LOE

Lösungen
zum Seminar 4051

1 Lösungen

1.1 Lösungen zum Datenmodell

Das Datenmodell betrachten und die logischen Beziehungen zwischen den Tabellen nachvollziehen, da sämtliche Beispiele und Übungen in diesem Seminar auf diesen Daten basieren.

1.2 Lösungen zu RDBMS

1.2.1 Lösungen zu relationalem Modell

a) Eine Tabelle ist eine in Spalten und Zeilen aufgeteilte organisierte Sammlung von Daten. **Alle Daten einer Datenbank befinden sich in Tabellen, nirgendwo anders!**

b) Die Zeilen, auch als Datensätze bezeichnet, stellen einzelne Objekte der Tabelle dar.

c) In den Spalten sind bestimmte Eigenschaften (Attribute) eines Objekts abgelegt.

d) In einem Feld ist genau EINE Eigenschaft eines Objektes abgelegt. Z. B. der Mitarbeiter mit der Personalnummer `7369` heißt `SMITH` (und nur `SMITH`). Diese Information steht im Feld `ENAME` der Tabelle `EMP`.

e) Sonst wäre die DB nicht relational!

1.2.2 Lösungen zu ERM

a) Auf einem Blatt Papier das Modell entwerfen. Siehe Beispiel im Kapitel 1.3.2 ERM

b) AbtNr, Name, Standort, Abtleiter, AbtLeiterStellvertreter, AbtTelNr, AbtFax, AbteMail, …

c) PersonalNr, Name, Vorname(n), Adresse, PLZ, Stadt, TelNr, TelNrPrivat, eMail, Handy, Geburtsjahr, Einstellungsdatum, Verdienst, ...

d) Beispiel:

⇒ Kunden/Lieferanten vs. Interne Praktikanten,

⇒ Externe Kundenberater vs. Interne Mitarbeiter,

⇒ Netzwerkadministrator vs. Sachbearbeiter,

⇒ Versicherungsbranche vs. Schraubenhersteller,

⇒ etc…

e) MUSS, KANN, 1:1, 1:n, n:1, n:m

f) Personalnummer, weil es diese in der ganzen Firma nie zweimal gibt.

g) Primärschlüssel. Rest kann nachgefüllt werden.

h) Weil man sie nachträglich ergänzen kann.

i) Nein, weil es mehrere Menschen auf der Welt gibt (und somit in der Tabelle geben könnte), die den gleichen Namen tragen.

j) Wenn z. B. ein Mitarbeiter nicht zwingend notwendig einer Abteilung zugeordnet werden muss, spricht man von einer KANN-Beziehung, ansonsten von einer MUSS-Beziehung.

k) Sinnvoll ist es, wenn man eine Abteilung aus dem Unternehmen entfernen möchte und sämtliche Mitarbeiter gleichzeitig entlässt. Wenn man wiederum die Mitarbeiter behalten und nur die Abteilung entfernen (Umstrukturieren der Firma) möchte, wäre es unklug die Mitarbeiter der betroffenen Abteilung auch zu löschen.

l) Alle betroffenen Gruppen, die mit diesen Daten in Kontakt kommen. Meist sind es DIE FachabteilungEN.

1.2.3 Lösungen zu Data Dictionary (DD)

a) Externe Ebene und Interne Ebene

b) `USER_`, `ALL_`, `DBA_`

c) Alle Tabellen, die mir als Benutzer `sl01` (`sl02`, ...) persönlich gehören.

d) Alle Tabellen aller Benutzer auf die ich als Benutzer `sl01` (`sl02`, …) Zugriff habe.

e) Alle Tabellen der Datenbank.

1.3 Lösungen zu SQL*Plus

a)
```
DESC emp;
DESC dept;
DESC salgrade;
```

b)
```
SET LINESIZE 200;
```
→ stellt die Ausgabebreite auf 200 Zeichen pro Zeile

```
SET PAGESIZE 100;
```
→ stellt die Seitenlänge auf 100 Zeilen

```
COL <spalte> FORMAT a30;
```
→ formatiert die Ausgabebreite der Textspalte auf 30 Zeichen

```
COL <spalte> FORMAT L99,999.99;
```
→ formatiert die Ausgabe einer numerischen Spalte im angegebenen Format

c)
```
SPOOL mitarb.lst;
SELECT * FROM emp;
SELECT * FROM dept;
SELECT * FROM salgrade;
SPOOL OFF;
```

d)
```
SET HEADING OFF ECHO OFF FEEDBACK OFF;
SPOOL mitarb.lst;
SELECT * FROM emp;
SELECT * FROM dept;
SELECT * FROM salgrade;
SPOOL OFF;
```

e)
```
LIST;
```

Es wird das letzte SQL Kommando angezeigt, jedoch kein "`DESC emp;`", weil SQL*Plus-Kommandos nicht im Puffer abgespeichert werden.

f) Erst im Explorer das Verzeichnis "Uebungen" anlegen, dann im SQL*Plus:

```
SAVE C:/Uebungen/test1.sql;
```

g)
```
START C:/Uebungen/test1.sql;
```

h)
```
SET TIMING ON;
START C:/Uebungen/test1.sql;
(oder @ C:/Uebungen/test1.sql;)
```

i)
```
COL(UMN) ename FOR(MAT) A30;
```

j)
```
COL(UMN) sal FOR(MAT) 09,999.99;
```

k) Weil "." NUR als Dezimaltrennzeichen benutzt werden kann! Nach 09 kommt "," und kein "."

l) Die Einstellungen bleiben bis zum Ende der Session bestehen, wenn sie nicht vom Benutzer auf den Default-Wert oder einen anderen Wert gesetzt werden. Parameter in der `GLOGIN.SQL` Datei werden beim Start von SQL*Plus automatisch ausgeführt.

