem *Hockenheimring*) statt.
   In der ersten Saison fand u.a. am 3.9.1950 ein Rennen in *Monza* statt.

8. Für jede Teilnahme eines Fahrers an einem Rennen wird gespeichert, welche Platzierung er erreicht hat, oder ob er ausgefallen ist.
   In der Saison 2016 hat *Nico Rosberg* das Rennen in *Monza* auf dem 1. Platz beendet; *Lewis Hamilton* wurde Zweiter, *Sebastian Vettel* wurde Dritter. In der Saison 2012 hat *Sebastian Vettel* das Rennen in *Suzuka* auf dem 1. Platz beendet, *Nico Rosberg* und *Fernando Alonso* sind in diesem Rennen ausgefallen. In der Saison 2008 hat *Sebastian Vettel* das Rennen in *Monza* auf dem 1. Platz beendet. Das Rennen in *Monza* in der Saison 1950 hat *Giuseppe Farina* gewonnen.

[1] engl. für "rotes männliches unkastriertes Rindvieh"
[2] ital. für "rotes männliches unkastriertes Rindvieh"

Aufgabe 1 (ER-Modell [16 Punkte])
Entwickeln Sie ein ER-Modell für das Szenario. Geben Sie darin die Schlüsselattribute sowie die Beziehungskardinalitäten an.

Lösung



- *Rennen* ist ein schwacher Entitätstyp, entsprechend "das F1-Rennen in Suzuka in der Saison 2016". Man könnte es auch als starken Entitätstyp mit *Datum* als alleinigem Schlüssel machen (und diesen entsprechend bei den Ergebnissen als Referenz verwenden).

  Das wäre aber unintuitiv. Es sieht es einfacher aus (Aufgabe 2a, einstelliger Schlüssel), andererseits muss man bei Anfragen (Aufgabe 3a+d, in denen man z.B. auf ein Ergebnis via Rennstrecke/Jahr zugreifen möchte, jedesmal joinen.

- Die Beziehung *Teilnahme/Ergebnis* kann auch als (schwacher) Entitätstyp mit den identifizierenden Beziehungen zu *Fahrer* und *Rennen* modelliert werden.

- Die Beziehung *vertrag* (hatteVertrag) kann auch als (schwacher) Entitätstyp zwischen *Fahrer* und *Team* modelliert werden. Dann ist die Beziehung zu *Fahrer* identifizierend, und *von* der zusätzliche lokale Key (womit in Aufgabe 2 dieselbe Tabelle erreicht wird).

- Der Entitätstyp *Saison* ist nicht notwendig, man kann *Saison* bzw. *Jahr* auch einfach als Schlüsselattribut zu *Rennen* machen (in Aufgabe 2 wird es das sowieso).

  Über die Redundanz der Jahreszahl als *Saison* und in *Datum* sollte man sich an dieser Stelle keine Gedanken machen (siehe auch Anmerkung bei Aufgabe 2).

  Wenn man allerdings irgendwie noch erfassen wollte, wer in einer Saison Weltmeister wurde, muß *Saison* ein Entitätstyp sein.

- Bei *Teilnahme/Ergebnis.Platz* bedeutet in dieser Modellierung NULL einen Ausfall.

  Man kann dagegen einwenden, dass nach dem Anlegen (zukünftiger) Resultate, dort auch erstmal eine Null steht. Dieser Einwand ist eher schwach, da man ja sowieso erst beim Eintragen der Ergebnisse nur diejenigen Paare (Fahrer,Rennen) erzeugt, die man haben möchte.

Als (eher weniger gute) Varianten kann man "0" als Ausfall vereinbaren, oder noch ein zusätzliches Attribut *ausgefallen* yes/no verwenden.

- Nebenbemerkung: Um die Wertung als Saison-Weltmeisterschaft in der Datenbank umzusetzen, müsste man die Punktewertung reinnehmen, die sich jedoch mehrmals verändert hat (1980er: 9-6-4-3-2-1, 1990er: 10-6-4-3-2-1, aktuell 25-18-15-12-10-8-6-4-2-1). Dies ginge noch relativ einfach mit einer separaten Tabelle; Noch komplizierter wird es bei den Regelungen zur Berücksichtigung von Streichresultaten, schnellsten Runden, besten Trainingsrunden, Fahrer-/Autowechseln (die es in der Anfangszeit gab) etc.

