Sprachen für das Relationenmodell: SQL

Sprachen für das Relationenmodell (1)

Sprachtypen nach Beschreibungsart:

- Prozedurale Sprachen:
  operational; Beschreibung des Prozesses, der die Antwortmenge erzeugt.
- Nicht-prozedurale Sprachen:
  deskriptiv; Beschreibung der Bedingung, die erfüllt sein muß, damit ein Datenelement zur Antwortmenge gehört

Sprachen für das Relationenmodell (2)

- Tupelorientierte Sprachen:
  Operationen liefern einzelne Tupel/Sätze als Resultat zurück, werden mehrere Tupel/Sätze als Resultat gewünscht, so müssen sie explizit sequentiell abgearbeitet werden
- Mengenorientierte Sprachen:
  Operationen liefern Mengen von Tupeln zurück
  Bsp: SQL

Cursor-Konzept: Umsetzung von mengenorientierter in tupelorientierte Darstellung

SQL: Übersicht

1. Allgemeines und Datentypen
2. Abfragen
3. Datendefinition
4. Datenmanipulation
5. Transaktionen
6. SQL-3-Standard

1. Allgemeines

- Sprache für Datendefinition, Datenmanipulation, Datenschutz
- nichtprozedural, mengenorientiert
- basierend auf Relationenmodell
- Definition der Sprache durch Standards (SQL-86, SQL-89, SQL-92, dzt. in Arbeit: SQL-3
- Beteiligt an Standardisierung: ANSI X3H2, ISO TC97/SC21

Datentypen

- **Zeichenketten:**
  CHAR (n), VARCHAR (n), LONG
- **Zahlen:**
  NUMBER (n), NUMBER (p,s), INTEGER = NUMBER (38), DECIMAL = NUMBER
- **Datum + Zeit:**
  DATE

Zeichenketten

CHAR (n) .... Zeichenkette mit fixer Länge n (n<=255)

VARCHAR (n) .... Zeichenkette variabler Länge (n <= 255)

LONG .... Zeichenketten bis 65536 Zeichen

Zeichenketten werden mit einfachen Anführungsstrichen begrenzt - einfache Anführungszeichen als Zeichen werden doppelt geschrieben.
Bsp: ‘Hallo’, ‘John”s car’

Zeichenkettenfunktionen

Name	Beschreibung
INITCAP (c)	Das erste Zeichen der Zeichenkette c wird groß geschrieben
LOWER (c)	Alle Zeichen von c werden klein geschrieben
UPPER (c)	Alle Zeichen von c werden groß geschrieben
INSTR (c1,c2)	Position des 1. Auftretens von c2 in c1
SUBSTR (c,m,n)	Substringfunktion: liefert von der Zeichenkette c n Zeichen ab Position m
||	Zeichenketten-Konkatenation
LIKE	Mustervergleich

Zahlen

NUMBER .... Zahl mit max. 38 Stellen
(auch DECIMAL)
NUMBER (n)............. ganze Zahl mit n Stellen
NUMER (p,s) .... Zahl mit insgesamt max. p Stellen, davon s Stellen hinter dem Komma
INTEGER.................. gleichbedeutend mit NUMBER (38)

Zahlenformat:
- Ziffernfolge evtl. mit Dezimalpunkt und/oder Exponent
- Ganze Zahl mit ‘M’ oder ‘K’

Bsp.: 12.345, .678, 356, 1K (=1024), 2M (=2*1048576), 123.45e-03, 18E5

Numerische Operatoren

() ....Klammerung hat Priorität

+, - ....unärer Operator Vorzeichen
*, / ....Multiplikation, Division
+, - ....Addition, Subtraktion

Zahlenfunktionen

<u>Name ....Beschreibung</u>
ABS(n) ....Gibt den Absolutwert der Zahl n zurück
MOD (m,n)........Modulodivision m MOD n
ROUND (m,n) ..Rundung von m auf n Stellen wenn für n kein Wert angegeben wird, gilt n=0
TRUNC (m,n)....m wird auf n Stellen abgerundet (entspricht einem Abschneiden der nachfolgenden Stellen)
SIGN (n) ....Signum-Funktion:
SIGN (n) = -1 falls n < 0,
SIGN(0)=0, SIGN(n)=1 falls n > 0
SQRT (n) ....Quadratwurzel von n

