SQL-Übungsskript

zur Vorlesung

„Informationssysteme“

Technische Universität Dortmund

Fakultät für Informatik

Version vom Sommersemester 2020

Vorwort

Liebe Teilnehmerinnen und Teilnehmer der Veranstaltung Informationssysteme,

das vorliegende Übungsskript dient dazu, den praktischen Umgang mit dem objektrelationalen Datenbanksystem Oracle zu erlernen und zu üben. Auf den folgenden Seiten findet sich eine Fülle von Aufgaben zu den Kapiteln der Informationssysteme-Vorlesung.

Auf den Hausübungsblättern zur Vorlesung werden wir regelmäßig Aufgaben aus diesem Skript stellen. Es werden aber nicht alle Aufgaben aus dem Skript gestellt und besprochen werden. Daher bietet Euch dieses Skript durch die große Aufgabenauswahl die Möglichkeit, jedes Thema auch individuell zu üben. Es steht Euch frei, Aufgaben aus späteren Kapiteln vorzuziehen. Das Übungsskript wird damit auch unterschiedlichen Lerntempos gerecht. Da nicht alle praktischen Aufgaben in den Übungsgruppen besprochen werden können, könnt Ihr in der parallel zu Vorlesung und Übung stattfindenden (freiwilligen) Lernraumbetreuung im Rechnerpool (vgl. Homepage der VL) Fragen zu Euren Lösungsansätzen und zu Problemen mit den Übungsleitern besprechen.

Rückmeldungen zu diesem Skript sind gerne gesehen, so dass wir wir missverständliche oder fehlerhafte Aufgabenstellungen schnell korrigieren können.

Marcel Preuß,
Thomas Lindemann,
Jens Teubner
März 2020

Inhaltsverzeichnis

1 Tabellen anlegen

Bemerkung: Das Konzept von funktionalen Abhängigkeiten wird im Kapitel Schema Normalization der Vorlesung diskutiert.

Aufgabe 1.1

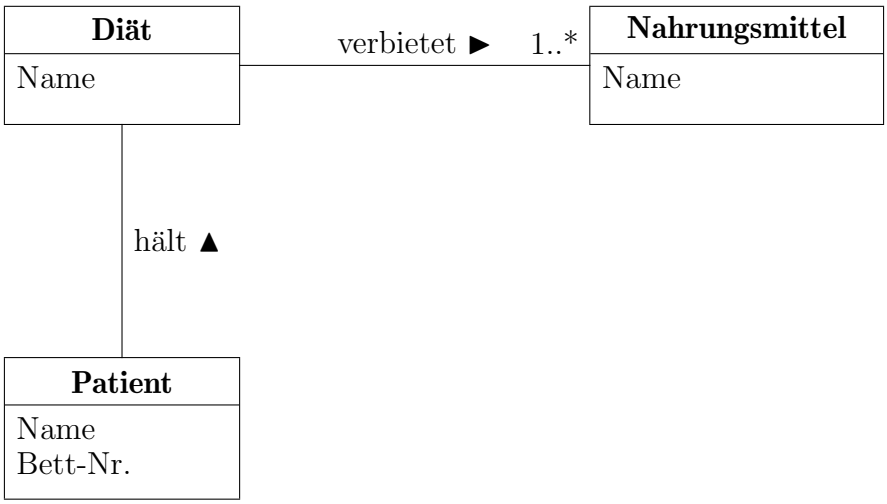
Gegeben sei das folgende relationale Datenbankschema (Schlüsselattribute sind unterstrichen):

Student	(Name, MatrNr, Semester, Studiengang))
Veranstaltung	(Nummer, Name, Studiengang, Ort)
Anmeldung	(Kursnummer, MatrNr)

- Setzt das vollständige Schema in `CREATE-TABLE`-Befehle um. Überlegt euch für die Attribute geeignete SQL-Typen. Welche semantischen Bedingungen und funkionalen Abhängigkeiten sollten zwischen den Attributen gelten?

Aufgabe 1.2

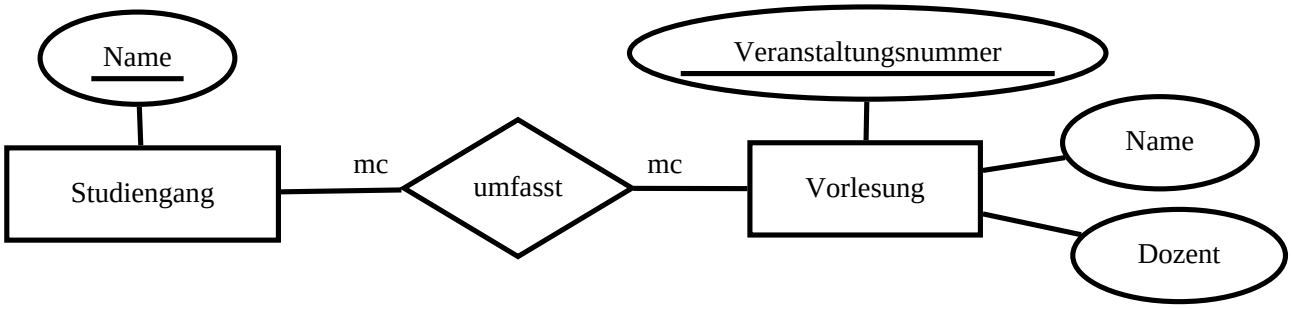
Gegeben sei folgendes UML-Klassendiagramm



Drückt das UML-Diagramm als ER-Diagramm aus.
Implementiert das Schema durch SQL-Tabellen.
Was sind dafür geeignete `CREATE-TABLE`-Befehle?

Aufgabe 1.3

Gebt zu folgendem ER-Modell ein passendes relationales Datenbankschema an und setzt dieses in „`CREATE TABLE`“-Befehle um:



Hinweis: Die Kardinalitäten sind in der modifizierten Chen-Notation vermerkt:

`1` = „genau 1“, `c` = „0 oder 1“, `m` = „mind. 1“, `mc` = „beliebig viele“.

Ihr solltet übrigens beim Anlegen der Tabellen statt „Umfasst“ lieber „Enthaelt“ als Tabellenname wählen, da „Umfasst“ schon in der Lebensmitteldatenbank vergeben ist und das zu Problemen führen kann.

2 Anfragen stellen

Die Präsidentendatenbank zur Vorlesung enthält Informationen zu den amerikanischen Präsidenten, zum Beispiel Infos über deren Hobbies und Familie oder zu Amtszeiten der Präsidenten. Eine Übersicht über die Tabellen und mit welchen Befehlen diese angelegt wurden, findet sich im Foliensatz der Vorlesung ab Seite 124ff.

2.1 Anfragen auswerten

Aufgabe 2.1

Welche Ausgabe erzeugen die folgenden SQL-Anfragen an die Präsidentendatenbank?