1.4 Lösungen zu SELECT

1.4.1 Lösungen zu Allgemeine Übungen zu SELECT

a)	Ja	b)	Ja	c)	Nein
d)	Ja	e)	Ja	f)	Nein
g)	Ja	h)	Ja	i)	Nein

1.4.2 Lösungen zu Verständnisübungen zu SELECT

a) Viermal aktuelles Tagesdatum.

b) Jede mögliche Berufs/Abteilungsnummerkombination ohne doppelte Einträge.

c) Erzeugt 14-mal den Satz "Select xxx from emp", für jeden Mitarbeiter mit entsprechendem Namen.

d) Es wird 14-mal der Satz ausgegeben:

```
Hallo <name> du verdienst <gehalt>.
```

e) Es wird das doppelte Gehalt und der Mitarbeitername für jeden Mitarbeiter ausgegeben.

f) Ausgabe von Mitarbeiternummer, Name und der Zahl 9999 für jeden Mitarbeiter.

1.4.3 Lösungen zu Übungen zu SELECT

a)
```
SELECT * FROM emp;
SELECT * FROM dept;
```

b)
```
SELECT      e.empno,
            e.ename    AS "Mitarbeiter",
            e.sal      AS "Monatsverdienst"
FROM   emp e;
```

Die Spalte `ENAME` ist "nur" 10 Zeichen, das Wort "Mitarbeiter" jedoch 11 Zeichen breit. Die Spaltenüberschrift `"Mitarbeite"` kann mit dem `COLUMN` Befehl auf die richtige Breite erweitert werden.

c)
```
SELECT      d.deptno,
            d.dname    AS "Abteilung",
            d.loc      AS "Ort"
FROM   dept d;
```

d)
```
SELECT DISTINCT e.job FROM emp e;
```

e)
```
SELECT DISTINCT e.sal FROM emp e;
```

1.4.4 Lösungen zu WHERE Klausel, Operatoren und NULL-Werten

a)
```
SELECT * FROM emp WHERE comm IS NULL;
SELECT * FROM emp WHERE comm = 0;
SELECT * FROM emp WHERE NVL(comm, 0) = 0;
```

b)
```
SELECT      e.empno,
            e.ename,
            e.sal AS "Monatsverdienst",
            e.sal + NVL(e.comm, 0)
                          AS "Monatsgehalt"
FROM        emp e;
```

c)
```
SELECT      e.empno,
            e.ename,
            12 * (e.sal + NVL(e.comm, 0))
                           AS "Jahresgehalt"
FROM        emp e;
```

d)
```
SELECT      e.empno,
            e.ename,
            12 * NVL(e.sal, 0),
            12 * NVL(e.comm, 0),
            12 * (e.sal + NVL(e.comm, 0))
                          AS "J-Gehalt"
FROM        emp e;
```

e)
```
SELECT * FROM emp WHERE mgr = NULL;
 no rows selected.
```

Die Abfrage ist falsch formuliert. Korrekt lautet sie:

```
SELECT e.* FROM emp e WHERE e.mgr IS NULL;
```

f)
```
SELECT      e.empno,
            e.ename,
            NVL(e.mgr, e.empno) "Chef"
FROM        emp e;
```

g)
```
SELECT      e.*
FROM        emp e
WHERE       e.mgr IS NULL
       AND  e.comm IS NULL;
```

1.4.5 Lösungen zu ORDER BY

a)
```
SELECT      e.*
FROM        emp e
ORDER BY e.sal DESC;
```

b)
```
SELECT DISTINCT
            e.job,
            e.deptno
FROM        emp e
ORDER BY    e.deptno;
```

c)
```
SELECT * FROM emp ORDER BY 1,2,3;
```
→ OK

```
SELECT empno, sal, sal FROM emp ORDER BY sal;
```
→ Falsch, da Spaltenname nicht eindeutig

(z. B.
```
SELECT    empno, sal, sal "Gehalt"
FROM      emp
ORDER BY "Gehalt";)
```

```
SELECT empno, ename "Name", sal "Gehalt"
FROM emp ORDER BY Gehalt;
```
→ Falsch (z. B. Gehalt auch im ORDER BY mit "" angeben)

```
SELECT ename, sal, empno FROM emp ORDER BY "SAL";
```
→ OK

1.5 Lösungen zu Funktionen

1.5.1 Lösungen zu arithmetischen Funktionen

a)
```
SELECT
     e.empno,
     e.ename,
     e.sal,
     ROUND(e.sal * 1.05)          AS "SAL Neu"
FROM emp e;
```

b)
```
SELECT
    e.empno,
    e.ename,
    e.comm,
    ROUND(e.comm - 500)           AS "Prov Neu",
    SIGN(ROUND(e.comm - 500))     AS "Prov Saldo"
FROM   emp e
WHERE  e.comm IS NOT NULL;
```