Vergleichsoperatoren

=.....................................Test auf Gleichheit
!=, ^= oder <> ....Test auf Ungleichheit
<, > ....Test auf Kleiner- bzw. Größer-Beziehung
<=, >= ....Test auf Kleiner-oder-gleich- bzw. Größer-oder-gleich-Beziehung

Diverse andere Funktionen

Name	Beschreibung
TO_CHAR	konvertiert NUMBER oder DATE in CHARACTER
TO_DATE	konvertiert CHARACTER in DATE
TO_NUMBER	konvertiert CHARACTER in NUMBER
SYSDATE	aktuelles Datum
USER	Name des Benutzers

2. Abfragen in SQL: Übersicht

2.1 Grundprinzip: Projektion und Selektion

2.2 Mengenoperationen

2.3 Logische Operatoren, Prädikate,
arithmetische Ausdrücke, Sortierung

2.4 kartesisches Produkt, Verbund

2.5 Aggregatfunktionen mit und ohne Gruppierung

2.6 Teilabfragen

2.1 Abfragen: Prinzip

Grundkonstruktion: “Mapping”

Selektion gefolgt von einer Projektion

SELECT <Attributliste>

FROM <Relationen>

WHERE <Bedingungen>

Beispieldatenbank

Relation emp

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Relation dept

DEPTNO	DNAME	LOC
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
40	OPERATIONS	BOSTON

Projektion

Welche Abteilungen gibt es (Nummer und Name)?

```
SELECT dname, deptno
FROM   dept;

DNAME          DEPTNO
---------- ----------
SALES              30
ACCOUNTING         10
OPERATIONS         40
```


Projektion ohne Duplikat-Eliminierung

Gesucht sind die Jobs aller Personen, auszugeben sind nur die Berufsbezeichnungen.

```
SELECT job
FROM   emp;

JOB
---------
SALESMAN
SALESMAN
SALESMAN
MANAGER
MANAGER
PRESIDENT
SALESMAN
CLERK
CLERK
```


Projektion mit Duplikat-Eliminierung

Welche Jobs gibt es (jede Bezeichnung ist nur einmal auszugeben)?

```
SELECT DISTINCT job
FROM   emp;

JOB
---------
CLERK
MANAGER
PRESIDENT
SALESMAN
```


Selektion

Gesucht ist die Zeile für die Abteilung Nr. 30.

```
SELECT * FROM dept
         WHERE deptno=30;

    DEPTNO DNAME      LOC
---------- ---------- ----------
        30 SALES      CHICAGO
```

Projektion und Selektion

Gesucht ist Name und Nummer der Abteilung 30.

```
SELECT   dname, deptno
FROM     dept
WHERE    deptno=30;

DNAME          DEPTNO
---------- ----------
SALES              30
```


2.2 Mengenoperationen

Vereinigung	Durchschnitt	Mengendifferenz
`SELECT * FROM A`	`SELECT * FROM A`	`SELECT * FROM A`
`UNION`	`INTERSECT`	`MINUS`
`SELECT * FROM B;`	`SELECT * FROM B;`	`SELECT * FROM B;`

2.3 Boolesche Operatoren, Prädikate

Verknüpfung von logischen Ausdrücken:

NOT .... logisches NICHT
AND .... logisches UND
OR .... logisches ODER

x BETWEEN a AND b ... entspricht x>=a AND x<=b

IN .... Mengenoperator “ist Element von”

LIKE .... Mustervergleich für Zeichenketten

Konjunktion

Wer ist Manager und verdient mehr als 2800?

```
SELECT ename, job, sal
FROM  emp
WHERE job='MANAGER'
      AND sal > 2800;

ENAME  JOB              SAL
------ --------- ----------
BLAKE  MANAGER         2850
```

Disjunktion

Wer ist Manager oder verdient mehr als 2800?

```
SELECT ename, job, sal
FROM  emp
WHERE job='MANAGER'
      OR sal > 2800;

ENAME  JOB              SAL
------ --------- ----------
BLAKE  MANAGER         2850
CLARK  MANAGER         2450
KING   PRESIDENT       5000
```


