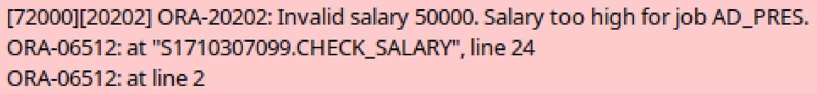
Ausarbeitung UE08

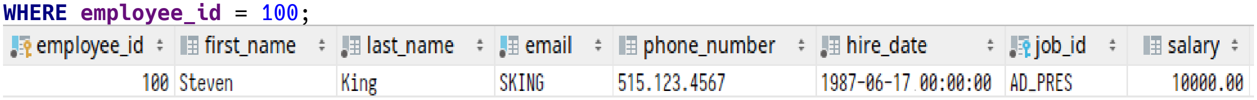
1. Trigger

```
-- 1.1
CREATE OR REPLACE PROCEDURE check_salary(jid IN jobs.job_id%TYPE, salary IN jobs.min_salary%TYPE) IS
  min_sal jobs.min_salary%TYPE;
  max_sal jobs.max_salary%TYPE;
  salary_too_high EXCEPTION;
BEGIN
  -- get salaries from job
  SELECT min_salary,
         max_salary
         INTO min_sal, max_sal
  FROM jobs
  WHERE job_id = jid;

  IF (salary > max_sal) THEN -- if max salary < salary -> raise exception
    RAISE salary_too_high;
  ELSIF (salary < min_sal) THEN -- if min salary > salary -> change min salary
    UPDATE jobs
    SET min_salary = salary
    WHERE job_id = jid;
  END IF;

  EXCEPTION
  WHEN salary_too_high
  THEN
    RAISE_APPLICATION_ERROR(-20202, 'Invalid salary ' || salary || '. Salary too high for job ' ||
jid || '.');
END;

BEGIN
  check_salary('AD_PRES', 50000);
END;
/
```



```
[72000][20202] ORA-20202: Invalid salary 50000. Salary too high for job AD_PRES.
ORA-06512: at "S1710307099.CHECK_SALARY", line 24
ORA-06512: at line 2
```

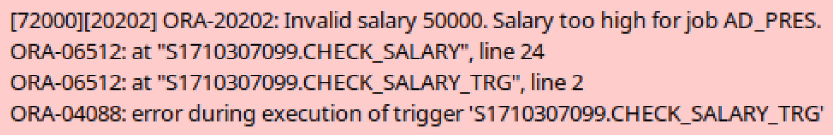
```
-- 1.2
CREATE OR REPLACE TRIGGER check_salary_trg
  BEFORE INSERT OR UPDATE
  ON employees
  FOR EACH ROW
BEGIN
  CHECK_SALARY(:new.job_id, :new.salary);
END;

-- 1.3
UPDATE employees
SET salary = 10000
WHERE employee_id = 100;

SELECT *
FROM employees
WHERE employee_id = 100;
```



employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary
100	Steven	King	SKING	515.123.4567	1987-06-17 00:00:00	AD_PRES	10000.00

```
UPDATE employees
SET salary = 50000
```

```
WHERE employee_id = 100;
```



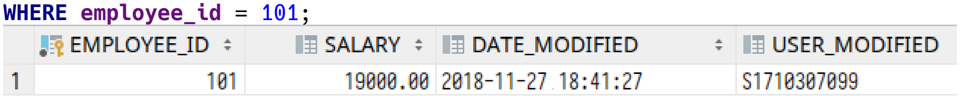
```
[72000][20202] ORA-20202: Invalid salary 50000. Salary too high for job AD_PRES.
ORA-06512: at "S1710307099.CHECK_SALARY", line 24
ORA-06512: at "S1710307099.CHECK_SALARY_TRG", line 2
ORA-04088: error during execution of trigger 'S1710307099.CHECK_SALARY_TRG'
```

```
ROLLBACK;
```

1.4

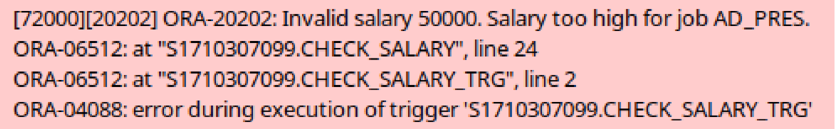
Informationen über den Mitarbeiter gehen beim Kontext-Wechsel vom Trigger in die Prozedur verloren. Man müsste entweder direkt die manager_id oder eine beliebige andere Information an die Prozedur übergeben, über die sich der Manager des eingefügten oder bearbeiteten Mitarbeiters ableiten lässt. Streng genommen gibt es kein Problem wenn man diese Überprüfung im Trigger machen will, anstatt in der Prozedur. (Zumindest habe ich keines entdeckt). Es wäre allerdings auch nicht sonderlich sauber.

```
-- 1.5
ALTER TABLE employees
  ADD (date_modified DATE, user_modified VARCHAR2(255));

CREATE OR REPLACE TRIGGER log_employees
  BEFORE INSERT OR UPDATE
  ON employees
  FOR EACH ROW
BEGIN
  :new.user_modified := USER;
  :new.date_modified := SYSDATE;
END;

-- TEST: default use case
UPDATE employees
SET salary = 19000
WHERE employee_id = 101;
```