1.5.2 Lösungen zu Zeichenkettenfunktionen

a)
```
SELECT * FROM emp WHERE LENGTH(ename) = 5;
```

b)
```
SELECT * FROM emp WHERE ename LIKE 'A%';
```

oder

```
SELECT * FROM emp WHERE SUBSTR(ename,1,1) = 'A';
```

c)
```
SELECT empno, ename, REPLACE(ename,'A','Ä') "Name"
FROM emp;
```

d)
```
SELECT * FROM emp WHERE INSTR(ename, 'T') > 1;
```

e)
```
SELECT      e.empno, e.ename,
            REPLACE(e.ename, 'T', 'TT') "Name"
  FROM emp e
  WHERE
                e.ename LIKE '%T%'
            AND  LENGTH(e.ename) = 5;
```

f)
```
SELECT e.empno,
       LENGTH(e.ename || e.job)      "Länge",
       e.ename || e.job              "Name+Job"
FROM emp e;
```

g)
```
SELECT empno,
       ename,
       SUBSTR(ename,
              INSTR(ename, 'A'),
              LENGTH(ename) - INSTR(ename, 'A') +1)
                                        AS "Name"
  FROM emp;
```

h)
```
SELECT empno,
       ename,
       sal,
       SUBSTR('==============',1,ROUND(sal/500))
       || '>' || sal AS "Gehalt"
  FROM emp;
```

oder

```
COL gehalt FOR a30;

SELECT e.empno,
       e.ename,
       e.sal,
       LPAD('>' || e.sal,
            ROUND(e.sal/500) + LENGTH(sal) + 1,
            '=')
                                  AS "Gehalt"
FROM emp e;
```

1.5.3 Lösungen zu Konvertierungsfunktionen

a)
```
SELECT      e.empno,
    e.ename,
    e.job,
    TO_CHAR(e.hiredate, 'DD.MM(MON-MONTH).YYYY')
                        AS "Anstellungsdatum"
  FROM        emp e
  WHERE       e.job LIKE 'MANAGER'
         OR   e.job LIKE 'PRESIDENT'
  ORDER BY    e.ename;
```

b)
```
SELECT      e.empno,
            e.ename,
            e.hiredate,
            TO_CHAR(e.hiredate, 'WW') AS "KW"
  FROM        emp e
  WHERE       e.job LIKE 'MANAGER'
         OR   e.job LIKE 'PRESIDENT'
  ORDER BY    e.ename;
```

c)
```
COL "Monatswoche" FOR A25;
COL "Wochentag" FOR A9;

SELECT      e.empno,
            e.ename,
            e.hiredate,
            TO_CHAR(e.hiredate, 'WW') AS "KW",
  TO_CHAR(e.hiredate, 'W')  || '-te Woche im ' ||
  TO_CHAR(e.hiredate, 'MONTH') AS "Monatswoche",
            TO_CHAR(e.hiredate, 'DAY') AS "Tag",
            TO_CHAR(e.hiredate, 'D') AS "Wochentag"
  FROM   emp e
  ORDER BY "Wochentag";
```

d)
```
SELECT      e.empno,
            e.ename,
            e.hiredate,
            MONTHS_BETWEEN(SYSDATE, e.hiredate)
                              AS "Monate in FA"
  FROM        emp e
  ORDER BY    e.ename;
```

e)
```
COL "Quartal" FOR A7;

  SELECT
         e.empno,
         e.ename,
         e.hiredate,
         TO_CHAR(e.hiredate,'Q') "Quartal",
         DECODE(TO_CHAR(e.hiredate,'Q'), 1, 'erstes',
                                         2, 'zweites',
                                         3, 'drittes',
                                         4, 'viertes',
                                            'k.A.'
             ) || ' Quartal' AS "Quartalwort"
  FROM   emp e
  ORDER BY    e.ename;
```

f)
```
SELECT
  e.empno,
  e.ename,
  e.sal,
  TO_CHAR(DECODE(e.job, 'MANAGER', e.sal * 1.2,
                        'PRESIDENT', e.sal * 0.9,
                                     e.sal * 1.15
                ), '9999.99'
          ) AS "SALNeu"
FROM   emp e
ORDER BY e.ename;
```

1.5.4 Lösungen zu Gruppenfunktionen

a)
```
SELECT      AVG(e.sal) "Mittel",
            MAX(e.sal) "Max",
            MIN(e.sal) "Min"
FROM        emp e;
```

b)
```
SELECT      e.deptno,
            AVG(e.sal) "Durchschnitt"
FROM        emp e
GROUP BY    e.deptno;
```

c)
```
SELECT      e.job,
            MAX(e.sal) "Max Gehalt"
FROM        emp e
GROUP BY    e.job;
```

d)
```
SELECT      e.deptno,
            e.job,
            COUNT(e.job) "Anzahl"
FROM        emp e
WHERE       e.job = 'CLERK'
GROUP BY    e.deptno, e.job
HAVING      COUNT(e.job) > 1;
```

e)
```
SELECT      e.job,
            COUNT(*) "Anzahl"
FROM        emp e
GROUP BY    e.job
HAVING      COUNT(*) > 1;
```

f)
```
SELECT * FROM emp WHERE sal = MAX(sal);
```
→ Falsch, da Gruppenfunktionen in `WHERE`-Klausel nicht zulässig
z. B.
```
SELECT * FROM emp
    WHERE sal = (SELECT MAX(sal) FROM emp);
```

```
SELECT ename, job FROM emp GROUP BY job;
```
→ Falsch
z. B.
```
SELECT ename, job FROM emp ORDER BY job;
```

```
SELECT deptno, MAX(job) FROM emp GROUP BY deptno;
```
→ OK (aber Sinn ????)