BETWEEN

```
SELECT ename, sal
FROM   emp
WHERE  sal BETWEEN 1200 AND 1400;

ENAME         SAL
------ ----------
WARD         1250
MARTIN       1250
MILLER       1300
```


Mengenoperation IN

Gesucht werden die Abteilungen mit den Nummern 10 und 30.

```
SELECT *
FROM   dept
WHERE  deptno IN (10,30);

   DEPTNO DNAME      LOC
---------- ---------- ----------
        30 SALES      CHICAGO
        10 ACCOUNTING NEW YORK
```


Mustervergleich mit LIKE

- Vergleich mit Muster mit LIKE
- % steht für Zeichenkette mit beliebig vielen Zeichen
- _ steht für genau ein Zeichen

```
SELECT ename
FROM   emp
WHERE  ename LIKE '__R%';

ENAME
------
WARD
MARTIN
TURNER
```

Arithmetische Ausdrücke

Wieviel Gehalt und Provision bekommen die Verkäufer?

```
SELECT ename, sal, comm, sal + comm
FROM   emp
WHERE  job='SALESMAN';

ENAME           SAL       COMM   SAL+COMM
------ ---------- ---------- ----------
ALLEN          1600        300       1900
WARD           1250        500       1750
MARTIN         1250       1400       2650
TURNER         1500          0       1500
```


Sortierung nach einem Attribut

Abfrage der Personen in der Abteilung 30, ihrer Berufe und ihres Gehaltes, aufsteigend sortiert nach Gehalt

```
SELECT   sal, job, ename
FROM     emp
WHERE    deptno=30
ORDER BY sal;

       SAL JOB       ENAME
---------- --------- ------
       950 CLERK     JAMES
      1250 SALESMAN  WARD
      1250 SALESMAN  MARTIN
      1500 SALESMAN  TURNER
      1600 SALESMAN  ALLEN
      2850 MANAGER   BLAKE
```


Sortierung nach mehreren Spalten

Abfrage der Personen, ihrer Berufe und ihres Gehaltes, sortiert nach Job aufsteigend und Sal absteigend

```
SELECT   job, sal, ename
FROM     emp
ORDER BY job, sal desc;

JOB              SAL ENAME
--------- ---------- ------
CLERK           1300 MILLER
CLERK            950 JAMES
MANAGER         2850 BLAKE
MANAGER         2450 CLARK
PRESIDENT       5000 KING
SALESMAN        1600 ALLEN
SALESMAN        1500 TURNER
SALESMAN        1250 WARD
SALESMAN        1250 MARTIN
```


2.4 Verknüpfung von Tabellen

- kartesisches Produkt
- Gleichverbund
- Verbund zweier Kopien einer Tabelle

Kartesisches Produkt

Angabe von mehreren Tabellennamen in der FROM-Klausel

```
SELECT   *
FROM     emp, dept;

SELECT   ename, loc
FROM     emp, dept;
```


Gleichverbund

In welcher Stadt arbeitet der Mitarbeiter ALLEN?

```
SELECT   emp.ename, dept.loc
FROM     emp, dept
WHERE    ename='ALLEN'
AND      emp.deptno=dept.deptno;

ENAME  LOC
------ ----------
ALLEN  CHICAGO
```

Hinweis: In manchen Implementierungen genügt es, im SELECT-Teil nur die Attributnamen anzugeben, wenn der Tabellenname eindeutig ist und das entsprechende Attribut nur in einer Tabelle exisitiert.

Verbund zweier Kopien einer Tabelle

Unterscheidung der Kopien durch Verwendung von **Aliases**.

Gesucht sind alle Mitarbeiter, die mehr verdienen als die Hälfte des Gehalts ihres Managers.

```
SELECT e1.ename, e1.sal
FROM   emp e1, emp e2
WHERE  e1.mgr=e2.empno
AND    e1.sal > e2.sal/2;

ENAME         SAL
------ ----------
ALLEN        1600
BLAKE        2850
TURNER       1500
MILLER       1300
```


2.5 Aggregatoperationen

- Übersicht über Operationen
- Besonderheiten bei COUNT
- Gruppierung
- Gruppen und Verbund
- Selektion von Gruppen