1. ```
   SELECT PRES_NAME,BIRTH_YEAR FROM PRESIDENT;
   ```
2. ```
   SELECT DISTINCT PARTY FROM PRESIDENT;
   ```
3. ```
   SELECT HOBBY FROM PRES_HOBBY WHERE PRES_NAME='Roosevelt T';
   ```
4. ```
   SELECT P.PRES_NAME, P.BIRTH_YEAR, H.HOBBY
   FROM PRESIDENT P, PRES_HOBBY H
   WHERE P.PRES_NAME=H.PRES_NAME;
   ```
5. ```
   SELECT COUNT(*) FROM PRESIDENT
   WHERE DEATH_AGE IS NOT NULL;
   ```
6. ```
   SELECT COUNT(DEATH_AGE) FROM PRESIDENT;
   ```
7. ```
   SELECT PARTY, COUNT(*) AS Anzahl FROM PRESIDENT
   GROUP BY PARTY
   ORDER BY Anzahl DESC;
   ```
8. ```
   SELECT PARTY, COUNT(DISTINCT STATE_BORN)
   FROM PRESIDENT
   GROUP BY PARTY;
   ```
9. ```
   SELECT PRES_NAME, SUM(NR_CHILDREN) FROM PRES_MARRIAGE
   GROUP BY PRES_NAME HAVING SUM(NR_CHILDREN) >= 10;
   ```
10. ```
    SELECT P.PRES_NAME FROM PRESIDENT P
    WHERE NOT EXISTS
      (SELECT PRES_NAME FROM PRES_MARRIAGE M
       WHERE P.PRES_NAME=M.PRES_NAME);
    ```
11. ```
    SELECT PRES_NAME FROM PRESIDENT
    MINUS
    SELECT PRES_NAME FROM PRES_MARRIAGE;
    ```
    ```

2.2 Präsidentendatenbank

Aufgabe 2.2

Gebt zur Präsidentendatenbank SQL-Anfragen an, die die folgenden Fragen beantworten.

1. Welche Präsidenten wurden 1946 geboren?
2. Welche Präsidenten leben noch?
3. Gebt alle in der Datenbank verfügbaren Informationen über „Obama B H“ aus. Es ist sinnvoll, hierfür mehrere einzelne Anfragen zu verwenden.
4. Wie lauten die Namen, Geburtsjahre und Geburtsstaaten aller Präsidenten? Die Ausgabe soll alphabetisch nach dem Geburtsstaat und innerhalb eines Staates absteigend nach dem Geburtsjahr geordnet werden.
5. Welche Präsidenten starben in einem Alter von 50 bis 60 Jahren? Es sollen der Name und das Todesalter ausgegeben werden.
6. Welche Präsidenten waren auch einmal Vizepräsident? (Achtung: Nicht alle Präsidenten hatten auch einen Vizepräsidenten!)
7. Welche Präsidenten waren genau 1, 2 oder 12 Jahre im Amt? Es sollen die Namen der Präsidenten und ihre Amtszeiten sortiert nach Amtszeit und Präsidentennamen ausgegeben werden.
8. Gesucht werden die Daten der republikanischen Präsidenten, die zwischen 1910 und 1960 geboren wurden, sortiert nach dem Geburtsjahr. Es sollen solche Daten ausgegeben werden, die in der Tabelle President gespeichert sind.
9. Welcher Präsident hat mit wie vielen Vizepräsidenten zusammengearbeitet?
   Es soll jeweils der Name des Präsidenten und die Anzahl der Vizepräsidenten ausgegeben werden.
10. Bestimmt für jeden (in der Relation `PRESIDENT` erwähnten) Bundesstaat das minimale, maximale, das durchschnittliche Todesalter und das durchschnittliche Todesjahr der dort geborenen Präsidenten.
11. Welche Präsidenten traten ihr Amt mit mehr als 60 Jahren an? Gebt den Name und das Alter bei Amtsantritt an.
12. Was sind die Daten der Präsidenten, bei denen ein Vorname mit einem „R“ beginnt?
13. Was sind die Daten der Präsidenten, die als zweiten Buchstaben ihres Namens ein „e“ haben, deren Name aber nicht mit „R“ beginnt?
14. Welche Präsidenten haben genau einen Vornamen?
15. Welche Präsidenten waren nicht verheiratet? Es sollen nur die Namen der Präsidenten ausgegeben werden.

16. Welche republikanischen („Republican“) Präsidenten heirateten im Zeitraum von 1939 bis 1945?
    Es soll jeweils der Name des Präsidenten und das Jahr der Hochzeit ausgegeben werden.

17. Wie heißen die Ehefrauen derjenigen Präsidenten, deren Hobby Poker ist?

18. Zu welchen Parteien gehören die Präsidenten, die kinderlose Ehen hatten?
    Es soll jeweils der Name des Präsidenten und seine Partei ausgegeben werden.

19. Wie oft waren die Präsidenten verheiratet?
    Es soll jeweils der Name des Präsidenten und die Anzahl der Ehen ausgegeben werden.
    Die Ausgabe soll absteigend nach der Anzahl der Ehen sortiert werden.
    (Berücksichtigt dabei, dass nicht alle Präsidenten verheiratet waren.)

20. Wie heißen die Ehefrauen derjenigen Präsidenten, die 1924 geboren wurden?
    Es sollen nur die Namen der Ehefrauen ausgegeben werden.
    Die Ausgabe soll alphabetisch sortiert werden.

21. Welche Präsidenten waren bei ihrer Hochzeit genau 20 Jahre alt?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

22. Wie heißen die Ehefrauen der Präsidenten, die der demokratischen Partei („Democratic“) angehören?
    Es sollen nur die Namen der Ehefrauen ausgegeben werden.
    Die Ausgabe soll alphabetisch sortiert werden.

23. Welche Präsidenten gehören der republikanischen Partei („Republican“) an und sind *nicht* verheiratet?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

24. Welcher Präsident hat wie viele Kinder?
    Es soll jeweils der Name des Präsidenten und die Anzahl der Kinder (aus allen Ehen zusammen!) ausgegeben werden.
    (Präsidenten, die nicht verheiratet sind, brauchen *nicht* ausgegeben werden.)

25. Welche Präsidenten hatten in Summe (also mit allen Ehefrauen zusammen) 10 oder mehr Kinder? Es sollen die Namen der Präsidenten und die jeweilige Kinderanzahl gruppiert nach den Präsidentennamen ausgegeben werden.

26. Welche Präsidenten sind oder waren jünger als ihre Ehefrauen? Es soll der Name des Präsidenten und der Altersunterschied ausgegeben werden.

27. In welchen Jahren haben *genau* zwei Präsidenten geheiratet?

28. Welche Präsidenten haben als Hobby „Fishing“?

29. Welche Präsidenten haben *keine* Hobbys?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

30. Welche Präsidenten haben Golf und Reiten („Riding“) als Hobby?

31. Welche Hobbies haben diejenigen Präsidenten, die in Texas geboren wurden?
    Es sollen nur die Namen der Präsidenten und die Hobbies ausgegeben werden.

32. Von welchen Präsidenten sind mehr als drei Hobbys bekannt?

33. Welche Hobbies haben diejenigen Präsidenten, die der republikanischen Partei („Republican“) angehören?
    Die Hobbies sollen alphabetisch sortiert werden.
    Jedes Hobby soll nur einmal in der Ausgabe erscheinen.

34. Welchen Parteien gehören diejenigen Präsidenten an, deren Hobby Schießen („Shooting“) ist?
    Es sollen nur die Parteien ausgegeben werden, jede Partei soll höchstens einmal in der Liste erscheinen.

35. Welche Präsidenten sind Demokraten („Democratic“) und haben als Hobby „Riding“?

36. Welche Präsidenten haben Poker aber nicht Golf als Hobby?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

37. Welche Präsidenten gehören der republikanischen Partei („Republican“) an und haben *nicht* Schießen („Shooting“) als Hobby?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

38. Welche Präsidenten sind Republikaner („Republican“)?

39. Welchen *verschiedenen* Parteien gehören die Präsidenten an?

40. Welche Parteien stellten mehr als acht Präsidenten, die nach 1850 geboren sind?
    Gebt auch pro Partei die Anzahl dieser Präsidenten mit aus.