Aufgabe 2 (Transformation in das Relationale Modell [18 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, Diverses) 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.

Lösung

Land

Name	Kontinent
D	Europa
I	Europa
GB	Europa
R	Europa
J	Asien
:	:

Team

Name	Land	GrdJahr
Ferrari	I	1929
Alfa Romeo	I	1913
Mercedes	D	1894
Red Bull	A	2005
McLaren	GB	1965
:	:	:

Fahrer

Name	Land	GebDat
Sebastian Vettel	D	3.7.1987
Nico Rosberg	D	27.6.1985
Giuseppe Farina	I	30.10.1906
:	:	:

Vertrag

Fahrer	Team	von	bis
Sebastian Vettel	Sauber	1.1.2007	31.7.2007
Sebastian Vettel	Toro Rosso	1.8.2007	31.12.2008
Sebastian Vettel	Red Bull	1.1.2009	31.12.2014
Sebastian Vettel	Ferrari	1.1.2015	null
Nico Rosberg	Williams	1.1.2006	31.12.2009
Nico Rosberg	Mercedes	1.1.2010	31.12.2016
Nico Rosberg	Mercedes	1.1.2010	31.12.2016
:	:	:	:

Strecke

Name	Land
Hockenheimring	D
Nürburgring	D
Monza	I
Imola	I
Suzuka	J
:	:

Rennen

Saison	Strecke	Datum
2016	Hockenheimring	31.7.2016
2016	Monza	4.9.2016
2016	Suzuka	9.10.2016
2015	Suzuka	27.9.2015
2015	Monza	6.9.2015
1950	Monza	3.9.1950
:	:	:

Ergebnis

Fahrer	Strecke	Saison	Platz
Nico Rosberg	Monza	2016	1
Lewis Hamilton	Monza	2016	2
Sebastian Vettel	Monza	2016	3
Sebastian Vettel	Suzuka	2012	1
Nico Rosberg	Suzuka	2012	null
Fernando Alonso	Suzuka	2012	null
Giuseppe Farina	Monza	1950	1
:	:	:	:

- bei *Vertrag* sind nur *Fahrer* und *von* Schlüssel.
- Bei *Ergebnis* könnte man auf die Idee kommen, (*Strecke, Saison, Platz*) als Schlüssel zu wählen. Dann hat man jedoch ein Problem, wie man mit Ausfällen umgeht: *Platz*=NULL

darf nicht im Schlüssel vorkommen. Wenn man Ausfall als Platz=0 modelliert, ist der Schlüssel verletzt, wenn mehrere Fahrer im selben Rennen ausfallen.

b) Geben Sie die `CREATE TABLE`-Statements für diejenige Tabelle, in der die Rennergebnisse gespeichert sind, so vollständig wie möglich an (6 P).

Lösung

```
CREATE TABLE Ergebnis                          Basis 3P,
( Fahrer   VARCHAR2(50) REFERENCES Fahrer(Name),                    1/2P
  Strecke  VARCHAR2(40),
           --  REFERENCES Strecke(Name) ist im Rennen-FKey dabei
  Saison   NUMBER,
           -- optional: CHECK (Saison BETWEEN 1950 AND 2050),
           -- waere aber passender bei Rennen.Saison
  Platz    NUMBER CHECK Platz > 0 ,   -- NULL bedeutet Ausfall,     1/2P
  PRIMARY KEY (Fahrer, Strecke, Saison),                             1P
  FOREIGN KEY (Strecke, Saison) REFERENCES Rennen(Strecke, Saison))  1P
```

- Nebenbemerkung: Man könnte auf die Idee kommen, auch noch `UNIQUE (Strecke, Saison, Platz)` zu garantieren. Dies wäre jedoch problematisch, wenn Platz=NULL einen Ausfall beschreibt. NULL-Werte können zwar eigentlich keine Bedingungen (ausser NOT NULL) verletzen, trotzdem bekäme man zumindest bei Oracle eine Fehlermeldung, wenn zwei Fahrer im selben Rennen ausgefallen wären.
- Nebenbemerkung: Bei der Tabelle *Rennen* hat man ja ein bisschen Redundanz zwischen *Saison* und *Datum*. Man könnte überlegen, dass man dort als *Datum* ja nur Tag+Monat ablegen müsste. Zugriffe erfolgen jedoch üblicherweise über das volle Datum, was dann jedesmal als (Tag,Monat)+Saison umformuliert werden müßte. Außerdem umfasst der DATE-Datentyp bei Datenbanken natürlicherweise Tag+Monat+Jahr.