Übersicht: Operationen

Name der Operation .... Beschreibung

max (Spaltenname) .... Maximum

min (Spaltenname) .... Minimum

avg (Spaltenname) .... Durchschnitt der Werte

sum (Spaltenname) .... Summe der Werte

count (Spaltenname) .... Anzahl der Werte

Beispiel: Aggregatoperationen

Wie hoch ist das höchste Gehalt aller Angestellten?

```
SELECT MAX(sal)
FROM   emp;

  MAX(SAL)
----------
      5000
```


Variationen von COUNT

- COUNT (*) zählt alle Zeilen
- COUNT (*spalte*) zählt alle Zeilen, die in *spalte* einen Wert enthalten
- COUNT (DISTINCT *spalte*) zählt die Anzahl der verschiedenen Werte in *spalte*

Gruppierung: Merkmale

- Normalerweise: Aggregation auf gesamte Relation; durch Gruppierung Anwendung auf disjunkte Teilenrelationen (=Gruppen)
- Pro Gruppe wird eine Zeile ausgegeben.
- Die Gruppierung wird durch GROUP BY festgelegt.
- In der SELECT-Klausel dürfen nur Konstante, Aggregatoperationen sowie Spaltennamen vorkommen, die auch in der GROUP BY-Klausel enthalten sind.

Beispiel: Gruppierung

Wie hoch ist das höchste Gehalt in jeder Abteilung?

```
SELECT   deptno, MAX(sal)
FROM     emp
GROUP BY deptno;

    DEPTNO   MAX(SAL)
---------- ----------
        10       5000
        30       2850
```


Gruppen, Verbund: Auswertungsprinzipien

1. Bildung des Verbunds
2. Selektion der Zeilen der Relation gemäß WHERE-Bedingung
3. Gruppierung nach Gruppierungsattributen
4. Selektion der Gruppen gemäß HAVING-Bedingung
5. Projektion auf gesuchte Attribute

Beispiel: COUNT, SUM und AVG

Wieviele Angestellte mit demselben Beruf gibt es in jeder Abteilung? Wieviel verdienen die Mitarbeiter einer Berufsgruppe einer Abteilung insgesamt?

```
SELECT   dname, job, SUM(sal)
FROM     emp, dept
WHERE    emp.deptno=dept.deptno
GROUP BY dname, job;

DNAME      JOB        SUM(SAL)
---------- ---------  --------
ACCOUNTING CLERK          1300
ACCOUNTING MANAGER        2450
ACCOUNTING PRESIDENT      5000
SALES      CLERK           950
SALES      MANAGER        2850
SALES      SALESMAN       5600
```


Beispiel: unzulässige Abfrage

Unzulässig ist folgende Anweisung:

```
SELECT      job, empno, sum(sal)
FROM        emp
GROUP BY    job;
```

Grund:
empno im SELECT-Teil ist keine Konstante, keine Aggregatoperation und kein Attribut, das in der GROUP BY-Klausel vorkommt.

Auswahl von Gruppen: HAVING-Klausel

Wieviele Angestellte mit demselben Beruf gibt es in jeder Abteilung? Wieviel verdienen die Mitarbeiter einer Berufsgruppe einer Abteilung insgesamt und im Durchschnitt? Zu berücksichtigen sind nur Gruppen mit mindestens zwei Mitarbeitern.

```
SELECT   dname, job, SUM(sal), COUNT(*), AVG(sal)
FROM     emp, dept
WHERE    emp.deptno=dept.deptno
GROUP BY dname, job
HAVING   COUNT(*) >= 2;

DNAME      JOB         SUM(SAL)   COUNT(*)   AVG(SAL)
---------- --------- ---------- ---------- ----------
SALES      SALESMAN        5600          4       1400
```


2.6 Teilabfragen (Subqueries)

Teilabfrage:
Abfrage in der WHERE-Klausel einer anderen Abfrage (Schachtelung von SELECT-Statements)

Arten von Teilabfragen:
Einfache Teilabfrage
Teilabfrage, die für die gesamte Hauptabfrage nur einmal ausgeführt werden muß

Korrelierte Teilabfrage
Teilabfrage, die für jedes Tupel der Hauptabfrage einmal ausgeführt werden muß

Einfache Teilabfragen

Gesucht sind die Namen aller Personen, die denselben Job haben wie BLAKE.

```
SELECT ename, job
FROM   emp
WHERE  job =
       (SELECT job FROM   emp
        WHERE  ename='BLAKE');

ENAME  JOB
------ ---------
BLAKE  MANAGER
CLARK  MANAGER
```

Bei der obigen Abfrage muß die Subquery nur einmal ausgeführt werden.