41. Welche Präsidenten wurden in Texas geboren und waren *keine* Republikaner („Republican“)?
    Es sollen nur die Namen und die Parteien der Präsidenten ausgegeben werden.

42. Wieviele Präsidenten haben die verschiedenen Parteien gestellt?
    Es soll der Name der Parteien und die zugehörigen Präsidentenanzahlen nach den Anzahlen sortiert ausgegeben werden.
    Die Resultatspalte mit den Präsdidentenanzahlen soll mit „Anzahl“ bezeichnet werden.

43. Welche Präsidenten waren in der Whig-Partei („Whig“)?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

44. Berechnet für jede Partei die Gesamtzahl der Regierungsjahre der Präsidenten dieser Partei, die Anzahl der Präsidenten und die jeweilige durchschnittlichen Anzahl der Dienstjahre.
    Gebt alle diese Daten in einer Tabelle aus.

45. In welchen Staaten wurden keine Präsidenten geboren?
    Sortiert alphabetisch nach den Namen der Staaten.

46. Welche Präsidenten wurden in einem der Gründungsstaaten der USA geboren? (Die USA wurden 1776 gegründet.)
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

47. Welche Präsidenten wurden wann im Bundesstaat Texas geboren?
    Es sollen die Namen und die Geburtsjahre der Präsidenten ausgegeben werden.

48. In welchen Staaten wurden Präsidenten geboren?
    Es sollen die Namen der Staaten ausgegeben werden. Jeder Staat soll höchstens einmal in der Liste erscheinen.

49. In welchen Staaten wurden mindestens zwei Präsidenten geboren?
    Es soll jeweils der Name des Staates ausgegeben werden.
    Die Ausgabe soll aufsteigend nach der Anzahl der Präsidenten sortiert werden.

50. Aus wie vielen unterschiedlichen Staaten kamen die Präsidenten, die von den verschiedenen Parteien gestellt wurden?
    Es soll jeweils der Name der Partei und die Anzahl der Staaten ausgegeben werden.

51. Für welche US-Bundesstaaten gibts es keinen anderen Staat, der in einem späteren Jahr den USA beigetreten ist?
    Von diesen Staaten soll nur der Name ausgegeben werden.

52. Welche Präsidenten haben nie eine Präsidentschaftswahl gewonnen?

53. Welche Präsidenten haben mehrfach kandidiert? Es soll jeweils der Name ausgegeben werden.

54. Wie oft sind die Präsidenten jeweils zur Wahl angetreten? Wie oft haben sie eine Wahl gewonnen bzw. verloren?
    Es soll für jeden Präsidenten (auch für solche, die nie zur Wahl standen) der Name, sowie die Anzahl der Wahlteilnahmen, der Siege und der Niederlagen ausgegeben werden. Die Ausgabe soll absteigend nach Wahlsiegen und (nachrangig) nach Wahlteilnahmen sortiert sein.

55. Welche Präsidenten haben erst eine Wahl gewonnen und danach eine Wahl verloren?
    Es sollen jeweils der Name des Präsidenten, das Jahr der verlorenen Wahl und der Name des Gegenkandidaten ausgegeben werden, der diese Wahl gewonnen hat. Hinweis: Die Spalte `WINNER_LOSER_INDIC` der Tabelle `ELECTION` gibt zu jedem Kandidaten an, ob er die Wahl gewonnen („W“) oder verloren hat („L“).

56. In welchem Wahljahr hat der Gewinner der Wahl wie viele Stimmen bekommen?
    Es soll jeweils der Name und die Stimmenzahl ausgegeben werden.

57. Welche Präsidenten waren bei einer ihrer Hochzeiten jünger als 25 Jahre *oder* gehören der republikanischen Partei („Republican“) an?
    Es sollen nur die Namen der Präsidenten ausgegeben werden.

58. In welchen Jahren wurden *mindestens* zwei Präsidenten geboren?

59. Welche Präsidenten sind Republikaner und in Texas geboren?
(Gebt bitte zwei verschiedene Varianten an.)

60. Welche Präsidenten sind Republikaner oder in Texas geboren?
(Gebt bitte zwei verschiedene Varianten an.)

2.3 Stardatenbank

Aufgabe 2.3

Gegeben seien die folgenden Relationen (ohne semantische Bereichsnamen):

Star	Name	GebJahr	Geschlecht
	Michael Jackson	1958	m
	Danny DeVito	1944	m
	Helen Hunt	1963	w
	Elizabeth Taylor	1932	w

Episode	Code	Epi-Nr	Staffel-Nr	Titel
	7F24	1	3	Stark Raving Dad
	7F16	7	2	Oh Brother, Where Art Thou?
	8F23	12	3	Brother, Can You Spare Two Dimes?
	5F12	10	9	Dumbbell Indemnity
	9F19	13	4	Krusty Gets Kancelled
	9F08	7	4	Lisa's First Word

Gastauftritt	Name	Code
	Michael Jackson	7F24
	Danny DeVito	7F16
	Danny DeVito	8F23
	Helen Hunt	5F12
	Elizabeth Taylor	9F19
	Elizabeth Taylor	9F08

1. Gebt diese drei Tabellen ins Oracle–System ein.

2. Gebt nun SQL-Anfragen an, die die folgenden Fragen beantworten.

   (a) Welche Stars sind älter als 60 Jahre?
   Es sollen die Namen der Stars ausgegeben werden.

   (b) In welchen Episoden hatte Danny DeVito einen Gastauftritt?
   Es sollen die Titel der Episoden ausgegeben werden.

   (c) Welche Stars hatten in der dritten Staffel einen Gastauftritt?
   Es sollen die Namen der Stars ausgegeben werden.

(d) In welchen Episode hatten weibliche Stars einen Gastauftritt?
Es sollen die Episodennummern ausgegeben werden.

(e) Wie lauten die Namen der weiblichen Stars?

(f) In welcher Staffel hatte welcher Star einen Gastauftritt?
Es sollen die Namen der Stars und die Staffelnummern ausgegeben werden.

(g) In welchen Episoden hatte Elizabeth Taylor einen Gastauftritt?
Es sollen die Episodennummern ausgegeben werden.

(h) In welchen Episoden hatten Stars einen Gastauftritt, die jünger als 50 Jahre sind?
Es sollen die Titel der Episoden ausgegeben werden.

2.4 Hierarchische Anfragen

In der Datenbank ist in der Tabelle `umfasst` eine Lebensmittelhierarchie abgespeichert. Die Tabelle hat die Spalten `Gruppe` und `Nahrungsmittel`. Außerdem gibt es die Tabelle `Konkretes`, die die Spalten `Name` und `Brennwert` besitzt.

Aufgabe 2.4

Gebt zu den folgenden Fragen SQL-Anfragen an.

1. Welche Obstsorten sind in der Datenbank abgespeichert?

2. Welche Sorten Beerenobst haben einen Brennwert von mehr/weniger als 50 (kcal/100 g)?

Hinweis: Weiterführende Informationen zu hierarchischen Anfragen finden in der Oracle11g-Dokumentation. sich ab Seite 8-3 von „Oracle9i SQL Reference“ („Hierarchical Queries“).

Aufgabe 2.5

Gebt zu den folgenden Fragen bzgl. der Lebensmittelhierarchie SQL-Anfragen an.

1. Welche Brennwerte haben die pflanzlichen Nahrungsmittel? Es soll jeweils der Name des Nahrungsmittels und der Brennwert ausgegeben werden.

2. Welche tierischen Nahrungsmittel haben einen höheren Brennwert als Hirse? Es soll jeweils der Name des Nahrungsmittels und der Brennwert ausgegeben werden.

Aufgabe 2.6

Gebt zu den folgenden Fragen SQL-Anfragen an.

1. Welche konkreten Nahrungsmittel gehören zu welchen Nahrungsmittelgruppen?

2. Welche konkreten Obstsorten sind in der Datenbank gespeichert?
   Es sollen nur konkrete Obstsorten, z. B. Brombeere, nicht Klassen, wie Beerenobst, ausgegeben werden.

Aufgabe 2.7

Gegeben sei die folgende Tabelle, die einen gerichteten, markierten Graphen darstellt:

G	X	Y	L
	1	2	A
	1	3	B
	2	3	B
	3	4	A
	5	6	A

1. Was liefern die folgenden Anfragen?