  In einer realen Anwendung sollte man dort eine Integritätsbedingung anlegen:

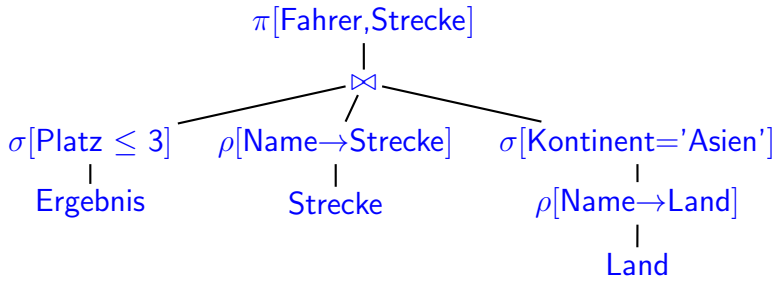
  ```
  CREATE TABLE RENNEN (
    Strecke ...,
    Saison NUMBER ...,
    Datum  DATE,
    ... primary key, foreign keys ...,
    CHECK (EXTRACT(YEAR from Datum) = Saison))
  ```

Aufgabe 3 (SQL und Relationale Algebra [42 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 oder -Baum an, die alle Paare (Fahrer, Rennstrecke) ausgeben, bei denen ein Fahrer bei einem Rennen in Asien einen der ersten 3 Plätze belegt hat. (2+2 P)

Lösung

```
SELECT DISTINCT fahrer, strecke  -- evtl. mehrmals auf dieser Strecke gewonn
FROM ergebnis, strecke, land
WHERE ergebnis.strecke = strecke.name
  AND strecke.land = land.name
  AND kontinent = 'Asien'
  AND platz <= 3
```



b) Geben Sie eine SQL-Anfrage an, die für jeden Fahrer, der *mindestens in zwei Saisons* ein Rennen gewonnen hat, dessen Namen und die Anzahl der in seiner gesamten Karriere gewonnenen Rennen ausgibt. (3 P)

Lösung

```
SELECT fahrer, count(*)
FROM ergebnis
WHERE platz = 1
GROUP BY fahrer
HAVING count(distinct saison) > 1
```

Wenn man auf die Idee mit dem HAVING nicht kommt, kann man es auch so machen:

```
SELECT fahrer, count(*)
FROM ergebnis e1
WHERE platz = 1
  AND EXISTS ( -- noch in einer anderen Saison
               SELECT *
               FROM ergebnis e2
               WHERE e2.fahrer = e1.fahrer
                 AND platz = 1
                 AND e2.saison != e1.saison )
GROUP BY fahrer
```

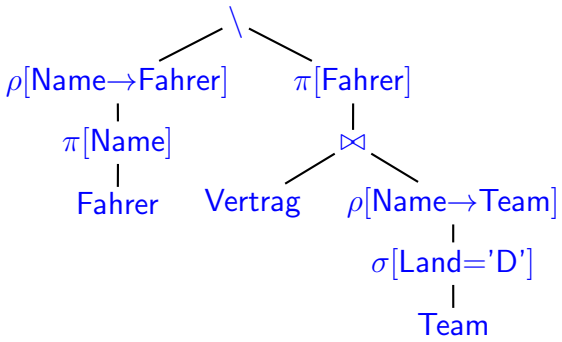
Wer eine andere, häufig Join-basierte, Umschreibung für das HAVING hat, sollte prüfen, ob das Ergebnis auch richtig ist, wenn ein Fahrer in mindestens *drei* verschiedenen Saisons gewonnen hat.

c) Geben Sie eine SQL-Anfrage **und** einen Algebra-Ausdruck oder -Baum an, die die Namen aller Fahrer ausgeben, die nie einen Vertrag bei einem deutschen Team hatten. (3+4 P)

Lösung

```
SELECT name
FROM fahrer
WHERE NOT EXISTS
 (SELECT *
  FROM vertrag v, team
  WHERE v.fahrer = fahrer.name
    AND v.team = team.name
    AND team.land = 'D')
```