Korrelierte Teilabfrage (mit Aliases)

Gesucht sind alle Mitarbeiter, deren Gehalt höher ist als die Hälfte des Durchschnittsgehalts ihrer Kollegen in derselben Abteilung.

```
SELECT   deptno, ename, sal
FROM     emp e1
WHERE    sal >
         (SELECT AVG(sal)/2
          FROM   emp e2
          WHERE  e1.deptno=e2.deptno
                 and e1.empNo = e2.empNo)
ORDER BY deptno;

DEPTNO ENAME         SAL
------ ------ ----------
    10 MILLER       1300
    30 ALLEN        1600
    30 BLAKE        2850
    30 TURNER       1500
```

Prädikate in Verbindung mit Teilabfragen

Ergebnis der Teilabfrage ist nicht ein einziges Tupel, sondern eine Menge von Tupeln.

--> Vergleich des aktuellen Tupels der Hauptabfrage mit einer Menge von Tupeln der Teilabfrage

- Prädikat ANY
- Prädikat ALL
- Prädikat IN
- Prädikat NOT IN
- Prädikat EXISTS

Beispiel: ALL

Gesucht sind alle Mitarbeiter, die mehr verdienen als der Durchschnitt jeder Abteilung.

```
SELECT ename, sal
FROM   emp
WHERE  sal > ALL
       (SELECT   AVG(sal)
        FROM     emp
        GROUP BY deptno);

ENAME         SAL
------ ----------
KING         5000
```


Beispiel: ANY

Gesucht sind alle Abteilungen, denen zumindest ein Mitarbeiter angehört.

```
SELECT dname
FROM   dept
WHERE  deptno = ANY
       (SELECT deptno
        FROM   emp);

DNAME
----------
ACCOUNTING
SALES
```

Hinweis: “IN” ist gleichbedeutend mit “=ANY”

Beispiel: IN

Gesucht sind alle Abteilungen, denen kein Mitarbeiter angehört.

```
SELECT dname
FROM   dept
WHERE  deptno NOT IN
       (SELECT deptno
        FROM   emp);

DNAME
----------
OPERATIONS
```

Beispiel: EXISTS

Gesucht sind die Abteilungen mit Mitarbeitern, die am 3. Dez. 81 eingestellt wurden.

```
SELECT dname
FROM   dept
WHERE  EXISTS
       (SELECT *
        FROM   emp
        WHERE  dept.deptno=emp.deptno
        AND    hiredate = '03-DEC-81');

DNAME
----------
SALES
```


3. Datendefinition

- Anlegen von Relationen
  CREATE TABLE
- Ändern des Relationenschemas
  ALTER TABLE
- Löschen von Relationen
  DROP TABLE

Erzeugen von Tabellen (1)

Tabelle emp für Mitarbeiterdaten

```
CREATE TABLE emp
             (empno     NUMBER (4) NOT NULL
                        PRIMARY KEY,
              ename     CHAR (30) NOT NULL,
              job       CHAR (15),
              mgr       NUMBER (4),
              hiredate  DATE,
              sal       NUMBER (7,2)
                        NOT NULL,
              comm      NUMBER (7,2),
              deptno    NUMBER (2)
              );
```


Erzeugen von Tabellen (2)

Tabelle dept für Abteilungen

```
CREATE TABLE dept
             (deptno  NUMBER(2) NOT NULL
                      PRIMARY KEY,
              dname   CHAR(10) NOT NULL,
              loc     CHAR(10)
             );
```

Ändern des Tabellenaufbaus

Einfügen eines zusätzlichen Attributs

```
ALTER TABLE emp ADD (projno NUMBER(3));
```

Ändern eines Attributs

```
ALTER TABLE emp MODIFY (sal NUMBER(10,2));
```


Löschen von Tabellen

```
DROP TABLE emp;
DROP TABLE dept;
```


4. Datenmanipulation

- Einfügen von Datensätzen:
  INSERT INTO table
- Ändern von Datensätzen:
  UPDATE table
- Löschen von Datensätzen:
  DELETE FROM table