```
select x, y, level
from g
start with x = 1
connect by x = prior y;
```

```
select x, y
from g
where l = 'A'
start with x = 1
connect by x = prior y;
```

2.5 Unteranfragen

Aufgabe 2.8

Was berechnen die folgenden SQL-Anfragen zur Präsidentendatenbank?
Hätte man die Anfragen auch anders formulieren können?

1. ```
   SELECT PRES_NAME
   FROM PRESIDENT
   WHERE PRES_NAME NOT IN
     (SELECT PRES_NAME
      FROM PRES_HOBBY
      WHERE HOBBY='Shooting');
   ```

2. ```
   SELECT S.STATE_NAME, S.YEAR_ENTERED
   FROM STATE S
   WHERE NOT EXISTS
     (SELECT * FROM PRESIDENT
      WHERE STATE_BORN=S.STATE_NAME);
   ```

3. ```
   SELECT DISTINCT STATE_NAME
   FROM STATE S1
   WHERE NOT EXISTS
     (SELECT * FROM STATE S2
      WHERE S1.YEAR_ENTERED < S2.YEAR_ENTERED);
   ```

4. ```
   SELECT PRES_NAME ||
   ' heiratete im Jahr '
   || TO_CHAR(MAR_YEAR)
       "Nachricht"
   FROM PRES_MARRIAGE
   WHERE PRES_NAME BETWEEN 'R' and 'Rf';
   ```

Aufgabe 2.9

Gebt zur Präsidentendatenbank SQL-Anfragen an, die die folgenden Fragen beantworten. Die Anfragen sollen mit `EXISTS` oder `IN` eingebundene Unteranfragen enthalten.

1. Welche Präsidenten wurden nie zum Präsidenten gewählt?
   Es soll jeweils der Name des Präsidenten ausgegeben werden.

2. Wann kam zum ersten Mal ein Präsident an die Macht, der in New York geboren wurde?
   Es soll das Jahr der Amtseinführung und der Name des Präsidenten ausgegeben werden.
   Andere Präsidenten sollen nicht in der Ausgabe enthalten sein.

Aufgabe 2.10

Gebt SQL-Anfragen an, die die folgenden Fragen beantworten. Gebt jeweils eine SQL-Anfrage an, die Unteranfragen verwendet, und eine, die keine Unteranfragen benutzt.

1. Welche Präsidenten waren nicht verheiratet?
2. Welche Präsidenten wurden in Texas geboren und haben Baseball als Hobby?

Aufgabe 2.11

Formuliert folgende Fragen zur Präsidentendatenbank als SQL-Anfragen. Benutzt dabei Unteranfragen mit den Operatoren `EXISTS` und `IN`.

1. In welchen Staaten wurden *keine* Präsidenten geboren?
2. Welche Präsidenten haben *keine* Hobbys?

Lassen sich die Anfragen auch so umformulieren, dass sie ohne Unteranfragen auskommen?

2.6 Komplexe Anfragen

Aufgabe 2.12

Um einem drohenden Rückgang der Bevölkerungszahl entgegenzuwirken, will die amerikanische Regierung eine Kampagne[1] ins Leben rufen, die dafür werben soll, mehr Kinder zu bekommen. Auf der Suche nach einem Namen für die Kampagne hat man sich auf die folgenden Regeln geeinigt:

- Die Kampagne soll nach einem amerikanischen Präsidenten benannt werden.
- Namensgeber soll derjenige Präsident werden, der die meisten Kinder hatte.
- Um allen Präsidenten die gleichen Chancen zu geben, soll die Anzahl der Kinder jeweils durch das Alter des Präsidenten bei seinem Tod geteilt werden.
- Es kommen deshalb nur Präsidenten in Frage, die schon gestorben sind.
- Weiterhin sollen nur Präsidenten berücksichtigt werden, die nicht mehrfach verheiratet waren.

Entwickelt eine (möglicherweise genestete) SQL-Anfrage, mit deren Hilfe sich der Namensgeber für die Kampagne bestimmen lässt. Es sollen die in Frage kommenden Präsidentennamen nebst Kinderanzahl/Todesalter-Quotient ausgegeben werden und die Ausgabe soll nach dem Quotienten sortiert werden.

Aufgabe 2.13

Welche Präsidenten wurden in einem Jahr geboren, in dem ein anderer Präsident ins Amt eingeführt wurde?
Es sollen die Namen der Präsidenten, die in dem Jahr geboren wurden, die Namen der Präsidenten, die ins Amt eingeführt wurden, und die Jahre angegeben werden.

[1] Diese Kampagne ist nur für diese Aufgabe ausgedacht und existiert nicht wirklich.

Aufgabe 2.14

Formuliert die folgende Frage an die Präsidentendatenbank als SQL-Anfrage und lasst diese von ORACLE auswerten.

- Bestimmt alle Paare von Präsidenten, die jeweils die gleiche Menge von Hobbys haben. Betrachtet dabei nur die Präsidenten, von denen überhaupt Hobbys bekannt sind.

Die folgenden Erläuterungen sollen Hilfe leisten bei dem Aufbau einer möglichen SQL-Anfrage, die von uns als Musterlösung gedacht ist. Für diese Lösung könnt Ihr die SQL-Operatoren `NOT EXISTS` und `NOT IN` verwenden. Ihr könnt aber sicherlich für die Lösung der Aufgabe auch andere Vorgehensweisen benutzen, wobei die ORACLE-Auswertung jeder richtigen Lösung die unten angegebene Instanz ausgeben soll. In einer Variante der Musterlösung haben wir anstatt `NOT IN`-Operator den `MINUS`-Operator benutzt.

- Die Präsidentenpaare müssen unterschiedlich sein, weil zwei gleiche Präsidenten gleiche Menge von Hobbies haben.
- Die Menge der Hobbies des ersten Präsidenten minus die Menge der Hobbies des zweiten Präsidenten muss eine leere Menge sein. $S1 - S2 = \{\}$
- Die Menge der Hobbies des zweiten Präsidenten minus die Menge der Hobbies des ersten Präsidenten muss ein leere Menge sein. $S2 - S1 = \{\}$
- Wenn $S1 - S2 = \{\}$ und $S2 - S1 = \{\}$ dann $S1 = S2$. Das ist ja genau das was wir wissen wollen: *Die Präsidenten mit der gleichen Menge von Hobbies*

ORACLE-Auswertung:

PRES_NAME	PRES_NAME
Arthur C A	Cleveland G
Cleveland G	Arthur C A
Jackson A	Johnson L B
Jackson A	Taylor Z
Jackson A	Van Buren M
Jefferson T	Washington G
Johnson L B	Jackson A
Johnson L B	Taylor Z
Johnson L B	Van Buren M
Taylor Z	Jackson A
Taylor Z	Johnson L B
Taylor Z	Van Buren M
Van Buren M	Jackson A
Van Buren M	Johnson L B
Van Buren M	Taylor Z
Washington G	Jefferson T

3 Optimierung

Aufgabe 3.1

Betrachtet die folgende SQL-Anfrage:

```
SELECT DISTINCT Pres_Name FROM
  (SELECT DISTINCT P.Pres_Name, P.Party, M.Spouse_Name
   FROM President P, Pres_Marriage M
   WHERE P.Pres_Name=M.Pres_Name)
WHERE Spouse_Name='Welch L'
```

Diese Anfrage gibt den Namen desjenigen Präsidenten aus, dessen Ehefrau 'Welch L' ist.

1. Wandelt die SQL-Anfrage in einen „naheliegenden“ äquivalenten relationalen Ausdruck um. Gebt den Syntaxbaum dieses Ausdrucks an.
2. Optimiert den relationalen Ausdruck nach den euch bekannten Regeln. Gebt den Syntaxbaum der optimierten Anfrage an.
3. Wandelt den optimierten Ausdruck wieder in eine SQL-Anfrage um und prüft, ob Oracle das gleiche Ergebnis wie bei der obigen Anfrage liefert.
4. Erstellt für eure beiden Anfragen einen Ausführungsplan (mit *Analyse* → *PLAN_TABLE leeren*, `EXPLAIN PLAN FOR <anfrage>` und *Analyse* → *Plan anzeigen*).

   Vergleicht die beiden Ausführungspläne und interpretiert die Unterschiede.

Aufgabe 3.2

In der Vorlesung wurden einige Heuristiken zur Optimierung von relationalen Ausdrücken vorgestellt. Einige dieser Heuristiken (und darüber hinaus natürlich noch viele andere) werden auch von Oracle bei der Auswertung von SQL-Anfragen genutzt, andere scheinbar nicht.

Als MaSS der Optimierung soll die Anzahl der Blockzugriffe verwendet werden, die Oracle für die Ausführung benötigt. Mit dem Befehl `SET AUTOTRACE ON` wird SQL*Plus so konfiguriert, dass nach jeder SQL-Anfrage eine Statistik ausgegeben wird. Dort findet sich u.a. der Wert `consistent gets`, der die Anzahl der Blockzugriffe angibt.

Achtung: Es kann vorkommen, dass der Optimierer das Ergebnis einer Anfrage verwendet, um diese beim nächsten Mal noch besser optimieren zu können. Deshalb soll in dieser Aufgabe jeder Befehl jeweils *dreimal* hinterheinander ausgeführt und erst dann die Anzahl der benötigten Blockzugriffe abgelesen werden (Tipp: Mit `r` oder `/` kann man in SQL*Plus den letzten Befehl nochmal ausführen.)

Gegeben sei nun die SQL-Anfrage