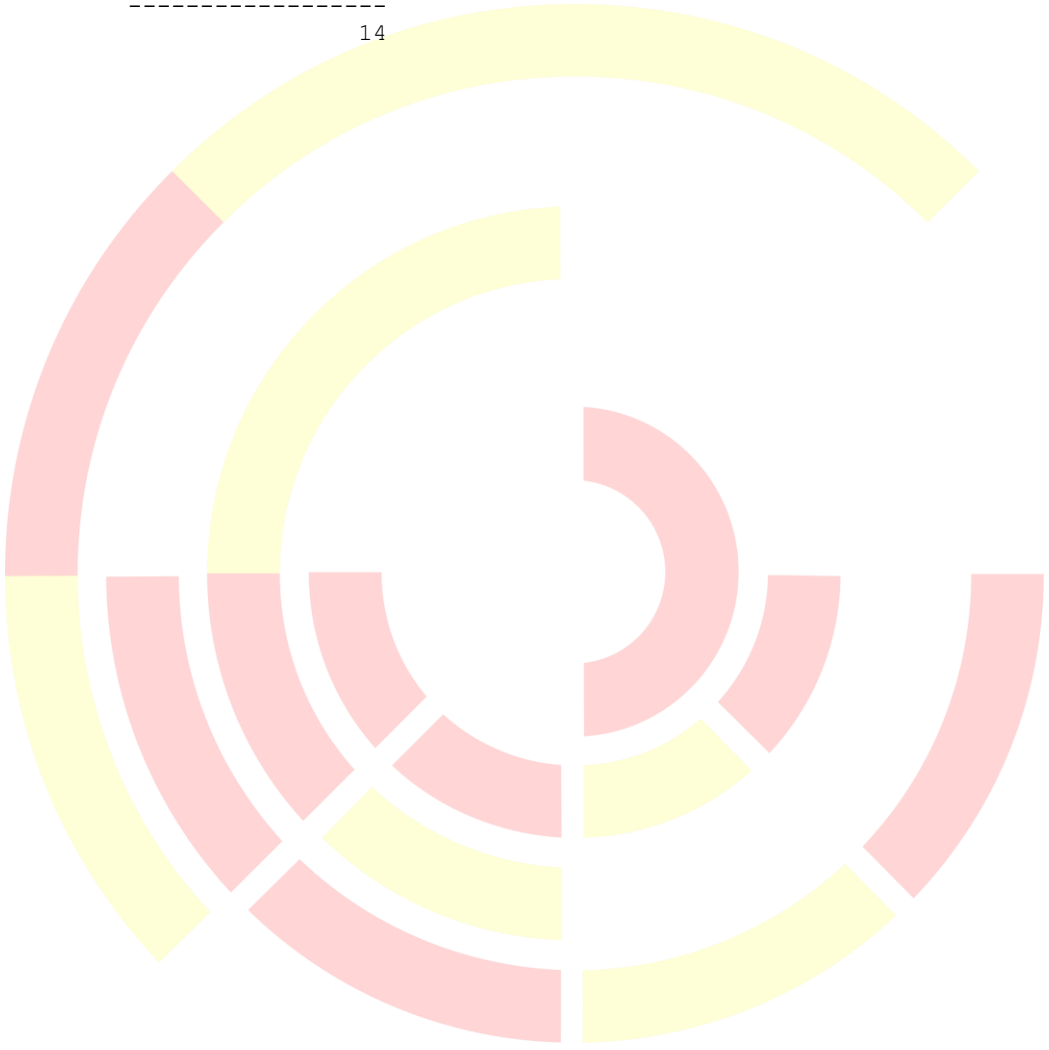
g) Ja, denn der COUNT-Befehl zählt nur Spaltenwerte, die nicht `NULL` sind.

```
SELECT COUNT(comm) FROM emp;

COUNT(COMM)
-----------
          4


SELECT COUNT(NVL(comm, 0)) FROM emp;

COUNT(NVL(COMM,0))
------------------
                14
```



1.6 Lösungen zu SELECTS mit mehreren Tabellen

1.6.1 Lösungen zu Joins

a) Es wird das kartesische Produkt der beiden Tabellen dargestellt mit 4*14= 56 Zeilen.

b)
```
SELECT empno, ename, sal, emp.deptno, dname
FROM emp, dept
WHERE emp.deptno = dept.deptno;
```

oder

```
SELECT e.empno, e.ename, e.sal, e.deptno, d.dname
FROM  emp e INNER JOIN dept d
      ON e.deptno = d.deptno;
```

oder

```
SELECT e.empno, e.ename, e.sal, e.deptno, d.dname
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

c)
```
SELECT emp.*, dept.*
FROM emp, dept
WHERE emp.deptno = dept.deptno;
```

oder

```
SELECT e.*, d.*
FROM  emp e INNER JOIN dept d
      ON e.deptno = d.deptno;
```

d)
```
SELECT empno, ename, sal, grade
FROM emp, salgrade
WHERE sal >= losal AND sal <= hisal;
```

oder

```
SELECT e.empno, e.ename, e.sal, s.grade
FROM emp e INNER JOIN salgrade s
      ON E.sal BETWEEN s.losal AND s.hisal;
```

e)
```
SELECT e1.empno "MAempno", e1.ename "MAName",
       e1.sal "MAGehalt", e2.empno "Chefempno",
       e2.ename "ChefName", e2.sal "ChefGehalt"
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno;
```

oder

```
SELECT e1.empno "MAempno", e1.ename "MAName",
       e1.sal "MAGehalt", e2.empno "Chefempno",
       e2.ename "ChefName", e2.sal "ChefGehalt"
FROM  emp e1 INNER JOIN emp e2
       ON e1.mgr = e2.empno;
```

f) Es werden nur 13 Zeilen ausgegeben, da Herr King keinen Vorgesetzten hat.

g)
```
SELECT e1.empno "MAempno", e1.ename "MAName",
       e1.sal "MAGehalt", e2.empno "Chefempno",
       e2.ename "ChefName", e2.sal "ChefGehalt"
FROM emp e1, emp e2
WHERE e1.mgr = e2.empno(+);
```

oder

```
SELECT e1.empno "MAempno", e1.ename "MAName",
       e1.sal "MAGehalt", e2.empno "Chefempno",
       e2.ename "ChefName", e2.sal "ChefGehalt"
FROM  emp e1 LEFT OUTER JOIN emp e2
       ON e1.mgr = e2.empno;
```

h)
```
SELECT empno, ename, sal, grade,
       dept.deptno, dname
FROM emp, salgrade, dept
WHERE sal BETWEEN losal AND hisal
  AND emp.deptno = dept.deptno;
```

oder