	EMPLOYEE_ID	SALARY	DATE_MODIFIED	USER_MODIFIED
1	101	19000.00	2018-11-27 18:41:27	S1710307099

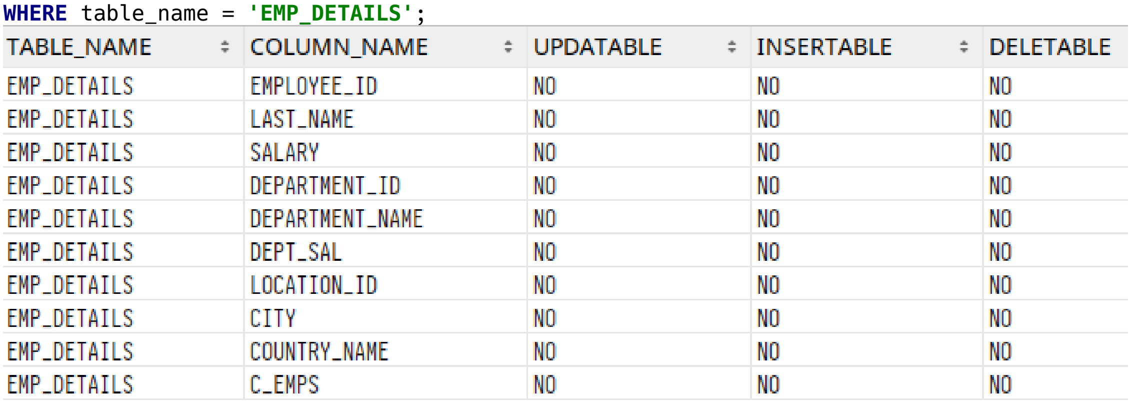
```
-- TEST: invalid employee
UPDATE employees
SET salary = 19000
WHERE employee_id = 10;

-- TEST: salary > max_salary
UPDATE employees
SET salary = 50000
WHERE employee_id = 100;
```



```
[72000][20202] ORA-20202: Invalid salary 50000. Salary too high for job AD_PRES.
ORA-06512: at "S1710307099.CHECK_SALARY", line 24
ORA-06512: at "S1710307099.CHECK_SALARY_TRG", line 2
ORA-04088: error during execution of trigger 'S1710307099.CHECK_SALARY_TRG'
```

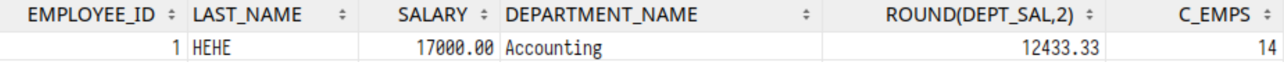
2. INSTEAD OF-Trigger

```
-- 2.1
CREATE OR REPLACE VIEW emp_details AS
SELECT employee_id,
       last_name,
       salary,
       department_id,
       department_name,
       dept_sal,
       location_id,
       city,
       country_name,
       c_emps
FROM new_emps
       INNER JOIN new_depts USING (department_id)
       INNER JOIN new_locs USING (location_id)
       INNER JOIN new_countries USING (country_id);

SELECT *
FROM user_updatable_columns
WHERE table_name = 'EMP_DETAILS';
```