```
SELECT HOBBY FROM
  (SELECT DISTINCT HOBBY, PARTY
   FROM PRESIDENT NATURAL JOIN PRES_HOBBY)
WHERE PARTY='Democratic';
```

die die Hobbys der demokratischen Präsidenten ausgibt. Nach dreimaliger Ausführung benötigt Oracle 67 Blockzugriffe, um die Anfrage zu berechnen:

```
Statistics
----------------------------------------------------------
         67  consistent gets
```

1. Wandelt die SQL-Anfrage in einen äquivalenten relationalen Ausdruck um. Gebt den Syntaxbaum dieses Ausdrucks an.

2. Optimiert den relationalen Ausdruck nach den euch bekannten Regeln. Gebt den Syntaxbaum der optimierten Anfrage an.

3. Wandelt den optimierten Ausdruck wieder in eine SQL-Anfrage um. Wie viele Blockzugriffe benötigt Oracle nun noch, um die Anfrage zu berechnen?

4. Die folgende SQL-Anfrage ist äquivalent zu der Anfrage oben und benötigt ebenfalls 67 Blockzugriffe:

   ```
   SELECT DISTINCT HOBBY
   FROM PRES_HOBBY
   WHERE PRES_NAME IN
     (SELECT PRES_NAME
     FROM PRESIDENT
     WHERE PARTY='Democratic');
   ```

   Durch eine kleine Änderung lässt sich eine äquivalente Anfrage finden, die erstaunlicherweise mit nur 8 Blockzugriffen auskommt. Wie lautet die modifizierte Anfrage? Warum ist sie so viel schneller?

Für die Beantwortung der zweiten Frage ist es möglicherweise sinnvoll, sich die detaillierten Ausführungspläne anzeigen zu lassen:

```
SET AUTOTRACE OFF
EXPLAIN PLAN FOR SELECT ... FROM ...;
SELECT * FROM TABLE(dbms_xplan.display);
DELETE PLAN_TABLE;
```

Aufgabe 3.3

Findet ihr eine weitere, möglichst einfache, SQL-Anfrage an die Präsidentendatenbank, für die ihr eine zweite äquivalente SQL-Anfrage angeben könnt, die mit weniger lesenden Blockzugriffen (siehe Hinweis) auskommt, als die von Oracle optimierte erste Anfrage? Auch hier solltet ihr die Anfrage drei Mal ausführen, um den Effekt der weiteren Optimierung weitestgehen auszuschalten.

Tipp: Es ist gar nicht so leicht, solche Anfragen zu finden; die Ausnutzung des Absorbtiv- bzw. Idempotenz-Gesetzes des natürlichen Verbunds hat sich als vielversprechend erwiesen.

Eure Lösung sollte aus folgenden Teilen bestehen:

- eure Ausgangsanfrage als SQL-Statement
- die Ausgabe (Ergebnis und Statistik) des *dritten* Anfragedurchlaufs der Ausgangsanfrage
- eure Ausgangsanfrage als Ausdruck der relationalen Algebra in Form eines Syntaxbaums
- eure von Hand optimierte Anfrage als SQL-Statement
- die Ausgabe des *dritten* Anfragedurchlaufs der von Hand optimierten Anfrage
- eure von Hand optimierte Anfrage als Ausdruck der relationalen Algebra in Form eines Syntaxbaums

4 XPath-Aufgaben

Aufgabe 4.1

Ein Teil der Präsidentendatenbank wurde in XML-Dokumente exportiert (`XMLPRES`). Für jeden Präsidenten gibt es ein Dokument, das jeweils Informationen über ihn, seine Hobbys und seine Ehefrauen enthält. Die folgende DTD beschreibt den Aufbau dieser Dokumente:

```
<!ELEMENT president (pres_name, birth_year, years_serve,
     death_age?, party, state_born, hobby*,
     pres_marriage*)>

<!ELEMENT pres_marriage (spouse_name, pr_age, sp_age,
                         nr_children, mar_year)>
```

(Alle anderen Elemente sind vom Typ `#PCDATA`.)

1. Gebt XPath-Anfragen an, die genau die folgenden Knoten selektieren:
   (a) Das Todesalter eines Präsidenten.
   (b) Alle Hobbys eines Präsidenten.
   (c) Die Namen aller Ehefrauen eines Präsidenten.
2. Gebt Oracle-Anfragen an, die mit Hilfe von XPath folgenden Fragen beantworten:
   (a) Wie heißen die Ehefrauen derjenigen Präsidenten, deren Hobby Poker ist?
   (b) Wie alt wurde der Ehemann von „Davis N“?

Hinweise zu Oracle und XML

- Die Dokumente befinden sich in einer Tabelle namens `XMLPres`. Zu jedem Präsidenten gibt es ein Tupel. Jedes Tupel besitzt genau ein Attribut namens `document`, das das XML-Dokument enthält.
- XPath-Anfragen können mit der Funktion `EXTRACT(<XML-Doc>, <XPath-Expr>)` ausgewertet werden. Dabei ist `<XML-Doc>` ein XML-Dokument und `<XPath-Expr>` die XPath-Anfrage, die ausgewertet werden soll. Die Funktion liefert alle Knoten zurück, die die XPath-Anfrage selektiert hat. Um einen Knoten dieser Art darzustellen, benötigt man noch die Anweisung `getStringVal()`, die ihn in einen String konvertiert.
- Die Funktion `EXISTSNODE(<XML-Doc>, <XPath-Expr>)` gibt genau dann `1` zurück, wenn die XPath-Anfrage `<XPath-Expr>` mindestens einen Knoten im Dokument `<XML-Doc>` gefunden hat, sonst `0`.

- Beispiel für eine Anfrage: Welche Präsidenten waren nicht verheiratet?

```
SELECT EXTRACT(document, '/president/pres_name').getStringVal()
FROM XMLPres
WHERE EXISTSNODE(document, '/president/pres_marriage') = 0;
```

5 Transaktionen

Aufgabe 5.1

Betrachtet folgende Transaktionen, die alle auf eine Tabelle zugreifen. Startet diese Transaktionen parallel in zwei verschiedenen SQL*Plus-Sitzungen. Gebt die folgenden Befehle in den entsprechenden Sitzungen in der unten angegebenen Reihenfolge an. Protokolliert die Ausgaben und interpretiert den Ablauf der Ausführung der Transaktionen.

Sitzung 1	Sitzung 2

`SET TIME ON`
`ALTER SESSION SET ISOLATION_LEVEL =`
`SERIALIZABLE;`	

	`SET TIME ON`
	`ALTER SESSION SET ISOLATION_LEVEL =`
	`SERIALIZABLE;`

`CREATE TABLE T (N VARCHAR2(1) PRIMARY`
`KEY, V NUMBER(1));`
`INSERT INTO T VALUES ('a', 1);`
`COMMIT;`	

	`SELECT * FROM T;`
`UPDATE T SET V = 8 WHERE N = 'a';`	
	`UPDATE T SET V = 7 WHERE N = 'a';`
`COMMIT;`	
	`COMMIT;` `SELECT * FROM T;`

Aufgabe 5.2

Gegeben sei die folgende Oracle-Sitzung.

```
SQL> create table x (n varchar2(1) primary key, v number(1));
Table created.

SQL> insert into x values('a', 1);
1 row created.

SQL> insert into x values('b', 2);
1 row created.

SQL> select * from x;
N          V
- ----------
a          1
b          2

SQL> savepoint sp1;
```

```
Savepoint created.

SQL> insert into x values('d', 4);
1 row created.