```
SELECT e.empno, e.ename, e.sal, s.grade,
       d.deptno, d.dname
FROM  emp e INNER JOIN dept d
       ON e.deptno = d.deptno
       INNER JOIN  salgrade s
       ON e.sal BETWEEN         s.losal
                        AND  s.hisal;
```

1.6.2 Lösungen zu Unterabfragen

a)
```
SELECT     e.empno, e.ename, e.sal
FROM       emp e
WHERE e.deptno IN    (
                     SELECT    e2.deptno
                     FROM      emp e2
                     WHERE e2.ename = 'SMITH')
AND e.ename != 'SMITH';
```

b)
```
SELECT     e.empno, e.ename, e.sal
FROM       emp e
WHERE e.sal <   (
                SELECT    AVG(e2.sal)
                FROM      emp e2)
ORDER BY e.sal, e.empno;
```

c)
```
SELECT     e.empno, e.ename, e.deptno
FROM       emp e
WHERE e.deptno IN    (
                     SELECT    e2.deptno
                     FROM      emp e2
                     WHERE e2.ename LIKE 'A%');
```

d)
```
SELECT     e.empno, e.deptno, e.ename, e.job
FROM       emp e
WHERE e.deptno IN    (
                     SELECT    d.deptno
                     FROM      dept d
                     WHERE d.dname = 'RESEARCH');
```

e)
```
SELECT     e.empno, e.ename, e.hiredate
FROM       emp e
WHERE e.deptno IN    (
                     SELECT    d.deptno
                     FROM      dept d
                     WHERE d.loc IN
                          ('NEW YORK', 'DALLAS')
                     );
```

f)
```
-- IN
SELECT      e.empno, e.ename,
            e.deptno, e.sal, e.comm, e.mgr
FROM        emp e
WHERE (e.deptno, e.mgr) IN (
                        SELECT    e2.deptno, e2.mgr
                        FROM      emp e2
                        WHERE     e2.comm > 0
                        )
ORDER BY e.ename, e.empno;
```

g)
```
-- JOIN
SELECT      e.empno, e.ename,
            e.deptno, e.sal, e.comm, e.mgr
FROM  emp e INNER JOIN
        (   SELECT DISTINCT e2.deptno, e2.mgr
            FROM      emp e2
            WHERE     e2.comm > 0
        ) e3
ON          e.deptno = e3.deptno
      AND   e.mgr = e3.mgr
ORDER BY e.ename, e.empno;

-- EXISTS
SELECT      e.empno, e.ename,
            e.deptno, e.sal, e.comm, e.mgr
FROM        emp e
WHERE EXISTS
        (   SELECT    1
            FROM      emp e2
            WHERE
                      e2.comm > 0
                AND   e.deptno = e2.deptno
                AND   e.mgr = e2.mgr)
ORDER BY e.ename, e.empno;
```

Lösung zu komplexer Übung zu Funktionen, JOIN und Unterabfragen

a)
```
SELECT
    d.deptno            AS abtnr,
    grE.SummeV          AS "SUMMEV_proAbt",
    grE.DV              AS "Durchschnitt",
    grE.MinV            AS "MinimalV",
    grE.MaxV            AS "MaximalV",
    grE.AnzV            AS "AnzahlMA"
FROM
        dept d
        LEFT OUTER JOIN
        (SELECT
                deptno,
                SUM(sal)        AS SummeV,
                AVG(sal)        AS DV,
                MAX(sal)        AS MaxV,
                MIN(sal)        AS MinV,
                COUNT(sal)      AS AnzV
        FROM
                emp
        GROUP BY
                deptno
            ) grE
        ON d.deptno = grE.deptno
;
```

b)
```
COL ABTEILUNG FOR A10;
COL StufeAVG FOR A8;
COL StufeMin FOR A8;
COL StufeMax FOR A8;
COL "MinV" FOR 9999;
COL "MaxV" FOR 9999;
COL "AnzMA" FOR 9999;

SELECT
  d.deptno                  AS ABTNR,
  grE.SummeV                AS "SUM_proAbt",
  grE.DV                    AS "Durchschnitt",
  DECODE( s3.Grade,   1, 'erste', 2, 'zweite',
                      3, 'dritte', 4, 'vierte',
                      5, 'fuenfte', 'k.A.')
                            AS "StufeAVG",
  grE.MinV                  AS "MinV",
  DECODE( s2.Grade,   1, 'erste', 2, 'zweite',
                      3, 'dritte',4, 'vierte',
                      5, 'fuenfte', 'k.A.')
                            AS "StufeMin",
  grE.MaxV                  AS "MaxV",
  DECODE( s1.Grade,   1, 'erste', 2, 'zweite',
                      3, 'dritte',4, 'vierte',
                      5, 'fuenfte', 'k.A.')
                            AS "StufeMax",
  grE.AnzV                  AS "AnzMA"
FROM    dept d
LEFT OUTER JOIN
           (SELECT
           deptno,
           SUM(sal)         AS SummeV,
           AVG(sal)         AS DV,
           MAX(sal)         AS MaxV,
           MIN(sal)         AS MinV,
           COUNT(sal)       AS AnzV
       FROM      emp
       GROUP BY  deptno
       ) grE
ON     d.deptno = grE.deptno
LEFT OUTER       JOIN      salgrade s1
ON     grE.MAxV       between s1.losal AND s1.hisal
LEFT OUTER       JOIN      salgrade s2
ON     grE.MinV       between s2.losal AND s2.hisal
LEFT OUTER       JOIN      salgrade s3
ON     grE.DV         between s3.losal AND s3.hisal
;
```

1.6.3 Lösungen zu SET Operatoren

a)
```
SELECT * FROM emp WHERE job = 'CLERK'
UNION ALL
SELECT * FROM emp WHERE deptno = 10;
```

b)
```
SELECT * FROM emp WHERE job = 'CLERK'
UNION
SELECT * FROM emp WHERE deptno = 10;
```

c)
```
SELECT * FROM emp WHERE job = 'CLERK'
INTERSECT
SELECT * FROM emp WHERE deptno = 10;
```

d)
```
SELECT * from emp where deptno IN (10, 30)
MINUS
SELECT * FROM emp where job NOT IN ('MANAGER',
'PRESIDENT');
```

1.7 Lösungen zu DML

1.7.1 Lösungen zu INSERT/UPDATE/DELETE

a)
```
INSERT INTO emp
  (empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES
  (8000,'OTTO','CLERK',7369,'01-JAN-99',6000,
   NULL,10);
```

b) `INSERT INTO <tabelle> VALUES ('Wert',....);`

oder