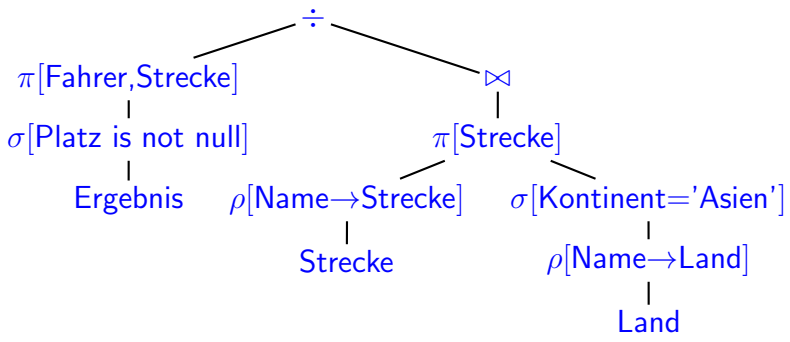
```
(SELECT name
 FROM fahrer)
MINUS
(SELECT fahrer
 FROM vertrag, team
 WHERE vertrag.team = team.name
   AND team.land = 'D')
```



d) Geben Sie eine SQL-Anfrage **und** einen Algebra-Ausdruck oder -Baum an, die die Namen aller Fahrer ausgeben, die auf allen in Asien gelegenen Rennstrecken schon mindestens einmal gefahren sind und dabei auch im Ziel angekommen sind. (5+5 P)

Lösung

```
SELECT name
FROM fahrer
WHERE NOT EXISTS
  (SELECT *
   FROM strecke, land
   WHERE strecke.land = land.name
     AND land.kontinent = 'Asien'
     AND NOT EXISTS
         (SELECT *
          FROM ergebnis
          WHERE ergebnis.fahrer = fahrer.name
            AND ergebnis.strecke = strecke.name
            AND platz IS NOT NULL))
```



Selten wird die Variante verwendet, die Division in SQL durch Zählen von Ergebnissen zu ersetzen. Dies ist riskant (vgl. eine Übungsaufgabe in manchen Jahren), weil man daran denken muss, auch aussen die Einschränkung auf "Asien" zu machen:

```
SELECT name
FROM fahrer
WHERE (SELECT COUNT(DISTINCT STRECKE)
       FROM ergebnis, strecke, land
       WHERE ergebnis.fahrer = fahrer.name
         AND platz IS NOT NULL
         AND ergebnis.strecke = strecke.name   -- muss in Asien sein!
         AND strecke.land = land.name
         AND land.kontinent = 'Asien')
     = -- Anzahl der Strecken in Asien
      (SELECT count(*)
       FROM strecke, land
       WHERE strecke.land = land.name
         AND land.kontinent = 'Asien')
```

Wenn man die äußere/erste Einschränkung nicht macht, und es (zur Zeit) 10 Strecken in Asien gibt, würde jeder Fahrer, der bisher auf genau 10 verschiedenen Strecken weltweit gefahren ist, im Ergebnis erscheinen.

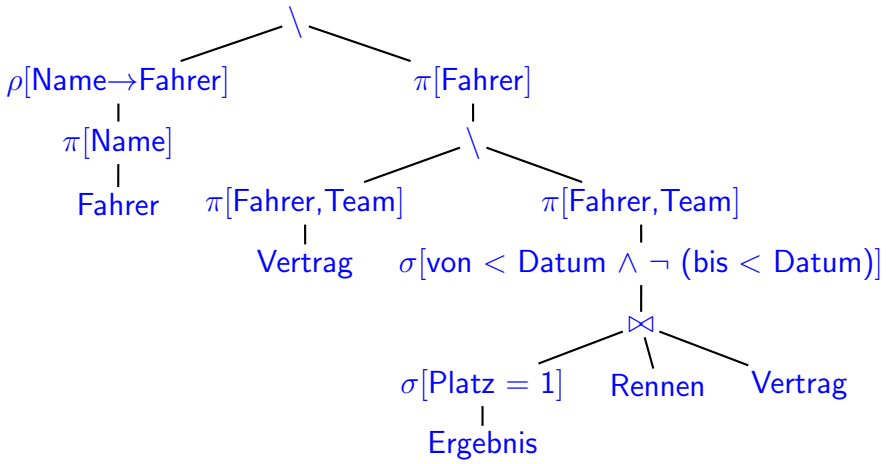
e) Geben Sie eine SQL-Anfrage **und** einen Algebra-Ausdruck oder -Baum an, die die Namen aller Fahrer ausgeben, die für jedes Team, bei dem sie einmal unter Vertrag standen, mindestens ein Rennen gewonnen haben. (6+6 P)