SQL> select * from x;
N          V
- ----------
a          1
b          2
d          4

SQL> rollback to savepoint sp1;
Rollback complete.

SQL> select * from x;
N          V
- ----------
a          1
b          2

SQL> commit;
Commit complete.

SQL> select * from x;
N          V
- ----------
a          1
b          2

SQL> insert into x values('c', 3);
1 row created.

SQL> rollback;

Rollback complete.

SQL> select * from x;
N          V
- ----------
a          1
b          2
```

1. Was passiert in dieser Sitzung?

2. In einer Oracle-Datenbank wurde eine Tabelle mittels:

```
CREATE TABLE X (N VARCHAR2(1) PRIMARY KEY, V NUMBER(1))
```

angelegt. Sie enthält zu Beginn eine Zeile mit den Werten `'A'` und `1`. Betrachtet zwei Transaktionen, die beide auf diese Tabelle zugreifen. Für jede dieser Transaktionen wird jeweils eine SQL*Plus-Sitzung gestartet. Die linke und rechte Spalte der auf der nächsten Seite abgebildeten Tabelle geben jeweils die in den entsprechenden Sitzungen ausgeführten Anweisungen an.

```
SQL> SET TIME ON
                                          SQL> SET TIME ON
15:18:12 SQL> ALTER SESSION SET
ISOLATION_LEVEL = SERIALIZABLE;

Session altered.
                                          15:18:30 SQL> ALTER SESSION SET
                                          ISOLATION_LEVEL = SERIALIZABLE;

                                          Session altered.
15:19:08 SQL> UPDATE X SET V = 2 WHERE
N = 'A';

1 row updated.

15:19:28 SQL> COMMIT;
Commit complete.
                                          15:20:39 SQL> SELECT V FROM X WHERE N =
                                          'A';

                                                   V
                                          ----------
                                                   1

                                          15:21:15 SQL> COMMIT;

                                          Commit complete.

                                          15:21:37 SQL> SELECT V FROM X WHERE N =
                                          'A';

                                                   V
                                          ----------
                                                   2

                                          15:22:24 SQL> COMMIT;
                                          Commit complete.
```

Interpretiert die Oracle-Sitzungen mit Hilfe des Read/Write-Modells als Plan.

Weitere Einzelheiten zu den oben vorgestellten Befehlen und zur Transaktionsverwaltung in Oracle finden sich in der Oracle11g-Dokumentation.

6 Andere Aufgaben

Aufgabe 6.1

Stellt Vermutungen an, was die folgenden Oracle-SQL-Anfragen berechnen.

1.
```
SELECT 2+3 FROM DUAL;
```

2.
```
SELECT USER FROM DUAL;
```

3.
```
SELECT election_year, candidate,
  DECODE(winner_loser_indic, 'W', 'Sieger', '')
FROM election
WHERE election_year >= 1980;
```

4.
```
SELECT pres_name,
  COALESCE(TO_CHAR(death_age), 'Lebt noch') todesalter
FROM president
WHERE birth_year >= 1900
ORDER BY todesalter;
```

5.
```
WITH pres_jh AS (SELECT p.*, TRUNC((birth_year+99)/100) Jh
                 FROM president p)
SELECT Jh, MIN(death_age), AVG(death_age), MAX(death_age)
FROM pres_jh
GROUP BY Jh
ORDER BY 1;
```

6.
```
WITH pres_nn AS (
  SELECT pres_name, birth_year,
    SUBSTR(pres_name, 1, INSTR(pres_name, ' ') - 1) nachname
  FROM president)
SELECT p1.pres_name, p1.birth_year, p2.pres_name, p2.birth_year
FROM pres_nn p1, pres_nn p2
WHERE p1.nachname = p2.nachname
  AND p1.pres_name != p2.pres_name
  AND p1.birth_year < p2.birth_year;
```

7.
```
SELECT pres_name
FROM president
WHERE pres_name NOT IN (
  SELECT candidate
  FROM election);
```

8.
```
SELECT pres_name
FROM president
WHERE NOT EXISTS (
  SELECT *
  FROM election
  WHERE candidate = pres_name
    AND winner_loser_indic = 'W');
```

Aufgabe 6.2

Welche der folgenden Oracle-SQL-Anfragen an die Präsidentendatenbank sind gleichwertig und welche nicht? Warum?

```
/* 1 */
SELECT p.pres_name,
       h.hobby, p.party
FROM pres_hobby h, president p
WHERE p.pres_name = h.pres_name
  AND state_born = 'Ohio';

/* 2 */
SELECT p.pres_name, hobby, party
FROM pres_hobby h JOIN president p
  ON (h.pres_name = p.pres_name)
WHERE state_born = 'Ohio';

/* 3 */
SELECT pres_name, hobby, party
FROM pres_hobby
  NATURAL JOIN
  president
WHERE state_born = 'Ohio';

/* 4 */
SELECT pres_name, hobby, party
FROM pres_hobby
  NATURAL RIGHT OUTER JOIN
  president
WHERE state_born = 'Ohio';

/* 5 */
SELECT pres_name, hobby, party
FROM pres_hobby
  NATURAL LEFT OUTER JOIN
  president
WHERE state_born = 'Ohio';
```

```
/* 6 */
SELECT p.pres_name, hobby, party
FROM pres_hobby h JOIN president p
  ON (h.pres_name = p.pres_name
      AND state_born = 'Ohio');

/* 7 */
SELECT pres_name, hobby, party
FROM pres_hobby NATURAL JOIN
  (SELECT *
   FROM president
   WHERE state_born = 'Ohio');

/* 8 */
SELECT h.*,
 (SELECT party
  FROM president p
  WHERE p.pres_name = h.pres_name)
   Party
FROM pres_hobby h
WHERE
  (SELECT state_born
   FROM president p
   WHERE p.pres_name = h.pres_name)
   = 'Ohio';

/* 9 */
SELECT p.pres_name,
 (SELECT hobby
  FROM pres_hobby h
  WHERE p.pres_name = h.pres_name)
    AS Hobby,
  p.party
FROM president p
WHERE state_born = 'Ohio';
```

Aufgabe 6.3

Vor wievielen Jahren fand die letzte Hochzeit eines Präsidenten statt? Die Anfrage soll in syntaktisch unveränderter Form auch in allen zukünftigen Jahren das korrekte Ergebnis liefern! Weiterführende Informationen finden sich in der „Oracle 11*g* SQL Language Reference“ unter dem Stichwort „date functions“.

Aufgabe 6.4

In der Oracle-Datenbank wurde eine Tabelle `cocktails` mit dem folgenden Befehl angelegt:

```
CREATE TABLE cocktails (
  name VARCHAR2(50) PRIMARY KEY,
  rezept VARCHAR2(4000) );
```

Dann wurden einige Dokumente in die Tabelle eingefügt und anschließend der folgende Befehl ausgeführt:

```
CREATE INDEX idx_cocktail_ir ON cocktails(rezept) INDEXTYPE IS CTXSYS.CONTEXT;
```

Die Tabelle enthält in der Spalte `rezept` Cocktailrezepte. Die Spalte `name` enthält den Namen des Cocktails.

1. Betrachtet die folgenden Oracle-Text-Anfragen und ihre Auswertungen:

```
SELECT name, SCORE(1)
FROM cocktails
WHERE CONTAINS(rezept,'gemixt',1) > 0;
```

NAME	SCORE(1)
Cosmopolitan	5
Manhattan	5
Margarita	5