```
INSERT INTO <tabelle> (spalte1,...)
VALUES ('Wert',....);
```

Die zweite Methode ist vorzuziehen, da sie bei einer großen Anzahl von Spalten übersichtlicher ist und beim Hinzufügen von neuen Spalten (`ALTER TABLE ADD COLUMN...`) nicht zum Fehler führt.

c)
```
UPDATE emp
SET sal = 1000, deptno = 40
WHERE empno = 8000;
```

d) Der `UPDATE` wird bei allen Zeilen der Tabelle durchgeführt, da keine `WHERE`-Klausel angegeben wurde.

e) Es werden alle Zeilen gelöscht. Sollten Sie es ausprobiert haben → `ROLLBACK`; !

f) `update emp set sal=100 where deptno=90;`
→ OK, aber es gibt keine Abteilung 90
→ kein Datensatz durch `UPDATE` betroffen

`update emp, dept set deptno=10;`
→ Falsch, da `UPDATE` nur auf *eine* Tabelle geht

`update emp set sal = sal;`
→ OK (keine Veränderung des Gehalts)

`update emp set sal =(select max(sal) from emp);`
→ OK (jeder bekommt EUR 5000,-)

`delete from emp where 1 = 1;`
→ True für alle Zeilen → alle gelöscht !

1.7.2 Lösungen zur referenziellen Integrität

Die Lösungen hier hängen von der Installation ihres Systems ab.

Finden Sie heraus, ob die FKs von der Tabelle emp auf dept mit:

`ON DELETE CASCADE` deklariert sind.

a) Geht nicht. Deptno = 20 existiert bereits.

```
UPDATE dept SET deptno = 20 WHERE deptno = 10;
```

b) Geht nicht, da die deptno 10 aus der Tabelle emp referenziert wird.

```
UPDATE dept SET deptno = 70 WHERE deptno = 10;
```

c) Geht, falls Sie nicht in der Abteilung 40 beschäftigt sind, s. Aufgabe 1.7.1 c).

```
UPDATE dept SET deptno = 70 WHERE deptno = 40;
```

d) Geht nicht. Es gibt Mitarbeiter in der Tabelle emp, die in den Department SALES beschäftigt sind.

```
DELETE FROM dept WHERE deptno = 30;
```

e) Geht nicht. Die deptno 10 ist besetzt.

```
INSERT INTO dept VALUES (10, 'EU Zentrale', 'BERLIN');
```

f) Geht. Die deptno 90 ist frei.

```
INSERT INTO dept VALUES (90, 'EU Zentrale', 'HAMBURG');
```

g) Geht nicht. Die deptno 10 ist besetzt.

```
INSERT INTO emp VALUES (...);
```

Es ist Ihre Entscheidung, wen Sie in der Zentrale zu welchen Konditionen beschäftigen. Sie sind nur sich und Herrn King verantwortlich. Herr King könnte aber allergisch reagieren, wenn Sie z. B. Ihre Mitarbeiter überbezahlen.

```
INSERT INTO dept VALUES (...);
```

h) Geht. (Aber nicht vergessen: einige Mitarbeiter haben KING als Vorgesetzten)

```
DELETE FROM emp WHERE empno = 7839;
```

i) Geht (falls niemand in dieser Abteilung zwischenzeitlich beschäftigt wurde s. Aufgabe 1.7.1 c)).

```
DELETE FROM dept WHERE deptno = 40;
```

1.8 Lösungen zu DDL Befehlen

1.8.1 Lösungen zur Tabellenerstellung

a)
```
CREATE TABLE mitarbeiter (
  ma_nr          NUMBER(4),
  m_name         VARCHAR2(10),
  beruf          VARCHAR2 (9),
  chef_nr        NUMBER(4),
  einstellungsdatum DATE,
  gehalt         NUMBER(7,2),
  provision      NUMBER(7,2),
  abt_nr         NUMBER(2)
);
```

b)
```
CREATE TABLE test AS SELECT * FROM emp;
```

oder

```
CREATE TABLE mitarbeiter (
      ma_nr,  m_name,  beruf ,  chef_nr ,
      einstellungsdatum,  gehalt ,  provision,
      abt_nr)
AS    SELECT empno, ename, job, mgr,
           hiredate, sal, comm, deptno
      FROM emp
      WHERE 1=2;
```

c)
```
ALTER TABLE mitarbeiter
ADD (bemerkung VARCHAR2(30));
```

d)
```
ALTER TABLE mitarbeiter DROP COLUMN bemerkung;
```

e) (*)
```
CREATE TABLE test AS SELECT * FROM emp;