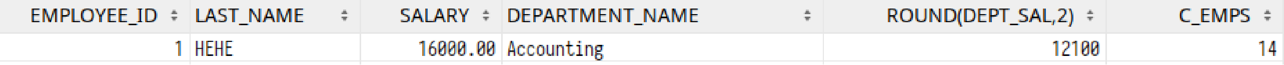
TABLE_NAME	COLUMN_NAME	UPDATABLE	INSERTABLE	DELETABLE
EMP_DETAILS	EMPLOYEE_ID	NO	NO	NO
EMP_DETAILS	LAST_NAME	NO	NO	NO
EMP_DETAILS	SALARY	NO	NO	NO
EMP_DETAILS	DEPARTMENT_ID	NO	NO	NO
EMP_DETAILS	DEPARTMENT_NAME	NO	NO	NO
EMP_DETAILS	DEPT_SAL	NO	NO	NO
EMP_DETAILS	LOCATION_ID	NO	NO	NO
EMP_DETAILS	CITY	NO	NO	NO
EMP_DETAILS	COUNTRY_NAME	NO	NO	NO
EMP_DETAILS	C_EMPS	NO	NO	NO

```
-- 2.2
CREATE OR REPLACE TRIGGER dml_emp_details
  INSTEAD OF INSERT OR UPDATE OR DELETE
  ON emp_details
  FOR EACH ROW
BEGIN
  IF INSERTING THEN -- (b)
    -- insert employee
    INSERT INTO new_emps
    VALUES (:new.employee_id, :new.last_name, :new.salary, :new.department_id);

    -- raise dept_sal
    UPDATE new_depts
    SET dept_sal = (SELECT AVG(salary) FROM new_emps WHERE department_id = :new.department_id)
    WHERE department_id = :new.department_id;

    -- increase country "population"
    UPDATE new_countries
    SET c_emps = c_emps + 1
    WHERE country_name = :new.country_name;

  ELSIF UPDATING ('salary') THEN -- (c)
    -- update salary
    UPDATE new_emps
    SET salary = :new.salary
    WHERE employee_id = :new.employee_id;

    -- also update average department salary
    UPDATE new_depts
    SET dept_sal = (SELECT AVG(salary) FROM new_emps WHERE department_id = :new.department_id)
    WHERE department_id = :new.department_id;

  ELSIF UPDATING ('department_id') THEN -- (d)
    -- update employee dept_id
    UPDATE new_emps
    SET department_id = :new.department_id
    WHERE employee_id = :new.employee_id;

    -- update average salary for both involved depts
    UPDATE new_depts
    SET dept_sal = (SELECT AVG(salary) FROM new_emps WHERE department_id = :old.department_id)
    WHERE department_id = :old.department_id;

    UPDATE new_depts
    SET dept_sal = (SELECT AVG(salary) FROM new_emps WHERE department_id = :new.department_id)
    WHERE department_id = :new.department_id;

    -- update country "population"
    UPDATE new_countries
    SET c_emps = (SELECT COUNT(*) FROM new_emps WHERE department_id = :old.department_id)
    WHERE country_name = :new.country_name;

  ELSIF DELETING THEN -- (a)
    -- delete from employees
    DELETE
    FROM new_emps
    WHERE employee_id = :old.employee_id;

    -- lower dept_sal
    UPDATE new_depts
    SET dept_sal = (SELECT AVG(salary) FROM new_emps WHERE department_id = :old.department_id)
    WHERE department_id = :old.department_id;

    -- decrease country "population"
    UPDATE new_countries
    SET c_emps = c_emps - 1
    WHERE country_name = :old.country_name;

  END IF;
END;

INSERT INTO emp_details (employee_id,
                         last_name,
                         salary,
                         department_id,
                         country_name)
VALUES (1, 'HEHE', 17000, 110, 'Canada');
```



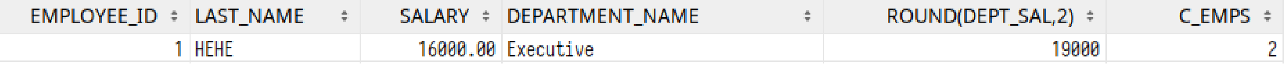
EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_NAME	ROUND(DEPT_SAL,2)	C_EMPS
1	HEHE	17000.00	Accounting	12433.33	14

```
UPDATE emp_details
SET salary = 16000
WHERE employee_id = 1;
```



EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_NAME	ROUND(DEPT_SAL,2)	C_EMPS
1	HEHE	16000.00	Accounting	12100	14

```
UPDATE emp_details
SET department_id = 90
WHERE employee_id = 1;
```

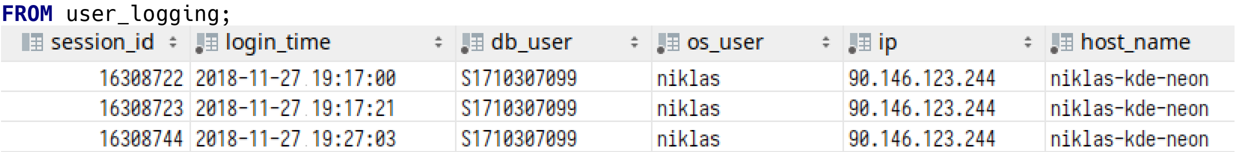


EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_NAME	ROUND(DEPT_SAL,2)	C_EMPS
1	HEHE	16000.00	Executive	19000	2

```
DELETE
FROM emp_details
WHERE employee_id = 1;

ROLLBACK;
```

3. Trigger für Systemereignisse

```
CREATE OR REPLACE TRIGGER user_logging_trg
  AFTER LOGON ON SCHEMA
BEGIN
  INSERT INTO user_logging
  VALUES (SYS_CONTEXT('USERENV', 'SESSIONID'),
          SYSDATE,
          USER,
          SYS_CONTEXT('USERENV', 'OS_USER'),
          SYS_CONTEXT('USERENV', 'IP_ADDRESS'),
          SYS_CONTEXT('USERENV', 'HOST'));
  -- A failing trigger can prevent user login
  -- so I added this exception handler, hoping that
  -- this is all it takes to bypass this behavior.
  EXCEPTION
  WHEN OTHERS
  THEN
    dbms_output.PUT_LINE('Error in trigger!');
END;

SELECT *
FROM user_logging;
```



session_id	login_time	db_user	os_user	ip	host_name
16308722	2018-11-27 19:17:00	S1710307099	niklas	90.146.123.244	niklas-kde-neon
16308723	2018-11-27 19:17:21	S1710307099	niklas	90.146.123.244	niklas-kde-neon
16308744	2018-11-27 19:27:03	S1710307099	niklas	90.146.123.244	niklas-kde-neon