Einfügen von Datensätzen

Variante 1:

```
INSERT INTO emp VALUES
(7654,'MARTIN', 'SALESMAN', 7698,
'28-SEP-81', 1250, 1400, 30);
```

Variante 2:

```
INSERT INTO emp
(empno, ename, job, mgr, hiredate, sal,
deptno)VALUES
(7369, 'SMITH','CLERK',7902, '17-DEC-80',
800, 20);
```

Ändern von Datensätzen

Adams, James und alle Verkäufer bekommen 500 Gehalts-erhöhung, die Provision wird auf 100 gesetzt.

```
UPDATE   emp
SET      sal = sal+500, comm=100
WHERE    ename in (‘ADAMS’, ‘JAMES’)
OR       job=’SALESMAN’;
```

Alle Mitglieder, die in New York arbeiten, bekommen eine Gehaltserhöhung.

```
UPDATE   emp
SET      sal = sal+1000
WHERE    deptno IN
         (SELECT deptno
          FROM   dept
          WHERE  loc=’NEW YORK’);
```


Löschen von Datensätzen

Löschen aller Datensätze einer Relation:

```
DELETE FROM emp;
```

Löschen einer ausgewählten Menge von Datensätzen:

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


5. Transaktionen

- Die meisten SQL-Statements können nur im Rahmen einer *Transaktion* abgewickelt werden. Dazu gehören die Befehle INSERT, UPDATE, SELECT, CREATE TABLE ...
- Eine Transaktion kann mehrere SQL-Befehle umfassen. Ist noch keine Transaktion aktiv und wird ein SQL-Statement an das System übergeben, wird **automatisch** eine neue **Transaktion gestartet**.
- Änderungen, die durch die Befehle einer Transaktion vorgenommen werden, werden erst nach erfolgreichem **Abschluß** der Transaktion **mit COMMIT** wirksam.
- Eine Transaktion kann **mit ROLLBACK abgebrochen** werden. Alle in dieser Transaktion durchgeführten Änderungen werden unwirksam.

6. Der SQL-3-Standard

- Veröffentlichung: 1995/96
- neue Konzepte: Triggers
- neue Datentypen (BOOLEAN)
- rekursive Operationen
- stored procedures

Triggers - Allgemeines

Triggers sind

- Aktionen, die
- beim Eintreffen eines bestimmten Ereignisses
- unter bestimmten Bedingungen
- vom DBMS automatisch ausgeführt werden.

Triggers: Anwendungsgebiete

- gespeicherte abgeleitete Attribute:
  Beispiel: Bruttopreis bei gespeichertem Nettopreis und USt-Satz
- Allgemeine Bedingungen:
  Beispiel: Nachbestellung, wenn der Lagerbestand einen bestimmten Schwellenwert unterschreitet

Triggers - Syntax

- Definition eines Triggers: CREATE TRIGGER
- Auslösende Ereignisse können sein: INSERT, UPDATE, DELETE
- Trigger kann vor oder nach Ausführung des Ereignisses “gefeuert” werden (BEFORE INSERT, AFTER INSERT etc.).
- Das Feuern kann von zusätzlichen Bedingungen abhängig gemacht werden.
- Ist das auslösende Ereignis UPDATE angegeben, so kann zusätzlich definiert werden, wie auf die Werte der zu verändernden Attribute vor und nach der Änderung zugegriffen werden kann.

Triggers - Beispiel

Das Attribut salary der Relation EMP enthält das Grundgehalt der Mitarbeiter.
Der Präsident soll immer das Durchschnittsgehalt aller anderen Mitarbeiter + 3000 erhalten.
Wird das Gehalt eines Mitarbeiters verändert, muß daher auch das Präsidenten-Gehalt korrigiert werden.

```
CREATE TRIGGER Sal_Modification
AFTER UPDATE (salary) ON emp
REFERENCING NEW AS new_emp

WHEN (job<>'PRESIDENT')
UPDATE emp
    SET salary = (SELECT AVG(salary) FROM new_emp
                  WHERE job<>'PRESIDENT') + 3000
    WHERE new_emp.job='PRESIDENT'
```