INSERT INTO test SELECT * FROM test;
```

→ ca. 10 mal '/' in SQL*Plus betätigen, um den Befehl wiederholt auszuführen.

oder

```
CREATE TABLE test AS
SELECT e.* FROM emp e, emp, emp;
```

f) (**)
```
ALTER TABLE test ADD (nr number);

UPDATE test SET nr = ROWNUM;

ALTER TABLE test
ADD CONSTRAINT test_nr_pk PRIMARY KEY (nr);
```

1.8.2 Lösungen zu Constraints

a)
```
CREATE TABLE firma (
  f_nr          NUMBER(4) PRIMARY KEY,
  name          VARCHAR2(30),
  adresse       VARCHAR2(30));

CREATE TABLE person (
  f_nr          NUMBER(4) CONSTRAINT person_fk
                          REFERENCES  firma(f_nr),
  p_nr          NUMBER(4),
  name          VARCHAR2(30),
  abteilungs_nr NUMBER(4),
  CONSTRAINT person_pk PRIMARY KEY (f_nr,p_nr));

CREATE TABLE notiz (
  f_nr          NUMBER(4),
  p_nr          NUMBER(4),
  k_nr          NUMBER(4),
  notiz_text    VARCHAR2(200),
  notiz_status  VARCHAR2(20) CONSTRAINT notiz_ck
                CHECK (notiz_status IN
                ('OFFEN','ERLEDIGT','TERMIN')),
  datum   DATE DEFAULT SYSDATE,
  CONSTRAINT notiz_pk
              PRIMARY KEY (f_nr, p_nr, k_nr),
  CONSTRAINT notiz_fk FOREIGN KEY (f_nr, p_nr)
              REFERENCES person (f_nr, p_nr));
```

b)
```
INSERT INTO firma VALUES
      (1,'Integrata Training AG','München');
INSERT INTO firma VALUES
      (2,'UITAG','Stuttgart');
INSERT INTO firma VALUES
      (3,'ALTER data GmbH','Tübingen');

INSERT INTO person VALUES (1,1,'Herr Huber',10);
INSERT INTO person VALUES (1,2,'Herr Maier',20);
INSERT INTO person VALUES (2,1,'Herr Schmidt',10);
INSERT INTO person VALUES (3,1,'Herr Bejic',10);

INSERT INTO notiz VALUES
(1,1,1,'Hat angerufen','OFFEN','01-JAN-99');
INSERT INTO notiz VALUES
(1,1,2,'Termin vereinbart','ERLEDIGT', '01-MAI-99');
```

c) Diese Daten verstoßen gegen die Integritätsregel und werden nicht angenommen. Fehler:
```
ORA-00001: unique constraint (SCOTT.SYS_C00...)
violated
```

d) Nein. Fehler: `ORA-02292: integrity constraint (SCOTT.PERSON_FK) violated - child record found`

Durch die Verknüpfung der Tabellen über eine Primary/Foreign Key Beziehung kümmert sich die Datenbank um die Konsistenz der Daten. Wenn Sie die Firmen löschen könnten, die noch abhängige Datensätze haben, würden diese später alleine bestehen (Mitarbeiter einer Firma ohne Ihre Firma). Deshalb ist ein Löschen nicht möglich.

e)
```
ALTER TABLE person DROP CONSTRAINT person_fk;
INSERT INTO person VALUES (4,1,'Herr Habicht',10);
ALTER TABLE person ADD CONSTRAINT person_fk
      FOREIGN KEY (f_nr) REFERENCES firma (f_nr);
```

oder

```
ALTER TABLE person DISABLE bzw. ENABLE CONSTRAINT
person_fk;
```

f) Ausführen des Skripts: `UTLEXCPT.SQL` im `%ORACLE_HOME%\RDBMS\ADMIN`-Verzeichnis oder

```
CREATE TABLE exceptions(
  row_id     rowid,
  owner      varchar2(30),
  table_name varchar2(30),
  constraint varchar2(30));

ALTER TABLE PERSON
DROP CONSTRAINT PERSON_FK;

INSERT INTO PERSON
VALUES (4, 1, 'HERR HABICHT', 10);

-- FK anlegen
ALTER TABLE PERSON
ADD CONSTRAINT PERSON_FK
      FOREIGN KEY (F_NR)
      REFERENCES FIRMA (F_NR)
      EXCEPTIONS INTO EXCEPTIONS;

FEHLER in Zeile 2:
ORA-02298: (SCOTT.PERSON_FK) kann nicht validiert
werden - übergeordnete Schlüssel nicht gefunden
```

-- nach der Fehlermeldung feststellen, welches DS Probleme macht:

```
SELECT
     p.*,
     e.*
FROM      person p INNER JOIN exceptions e
          ON p.rowid = e.row_id;

-- dieses manuell reparieren!
DELETE
FROM      person
WHERE     f_nr = 4;

ALTER TABLE person
ADD CONSTRAINT person_fk
     FOREIGN KEY (f_nr)
     REFERENCES firma (f_nr)
     EXCEPTIONS INTO exceptions;


-- oder Fehler dulden:
ALTER TABLE person
ADD CONSTRAINT person_fk
     FOREIGN KEY (f_nr)
     REFERENCES firma (f_nr) NOVALIDATE;
```

(aber Constraint-Verletzung bleibt, sie wird nur akzeptiert)

Möglichkeit 2:
mit dulden der Fehler und enablen und disablen des FK:

```
ALTER TABLE person
DISABLE CONSTRAINT person_fk;

INSERT INTO person
VALUES (4, 1, 'Herr Habicht', 10);

