

LB-DB 3 - 14.3.2015

Dipl.-Ing. Reinhard Schlager

its
FH Salzburg

2015/ LB-Datenbanksysteme

Gliederung

- 1 SQL
 - Aggregatfunktionen
 - GROUP BY
 - Sub-SELECT
- 2 SQL scripts - PL/SQL -
 - stored procedure
 - stored function
 - Trigger
 - sequence
 - index
- 3 Übung

Agregatfunktionen

MAX(),MIN(),SUM(),AVG()

```
SELECT aggfunction(att1) FROM tab1 [WHERE ...]
```

```
SELECT MAX(salary) FROM employees
```

```
SELECT AVG(salary) FROM employees  
WHERE department_id = 100
```

GROUP BY

Gruppieren gleicher Datensätze

```
SELECT department_id,AVG(salary)  
FROM employees  
GROUP BY department_id
```

DEPARTMENT_ID	AVG(SALARY)
100	8600
30	4150
...	...

Sub-SELECT

```
SELECT e1.department_id,e1.first_name,e1.salary
, (SELECT avg(e2.salary) FROM employees e2
   WHERE e1.department_id = e2.department_id
   GROUP BY e2.department_id
   ) AS AVG_SAL
FROM employees e1
```

DEPARTMENT_ID	FIRST_NAME	SALARY	AVG_SAL
90	Steven	24000	19333.33...
90	Neena	17000	19333.33...
90	Lex	17000	19333.33...
60	Alexander		
...

PL/SQL

```
DECLARE
    m_salary NUMBER(6); nr_days  NUMBER(2);
    per_day  NUMBER(6,2);
BEGIN
    m_salary := 2290;
    nr_days  := 21;
    per_day := m_salary/nr_days;
    DBMS_OUTPUT.PUT_LINE
        (' per day=' ||TO_CHAR(per_day));
EXCEPTION
WHEN ZERO_DIVIDE THEN
    per_day := 0;
END;
```

stored procedure

```
CREATE PROCEDURE today_is AS
BEGIN
    DBMS_OUTPUT.PUT_LINE
        (' Today is ' || TO_CHAR(SYSDATE, ' DL ') );
END today_is;
--Aufruf durch
BEGIN
    today_is();
END;
```

stored function

```
CREATE FUNCTION worked_for (empid NUMBER)
RETURN VARCHAR2 IS
    years INT;
BEGIN
    SELECT round((sysdate-hire_date )/365)
    INTO years FROM employees
    WHERE employee_id = empid;
    RETURN (' worked for approx. '
        || years || ' years');
END worked_for;
```

Anwendung

```
SELECT hire_date, worked_for (employee_id)
FROM employees
ORDER BY hire_date
```

Trigger

```
CREATE OR REPLACE TRIGGER audit_sal
AFTER UPDATE OF salary
ON employees FOR EACH ROW
BEGIN
INSERT INTO emp_audit VALUES
( :OLD.employee_id, SYSDATE,
  :NEW.salary, :OLD.salary );
END;
```

sequence

```
CREATE SEQUENCE new_employees_seq  
START WITH 1000 INCREMENT BY 1;  
  
INSERT INTO employees  
(employee_id, first_name,  
last_name, email, hire_date, job_id)  
VALUES  
(new_employees_seq.nextval,  
' a', ' b', ' c', ' 14-mar-2015', ' SA_MAN')
```

CREATE INDEX

```
CREATE [UNIQUE] INDEX <index_name>  
ON <table_name>  
(<field_name>{<field_name>})
```

Beispiel

```
CREATE UNIQUE INDEX emp_mgr_id_ix  
ON employees  
(employee_id)
```

Übung 3 LB-DB 14.3.2015

- Schreiben Sie eine Funktion `dif_to_avg(employee_id)`, die die Abweichung des Gehalts des Mitarbeiters vom Durchschnitt ermittelt.
- Schreiben Sie einen Trigger, der das Reduzieren eines Gehalts verhindert. Wenn der neue Gehalt kleiner als der alte Gehalt ist, soll der Gehalt nicht verändert werden.
- Formulieren Sie 3 Abfrage auf Basis des HR Schemas. In diesen Abfragen müssen die Konstrukte `GROUP BY`, `HAVING`, `MAX` zumindest einmal vorkommen. Beschreiben Sie die Abfrage und zeigen Sie das Ergebnis.