```
SELECT name, SCORE(1)
FROM cocktails
WHERE CONTAINS(rezept,'Erdbeersirup',1) > 0;
```

NAME	SCORE(1)
Margarita	6

   Sowohl das Wort `gemixt` als auch das Wort `Erdbeersirup` kommt im Margarita-Rezept genau einmal vor. Stellt Vermutungen an, warum sich dennoch unterschiedliche `SCORE`-Werte ergeben.

2. Was berechnen die folgenden Anfragen?

   - ```
     SELECT name, SCORE(1)
     FROM cocktails
     WHERE CONTAINS(rezept, 'Eis AND (Rum OR Whisky)', 1) > 0
     ```
   - ```
     SELECT name, SCORE(1)
     FROM cocktails
     WHERE CONTAINS(rezept, 'Rum NOT Orange%', 1) > 0
     ```
   - ```
     SELECT name, SCORE(1)
     FROM cocktails
     WHERE CONTAINS(rezept, '?Whiskey', 1) > 0
     ```
   - ```
     SELECT name, SCORE(1)
     FROM cocktails
     WHERE CONTAINS(rezept, '$mixen', 1) > 0
     ```

3. Gebt zur `cocktails`-Tabelle eine Oracle-Text-Anfrage an, die die folgende Frage beantwortet:

   Welche Cocktailrezepte werden bzgl. der folgenden Anfrage als relevant angesehen?

   > Gesucht sind Cocktails, die mindestens eine der Spirituosen Wodka, Gin oder Tequila, aber keinen Orangensaft enthalten.

   Es soll jeweils der Name des Cocktails und der `SCORE`-Wert des Cocktailrezepts ausgegeben werden. Die Ausgabe soll absteigend nach der angenommenen Relevanz sortiert werden.

4. Gebt zur `cocktails`-Tabelle eine Oracle-Text-Anfrage an, die die folgende Frage beantwortet:

   Welche Cocktailrezepte werden bzgl. der folgenden Anfrage als relevant angesehen?

   > Gesucht sind Cocktails, die Cola oder eine Whisk(e)y-Sorte enthalten. Der Whisk(e)y soll dabei doppelt so stark bewertet werden wie die Cola.

   Es soll jeweils der Name des Cocktails und der `SCORE`-Wert des Cocktailrezepts ausgegeben werden. Die Ausgabe soll absteigend nach der angenommenen Relevanz sortiert werden.

Aufgabe 6.5

Implementiert eine kleine JAVA-Anwendung, die in die Tabelle `IS_TEILNEHMER` einträgt, ob ihr am Ende des Semesters einen Schein zu den Übungen erwerben wollt oder nicht (keine Sorge, der Eintrag ist nicht verbindlich und hat auch sonst keine Auswirkungen).

Euer Programm soll folgende Anforderungen erfüllen:

- Der Datenbankzugriff soll mit JDBC erfolgen.
- Jegliche Benutzerinteraktion soll im Textmodus stattfinden, d. h. Ihr sollt weder AWT noch Swing benutzen.
- Euer Passwort für die Oracle-Datenbank darf aus naheliegenden Gründen nicht im Klartext im Quellcode zu finden sein! Lest das Passwort am besten interaktiv von der Konsole ein. Achtet darauf, bei der Eingabe des Passwortes das Konsolen-Echo auszuschalten, z. B. indem ihr es von vornherein am Terminal abschaltet:

```
bash-2.05$ stty -echo; java <classname>; stty echo
```

7 SQL-Befehlsübersicht

Hier befindet sich eine (unvollständige) Übersicht über einige SQL-Befehle. Weiterführende Informationen finden sich in der Oracle11g-Dokumentation (`http://www.oracle.com/pls/db111/db111.homepage`):

Tabellen anlegen mit `CREATE TABLE` und wieder löschen mit `DROP TABLE`
Mit `INSERT INTO` werden neue Elemente mit folgender Syntax in Tabellen eingefügt:
`INSERT INTO <tableName>(<Liste der Attribute, durch Kommata getrennt>)`
`VALUES(<Liste der Werte f"ur die Attribute in ihrer Reihenfolge>);`

Anfragen mit `SELECT ... FROM ...` [`WHERE ...`]
Um sich alle Tabellen anzeigen zu lassen kann man den Befehl `SELECT table_name FROM all_tables` nutzen und mit `SELECT * FROM cat` werden alle selbst angelegten Tabellen angezeigt.

arithmetische Operatoren: `+`, `-`, `*`, `/`

Konkatenationsoperator: `||`

Mengenoperatoren: `UNION`, `UNION ALL`, `INTERSECT`, `MINUS`

Vergleichsoperatoren: `=`, `!=`, `^=`, `<>`, `¬=`, `>`, `<`, `>=`, `<=`, `IN`, `NOT IN`, `ANY`, `SOME`, `ALL`, [`NOT`] `BETWEEN` x `AND` y, `EXISTS`, x [`NOT`] `LIKE` y [`ESCAPE` *'z'*], `IS` [`NOT`] `NULL`

logische Operatoren: `NOT`, `AND`, `OR`

`CASE`**-Ausdruck:** `CASE` *expr* `WHEN` *comp-expr* `THEN` *return-expr*
(`,` `WHEN` *comp-expr* `THEN` *return-expr*)* [`ELSE` *else-expr*],
`CASE WHEN` *condition* `THEN` *return-expr*
(`,` `WHEN` *condition* `THEN` *return-expr*)* [`ELSE` *else-expr*]

skalarer Unteranfragenausdruck: (*subquery*)

Zahlenfunktionen: `ABS`, `ACOS`, `ASIN`, `ATAN`, `ATAN2`, `BITAND`, `CEIL`, `COS`, `COSH`, `EXP`, `FLOOR`, `LN`, `LOG`, `MOD`, `POWER`, `ROUND`, `SIGN`, `SIN`, `SINH`, `SQRT`, `TAN`, `TANH`, `TRUNC`, `WIDTH_BUCKET`

Zeichenfunktionen: `CHR`, `CONCAT`, `INITCAP`, `LOWER`, `LPAD`, `LTRIM`, `NLS_INITCAP`, `NLS_LOWER`, `NLS_UPPER`, `NLSSORT`, `REPLACE`, `RPAD`, `RTRIM`, `SOUNDEX`, `SUBSTR`, `TRANSLATE`, `TREAT`, `TRIM`, `UPPER`

Zeichenfunktionen, die Zahlen als Ergebnis liefern: `ASCII`, `INSTR`, `LENGTH`

Datumsfunktionen: `ADD_MONTHS`, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, `DBTIMEZONE`, `EXTRACT`, `FROM_TZ`, `LAST_DAY`, `LOCALTIMESTAMP`, `MONTHS_BETWEEN`, `NEW_TIME`, `NEXT_DAY`, `NUMTODSINTERVAL`, `NUMTOYMINTERVAL`, `ROUND`, `SESSIONTIMEZONE`, `SYS_EXTRACT_UTC`, `SYSTIMESTAMP`, `SYSDATE`, `TO_DSINTERVAL`, `TO_TIMESTAMP`, `TO_TIMESTAMP_TZ`, `TO_YMINTERVAL`, `TRUNC`, `TZ_OFFSET`

Umwandlungsfunktionen: `ASCIISTR`, `BIN_TO_NUM`, `CAST`, `CHARTOROWID`, `COMPOSE`, `CONVERT`, `DECOMPOSE`, `HEXTORAW`, `NUMTODSINTERVAL`, `NUMTOYMINTERVAL`, `RAWTOHEX`, `RAWTONHEX`, `ROWIDTOCHAR`, `ROWIDTONCHAR`, `TO_CHAR`, `TO_CLOB`, `TO_DATE`, `TO_DSINTERVAL`, `TO_LOB`, `TO_MULTI_BYTE`, `TO_NCHAR`, `TO_NCLOB`, `TO_NUMBER`, `TO_SINGLE_BYTE`, `TO_YMINTERVAL`, `TRANSLATE ... USING`, `UNISTR`

sonstige Funktionen: `BFILENAME`, `COALESCE`, `DECODE`, `DUMP`, `EMPTY_BLOB`, `EMPTY_CLOB`, `EXISTSNODE`, `EXTRACT`, `GREATEST`, `LEAST`, `NLS_CHARSET_DECL_LEN`, `NLS_CHARSET_ID`, `NLS_CHARSET_NAME`, `NULLIF`, `NVL`, `NVL2`, `SYS_CONNECT_BY_PATH`, `SYS_CONTEXT`, `SYS_DBURIGEN`, `SYS_EXTRACT_UTC`, `SYS_GUID`, `SYS_TYPEID`, `SYS_XMLAGG`, `SYS_XMLGEN`, `UID`, `USER`, `USERENV`, `VSIZE`

Aggregatfunktionen: `AVG`, `CORR`, `COUNT`, `COVAR_POP`, `COVAR_SAMP`, `CUME_DIST`, `DENSE_RANK`, `FIRST`, `GROUP_ID`, `GROUPING`, `GROUPING_ID`, `LAST`, `MAX`, `MIN`, `PERCENTILE_CONT`, `PERCENTILE_DISC`, `PRECENT_RANK`, `RANK`, `REGR_...`, `STDDEV`, `STDDEV_POP`, `STDDEV_SAMP`, `SUM`, `VAR_POP`, `VAR_SAMP`, `VARIANCE`

Objektreferenzfunktionen: DEREF, MAKE_REF, REF, REFTOHEX, VALUE

benannte Unteranfragen:
WITH *queryname* AS (*subquery*)(, *queryname* AS (*subquery*))* SELECT ...

Verbundoperationen: SELECT ... FROM *table* NATURAL JOIN *table* ...

Spaltenalias: SELECT *expr* [AS] *alias* ...

Sortieren SELECT ... ORDER BY *expr* [ASC|DESC](, *expr* [ASC|DESC])*

Gruppieren: SELECT ... GROUP BY *expr*(, *expr*)* [HAVING *condition*]

Outer Join: SELECT ... FROM *table* NATURAL (LEFT|RIGHT|FULL) [OUTER] JOIN *table* ...

connect_by_condition ≡ (PRIOR *expr comparison_condition expr* |
expr comparison_condition PRIOR *expr*)

LEVEL, SYS_CONNECT_BY_PATH(*column*, *char*)

Indexe: CREATE INDEX *index* ON *table* (*column*(, *column*)*) ;

Links: CREATE CLUSTER *cluster* (*column datatype*(, *column datatype*)*) [INDEX | HASHKEYS *integer*] ;

CREATE INDEX *index* ON CLUSTER *cluster* ;

CREATE TABLE ... CLUSTER *cluster* (*column*(, *column*)*) ;

Ausführungspläne: @?/rdbms/admin/utlxplan /* PLAN_TABLE anlegen */

ANALYZE (TABLE *table* | INDEX *index* | CLUSTER *cluster*) (COMPUTE STATISTICS | DELETE STATISTICS) ;

EXPLAIN PLAN [SET STATEMENT_ID = '*text*'] FOR *statement* ;

@?/rdbms/admin/utlxpls /* PLAN_TABLE auswerten */

algebraische Operationen: UNION, MINUS, INTERSECT

Vergleichsoperatoren: =, != (<>, ^=), <, <=, >, >=, ...BETWEEN...AND..., ...IS NULL, ...IS NOT NULL, ...LIKE... (z.B. x LIKE '__a%')

arithmetische Operatoren: +, -, *, /, MOD(...,...), POWER(...,...), ROUND(...,...), SQRT(...), ...

Textverarbeitungsoperatoren: ...||... (Konkatenation), LENGTH(...), SUBSTR(x, von_position, bis_position), ...

Typumwandlungsoperatoren: TO_CHAR(...), TO_NUMBER(...), ...

Unteranfragen: x IN (SELECT ...), x NOT IN (SELECT ...), EXISTS(SELECT ...), NOT EXISTS(SELECT ...)

Ausgabeformatierung: SELECT feldname "neue Überschrift"
...FROM...WHERE,
SELECT...FROM...WHERE...ORDER BY feldname [ASC|DESC][,...]

Hierarchische Anfragen `SELECT ... FROM ... [WHERE ...]`
`[START WITH` *condition*`]`
`/*` Auswahl der Wurzeln `*/`
`CONNECT BY` *connect_by_condition*
`/*` Verbindung zw Eltern und Kindern `*/`
`[ORDER SIBLINGS BY ...]`
`/*` Ordnung innerhalb der Kinder `*/`
connect_by_condition ::= `PRIOR` *expr comparison_condition expr* |
expr comparison_condition `PRIOR` *expr*

Pseudoattribut: `LEVEL` (gibt die Hierarchie-Ebene jeder ausgegebenen Zeile aus)

Funktion: `SYS_CONNECT_BY_PATH(`*column*`,` *char*`)`
(gibt für *column* den Pfad von der Wurzel bis zum Attributwert aus und verwendet dabei *char* als Trennzeichen)

`||` ist die Konkatenation von zwei Zeichenketten und `TO_CHAR` die Umwandlung in eine Zeichenkette.

Information Retrieval/Oracle Text Mit einem Index vom Typ `CTXSYS.CONTEXT` können Text-Retrieval-Anfragen auf den indizierten Spalten ausgeführt werden. Oracle SQL stellt dafür die beiden Operatoren `CONTAINS` und `SCORE` zur Verfügung:

- `CONTAINS(`*column*`,` *query* `[,` *label*`])`
  bestimmt, als wie relevant das Dokument in der Spalte *column* bezüglich der Anfragezeichenkette *query* angesehen wird. Als Ergebnis wird eine Zahl zwischen 0 (nicht relevant) und 100 (sehr relevant) zurückgegeben. Optional ist es möglich, eine Zahl als *label* anzugeben, um mit dem `SCORE`-Operator auf das Ergebnis zuzugreifen.
- `SCORE(`*label*`)`
  liefert das gleiche Ergebnis wie der `CONTAINS`-Operator mit dem gleichen *label*.

Genauere Informationen findet man in „Oracle® Text Reference“ und „Oracle® Text Application Developer's Guide“.

Ausführungsstatistiken Gibt man in SQL*Plus den Befehl `SET AUTOTRACE ON STATISTICS` ein, werden bei jedem im Folgenden bearbeiteten SQL-DML-Befehl (`SELECT`, `INSERT`, `UPDATE` oder `DELETE`) einige statistische Daten zur Ausführung angezeigt. Unter anderem wird der Wert `consistent gets` ausgegeben, der die Anzahl der lesenden Blockzugriffe angibt. Mit

```
SET AUTOTRACE OFF
```

kann man die automatische Anzeige der Ausführungsstatistiken wieder abstellen.

Wenn der „`SET AUTOTRACE ON`“-Befehl mit einer Fehlermeldung abbricht, wurde möglicherweise die Tabelle `PLAN_TABLE` für die Speicherung von Ausführungsplänen noch nicht angelegt (siehe Übungsblatt Nr. 9). Dies kann mit dem Befehl

```
@/home/is/sqlplus/utlxplan.sql
```

nachgeholt werden.

Transaktionen:

- `COMMIT` [`WORK`] `/*` oder ein DDL-Befehl `*/`
  SchlieSSt eine Transaktion ab.
- `SAVEPOINT` *savepoint*
  Setzt einen Savepoint mit dem Namen *savepoint* auf.
- `ROLLBACK` [`WORK`] [`TO` [`SAVEPOINT`] *savepoint*]
  Macht die Änderungen seit Beginn der aktuellen Transaktion oder seit dem angegebenen Savepoint rückgängig.
- `ALTER SESSION SET ISOLATION_LEVEL =` (`SERIALIZABLE` | `READ COMMITTED`)
  Setzt den Isolierungsgrad einer Sitzung.
- `SET TRANSACTION ISOLATION LEVEL` (`SERIALIZABLE` | `READ COMMITTED`)
  Setzt den Isolierungsgrad der aktuellen Transaktion.

Oracle unterscheidet dabei die folgenden zwei Isolierungsgrade:

`SERIALIZABLE`**:** Dieser Isolierungsgrad ist nötig damit Oracle dafür sorgt, dass die Transaktionen, wie in der Vorlesung vorgestellt, *serialisierbar* sind.

`READ COMMITTED`**:** Dies ist ein schwächerer Isolierungsgrad, der nicht unbedingt sicherstellt, dass die Transaktionen *serialisierbar* sind. Es kann hierbei dazu kommen, dass eine wiederholte Leseanforderung innerhalb einer Transaktion andere Daten liefert als zuvor, falls das Objekt, dessen Wert gelesen wird, mittlerweile durch andere Transaktionen geändert wurde, die seit der ersten Leseanforderung endeten.