Lösung

```
SELECT name
FROM fahrer f
WHERE NOT EXISTS
   (SELECT team
    FROM vertrag v1
    WHERE f.name = v1.fahrer
      AND NOT EXISTS
      (SELECT *
```

```
        FROM ergebnis e, rennen, vertrag v2     -- mit demselben Team
        WHERE e.name = f.name
          AND e.platz = 1
          AND e.saison = rennen.saison AND e.strecke = rennen.strecke
          AND e.fahrer = v2.name
          AND v2.von < datum AND NOT v2.bis < datum  -- v2.bis kann null sein
                        -- AND (bis > datum OR bis IS NULL)  geht auch
          AND v2.team = v1.team))
-- Da ein Fahrer mehrmals bei demselben Team unter Vertrag gewesen sein kann,
-- ist eine Anfrage, in der "Vertrag" nur einmal vorkommt, nicht richtig.
```

Hinweis: das ist keine Division, da die zu testende Menge (der Teams) nicht fest ist, sondern von dem jeweiligen Fahrer abhängt. Also muss man es mit minus-minus machen: Alle Fahrer, ohne diejenigen, bei denen bei (meine-teams minus teams-mit-denen-ich-gewonnen-habe) etwas übrig bleibt:



f) Ein bisschen Theorie (6 P).

Seien $R(A, B)$, $S_1(B)$ und $S_2(B)$ Relationen.

Beweisen oder widerlegen Sie, ob die folgende Gleichung allgemeingültig ist:

$$(R \div S_1) \cap (R \div S_2) \; = R \div (S_1 \cup S_2)$$

Lösung Die Gleichheit gilt.

- Gesucht sind: diejenigen A, die mit allen B in S_1 in R vorkommen, und die ebenfalls mit allen B in S_2 in R vorkommt. Offensichtlich sind das genau die A, die mit jedem B in $S_1 \cup S_2$ in R vorkommen.
- Man kann es sich mit (d) überlegen: Sei $R(Fahrer, Strecke)$ die Menge aller Tupel, so dass ein Fahrer auf einer Strecke mindestens einmal angetreten ist, S_1 die Menge der asiatischen Rennstrecken, und S_2 die Menge der amerikanischen Rennstrecken.
  Dann ist die Schnittmenge der Fahrer, die auf allen asiatischen Strecken einmal

gefahren sind mit der Menge der Fahrer, die auf allen amerikanischen Strecken einmal gefahren sind gleich der Menge der Fahrer, die auf allen Strecken, die in Asien oder Amerika liegen, einmal gefahren sind.

– Formaler Beweis:

$$
\begin{aligned}
& (R \div S_1) \cap (R \div S_2) \\
= \;& \{\mu \in \mathsf{Tup}(\bar{Z}) \mid \{\mu\} \times S_1 \subseteq R\} \cap \{\mu \in \mathsf{Tup}(\bar{Z}) \mid \{\mu\} \times S_2 \subseteq R\} \\
= \;& \{\mu \in \mathsf{Tup}(\bar{Z}) \mid \{\mu\} \times S_1 \subseteq R \text{ and } \{\mu\} \times S_2 \subseteq R\} \\
= \;& \{\mu \in \mathsf{Tup}(\bar{Z}) \mid \{\mu\} \times (S_1 \cup S_2) \subseteq R\} \\
= \;& R \div (S_1 \cup S_2)
\end{aligned}
$$

bzw. wenn man verwendet, wie die Division durch die anderen relationalen Operatoren ausgedrückt wird:

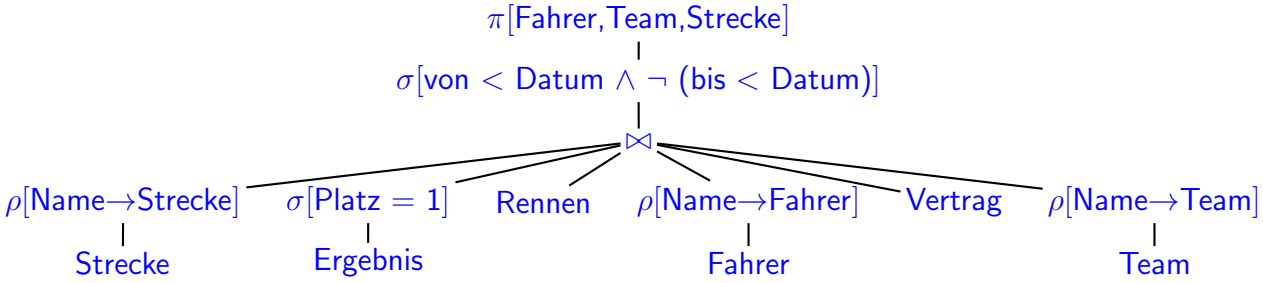
$$
\begin{aligned}
& (R \div S_1) \cap (R \div S_2) \\
= \;& \pi[\bar{A}](R) \setminus \pi[\bar{A}]((\pi[\bar{A}](R) \times S_1) \setminus R) \cap \pi[\bar{A}](R) \setminus \pi[\bar{A}]((\pi[\bar{A}](R) \times S_2) \setminus R) \\
= \;& \pi[\bar{A}](R) \setminus (\pi[\bar{A}]((\pi[\bar{A}](R) \times S_1) \setminus R) \cup \pi[\bar{A}]((\pi[\bar{A}](R) \times S_2) \setminus R)) \\
= \;& \pi[\bar{A}](R) \setminus (\pi[\bar{A}](((\pi[\bar{A}](R) \times S_1) \setminus R) \cup ((\pi[\bar{A}](R) \times S_2) \setminus R))) \\
= \;& \pi[\bar{A}](R) \setminus (\pi[\bar{A}](((\pi[\bar{A}](R) \times S_1) \cup (\pi[\bar{A}](R) \times S_2)) \setminus R)) \\
= \;& \pi[\bar{A}](R) \setminus (\pi[\bar{A}]((\pi[\bar{A}](R) \times (S_1 \cup S_2)) \setminus R)) \\
= \;& R \div (S_1 \cup S_2)
\end{aligned}
$$

Aufgabe 4 (Verschiedenes [14 Punkte])

a) Geben Sie eine SQL-Anfrage **und** einen Algebra-Ausdruck oder -Baum an, die alle Tupel (Fahrername, Teamname, Strecke) ausgeben, so dass ein Fahrer für ein Team seines Heimatlandes ein Rennen auf einer Strecke in seinem Heimatland gewonnen hat. (3+3P)

Lösung

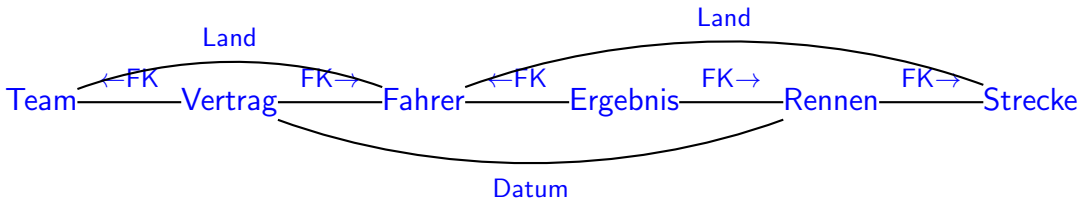
```
SELECT DISTINCT fahrer.name, team.name, strecke.name
FROM ergebnis e, rennen, strecke, fahrer, vertrag, team,
WHERE e.fahrer = fahrer.name AND
  AND e.saison = rennen.saison and e.strecke = rennen.strecke
  AND rennen.strecke = strecke.name
  AND e.fahrer = vertrag.name
  AND (vertrag.von < rennen.datum
       AND (vertrag.bis > rennen.datum OR vertrag.bis IS NULL))
  -- kuerzer: AND (vertrag.von < rennen.datum AND NOT vertrag.bis < rennen.datum)
  AND and vertrag.team = team.name
  AND fahrer.land = team.land AND fahrer.land = strecke.land
  AND platz = 1
```



- SQL: DISTINCT ist notwendig, da ein Fahrer ja mehrmals auf derselben Strecke für dasselbe Team gewonnen haben könnte
- Algebra: alle weiteren "="-Bedingungen, d.h., FK-PK-Joins, sowie die Gleichheit der jeweiligen Länder, stecken im Natural Join

b) Begründen Sie, warum die obige Anfrage als ein "zyklisches Join" bezeichnet wird. (2P)