ALTER TABLE person
ENABLE NOVALIDATE CONSTRAINT person_fk;
```

(aber Constraint-Verletzung bleibt, sie wird nur akzeptiert)

1.8.3 Lösungen zu Views

a)
```
CREATE VIEW emp_view AS
SELECT * FROM emp;
```

b)
```
CREATE VIEW emp_view10 AS
SELECT empno, ename,
       job, mgr, hiredate, deptno
FROM emp
WHERE deptno=10;

CREATE VIEW emp_view20 AS
SELECT empno, ename,
       job, mgr, hiredate, deptno
FROM emp
WHERE deptno=20;

CREATE VIEW emp_view30 AS
SELECT empno, ename,
       job, mgr, hiredate, deptno
FROM emp
WHERE deptno=30;
```

c)
```
CREATE VIEW manager_view AS
SELECT empno, ename,
       job, mgr, hiredate,
       DECODE(ename, 'KING',
              NULL, sal) AS sal,
       comm, deptno
FROM emp;
```

d)
```
CREATE FORCE VIEW xyz_view AS SELECT * FROM xyz;
```
→ Wenn Force weggelassen wird, ist Statement nicht möglich.
```
SELECT * FROM xyz_view;
→ ORA-04063: view "SCOTT.XYZ_VIEW" has errors
CREATE TABLE xyz AS SELECT * FROM emp;
SELECT * FROM xyz_view; → OK
```

e)
```
CREATE VIEW emp_dept AS
  SELECT e.*,dname FROM emp e, dept d
  WHERE e.deptno=d.deptno;
```

f)
```
CREATE VIEW emp_dept_salgrade AS
  SELECT t1.*,t2.grade
  FROM emp_dept t1,salgrade t2
  WHERE t1.sal BETWEEN t2.losal AND t2.hisal;
```

1.8.4 Lösungen zu Indizes

a) **Vor Indizierung:**

```
SELECT * FROM big_emp WHERE nr = 8000;

EMPNO  ENAME   JOB       MGR  HIREDATE     SAL  COMM  DEPTNO    NR
-----  ------  -------  ----  ---------  -----  ----  ------  ----
 7698  BLAKE   MANAGER  7839  01-MAY-81   2850            30  8000

real: 180

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'BIG_EMP'
```

```
CREATE UNIQUE INDEX big_emp_nr_ind ON big_emp (nr);
```

Nach Indizierung:

```
SELECT * FROM big_emp WHERE nr=8000;

EMPNO  ENAME   JOB       MGR  HIREDATE     SAL  COMM  DEPTNO    NR
-----  ------  -------  ----  ---------  -----  ----  ------  ----
 7698  BLAKE   MANAGER  7839  01-MAY-81   2850            30  8000

real: 70

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY ROWID) OF 'BIG_EMP'
   2    1     INDEX (UNIQUE SCAN) OF 'EMP_NR_PK' (UNIQUE)
```

→ Index wird benutzt.

b) Weil das Parsen entfällt. (Wenn Statement bereits geparst in der SGA vorliegt)

1.9 Lösungen zur Benutzerverwaltung

Der Referent muss ggf. allen TN die fehlenden (einzelnen) Berechtigungen hierfür erteilen (ggf. als User `SYS`):

```
GRANT CREATE USER, CREATE SESSION, CREATE TABLE, CRE-
ATE ROLE, CREATE SEQUENCE, CREATE VIEW TO PUBLIC WITH
ADMIN OPTION;
```

a) Der Benutzer muss einen eindeutigen Namen und ein Passwort bekommen.

Man sollte ihm beim Einrichten gleich einen Default & Temporary Tablespace einrichten.

Er sollte eine Quota auf die entsprechenden Tablespaces bekommen.

Er sollte die notwendigen Rechte/Rollen erhalten (auf jeden Fall benötigt er das `CREATE SESSION` Recht).

Evtl. sollte noch ein Profil für ihn eingerichtet werden.

Evtl. können noch AUDIT Option für den Benutzer eingestellt werden.

b)
```
SELECT username FROM all_users;
```

c)
```
SELECT * FROM session_privs;
```

d) Er darf dieses Recht in JEDEM Schema ausüben (z. B. Create ANY Table-Recht bedeutet, dass der Benutzer in jedem Schema eine Tabelle anlegen kann.)

e)
```
SELECT table_name FROM user_tables;
```

f)
```
GRANT select ON emp TO nachbar;
SELECT * FROM nachbar.emp;
```

g)
```
GRANT select ON emp TO public;
```

h) (*) Sie legen eine View auf die Tabelle EMP an:

```
CREATE VIEW empv AS
  SELECT empno, ename, job, mgr, hiredate, deptno
  FROM emp
  WHERE empno<>7839;
```

i)
```
CREATE USER mario IDENTIFIED BY super
DEFAULT TABLESPACE user_data
TEMPORARY TABLESPACE temporary_data
QUOTA 100K ON user_data
QUOTA 100K ON temporary_data;
```

j)
```
GRANT create session, create table TO mario;
```

k)
```
CREATE VIEW mariovu AS
  SELECT * FROM emp
  WHERE deptno IN (10,20)
WITH CHECK OPTION;

GRANT insert, update ON mariovu TO mario;
```

l)
```
CREATE USER Verona IDENTIFIED BY Feldbusch
DEFAULT TABLESPACE user_data
TEMPORARY TABLESPACE temporary_data
QUOTA 100K ON user_data
QUOTA 100K ON temporary_data;

CREATE USER Naddel IDENTIFIED BY Dieter
DEFAULT TABLESPACE user_data
TEMPORARY TABLESPACE temporary_data
QUOTA 100K ON user_data
QUOTA 100K ON temporary_data;

CREATE ROLE benutzer;

GRANT create session, create table,
      create sequence, create view
TO benutzer;

GRANT benutzer TO Verona, Naddel;

REVOKE insert, update ON mariovu FROM mario;
```