Lösung Die Anfrage besteht im Prinzip nur aus einem breiten Join mit einfachen Selektionen und Projektionen.
Der Graph, der enststeht, wenn man aufzeichnet, zwischen welchen Tabellen Vergleiche stattfinden, ist zyklisch:



c) Skizzieren (und begründen) Sie, wie die interne Auswertung obiger Anfrage am effizientesten durchgeführt werden könnte. (6 P)
(Die Datenbank enthält ca. 760 Fahrer aus 100 Ländern, 70 Strecken in 25 Ländern, 300 Teams aus 20 Ländern; 70 Saisons mit jeweils 5-25 Rennen; insgesamt ca. 1000 Rennen mit je ca. 20 Teilnehmern, und damit 20000 Ergebnis-Einträgen, davon 1000 Einträge für einen 1. Platz.)

Lösung Wichtig: Klein anfangen, restriktiv joinen:

- "naiv" auf nur-Join-Basis unter Betrachtung von Indexen: 20 Strecken, Teams über "Land" dranjoinen. Sind ca. 200 Tupel (in vielen Ländern gibt es zwar Strecken, aber keine Teams, dafür in D/I/GB mehrere Strecken und Teams), Fahrer über deren Heimatland dranjoinen (dabei fallen einige Fahrer, aber nur wenige Strecken/Teams aus deren Ländern es keine Fahrer gibt) komplett weg; übrig bleiben Kandidaten wie (N.Rosberg, Mercedes, Hockenheim), (N.Rosberg, Mercedes, Nürburgring), (G.Farina, Toro Rosso, Monza) ca. 400 "landesgleiche" Tripel.
  - Restproblem: hat der Fahrer mal auf dieser Strecke mit diesem Team gewonnen?
  - Jetzt könnte man mit *Ergebnis* joinen (ob der Fahrer jemals auf der Strecke gewonnen hat (gefahren ist er dort evtl. mehrmals), und wenn ja in welchen Saisons) [Ergebnis.Fahrer-Index (durchschnittlich 40 Einträge) benutzen, der ist restriktiver als Ergebnis.Strecke (durchschn. 400 Einträge)], um dann per (Semi)join zu überprüfen, ob er in dieser Saison bei dem gerade getesteten Team gefahren ist.
  - Oder zuerst schauen, ob (und in welchen Saisons) der Fahrer überhaupt für das Team gefahren ist – das reduziert die Menge der möglichen Ergebnisse stark (Sebastian Vettel ist z.B. bisher nie für ein deutsches Team gefahren, aber jedes Jahr einmal auf einer deutschen Strecke).
    Und dann schauen, ob er auf dieser Strecke in einer dieser Saisons gewonnen hat.
- Indexbasiert, d.h. im wesentlichen ohne Zugriff auf die Tupel-Seiten:
  Man iteriert sozusagen über die Länder: Tripel-Bildung der (Foreign-Key)-Indexe aus Fahrer.Land, Team.Land, Strecke.Land gibt die möglichen ca. 400 Kandidaten (Fahrer, Team, Strecke) als Tupelreferenzen von oben. Weiter wie oben:
  - Dann mit den FK-Indexen (Ergebnis.Fahrer, Ergebnis.Strecke, Ergebnis.Platz=1) schneiden. Aus diesen Tupeln die Saison holen, und in Vertrag nachschauen, oder
  - mit den FK-Indexen (Vertrag.Fahrer, Vertrag.Team) schneiden. Aus diesen Tupeln die Saison holen, und in Ergebnis nachschauen.
- Naiver "Star-Style" Join: anfangen mit Ergebnissen (20000), bzw. wenn wenigstens ein Index auf den Platzierungen ist, mit den 1000 Erstplatzierungen. Fahrer$\rightarrow$ Land = Strecke $\rightarrow$ Land sind dann schnelle indexbasierte Lookups (die nicht wirklich die Landesnamen, sondern die Landes-Tupelreferenzen vergleichen), und reduzieren die Menge der möglichen Ergebnisse deutlich.
  Dann muss noch über Ergebnis.Saison in Vertrag nachgeschaut werden, ob der Fahrer in dieser Saison bei dem gesuchten Team war.