Inhaltsverzeichnis

[**Übungszettel 1 – Einführung HR** 3](#_Toc472257400)

[1. Schreiben Sie ein Programm, welches das Einkommen der Mitarbeiter 119 und 124 vertauscht (1L) 3](#_Toc472257401)

[2. Erhöhen Sie das Einkommen der Mitarbeiter 110 wie folgt: 4](#_Toc472257402)

[3. Der Provisionsprozentsatz für den Mitarbeiter 148 soll berechnet werden. Das geht so: 4](#_Toc472257403)

[4. ermitteln Sie den Vorgesetzten des Mitarbeiters 103. Geben Sie den Namen und die Abteilung aus, wo dieser Vorgesetzte arbeitet. 4](#_Toc472257404)

[5. Suche Sie nach Lücken und den Mitarbeiter IDs (1L) 4](#_Toc472257405)

[6. Ermitteln Sie das Jahr, in dem die meisten Mitarbeiter eingestellt worden sind. Zeigen Sie - je Monat- die Anzahl der aufgenommenen Mitarbeiter an. 5](#_Toc472257406)

[**Übungszettel 2 – Cursor HR** 6](#_Toc472257407)

[1. Zeigen Sie den Job Title und den Namen des Mitarbeiters an, der am längsten in der Firma arbeitet (1L) 6](#_Toc472257408)

[2. Zeigen Sie den 2. bis zum 9 Mitarbeiter in der Mitarbeiter Tabelle an. (3L) 6](#_Toc472257409)

[3. Geben Sie den Mitarbeitern eine Gehaltserhöhung: (1L) 8](#_Toc472257410)

[**Übungszettel 3 – Functions HR** 9](#_Toc472257411)

[1. Erstellen Sie eine Funktion, die als Parameter die Abteilungsnummer hat und den Namen des Managers liefert. (1L) 9](#_Toc472257412)

[2. Erstellen Sie eine Funktion, die die MitarbeiterNr bekommt und die Anzahl seiner Jobs liefert (2L) 9](#_Toc472257413)

[3. Erstellen Sie eine Funktion, der die ManagerID mitgegeben wird. Diese Funktion soll eine Liste aller seiner Mitarbeiter liefern. (1L) 10](#_Toc472257414)

[**Übungszettel 4 – Exceptions HR** 11](#_Toc472257415)

[1. Ändern Sie das Gehalt des Mitarbeiters 129: (2L) 11](#_Toc472257416)

[2. Erstellen Sie eine Prozedur, der die Abteilungsnummer mitgegeben wird. Suchen Sie den Mitarbeiter mit dem höchsten Einkommen und machen Sie diesen zum Manager der Abteilung. (2L) 12](#_Toc472257417)

[**Übungszettel 5 – Trigger HR** 14](#_Toc472257418)

[1. Stellen Sie sicher, daß an der Mitarbeitertabelle vor 5:00 und nach 23:00 keine Änderungen gemacht werden können. (2L) 14](#_Toc472257419)

[2. Stellen Sie sicher, dass das Gehalt eines Mitarbeiters nicht verkleinert werden kann. (1L) 14](#_Toc472257420)

[3. Schreiben Sie einen Trigger, der sicher stellt, dass der Mitarbeiter und der Vorgesetzte zur gleichen Abteilung gehören. (1L) 14](#_Toc472257421)

[4. Jedes Mal, wenn der Job eines Mitarbeiters geändert wird, soll eine Logging Information (job history) geschrieben werden: (1L) 15](#_Toc472257422)

[**PLÜFen vom letzten Schuljahr (Teil 1)** 16](#_Toc472257423)

[1. Implmentieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie das Max\_Salary seiner Job\_ID (insert) 16](#_Toc472257424)

[2. Schreiben Sie einen Trigger, der die Email Adresse eines Mitarbeiters (beim Insert und falls leer) ändert auf <Vorname>,<Familienname>@<Department\_Name>“ 16](#_Toc472257425)

[3. Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters zum Min- bzw. Max\_Salary seiner Job\_ID passt. Geben Sie dem Mitarbeiter eine Gehaltsreduktion um 5%, falls er mehr als das Max\_Salary verdient. 16](#_Toc472257426)

[4. Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann. 16](#_Toc472257427)

[**PLÜFen vom letzten Schuljahr (Teil 2)** 17](#_Toc472257428)

[1. Implementieren Sie einen INSERT Trigger, der prüft ob MIN\_Salary und MAX\_Salary eines JOBS befüllt ist. Falls eines der beiden leer ist, setzten sie den vorhandenen Wert ein. Falls beide leer sind, erzeugen Sie eine entsprechende Exception. 17](#_Toc472257429)

[2. Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters grösser als das Durchschnittsgehalt seiner Kollegin in den anderen Abteilung ist. Geben Sie dem Mitarbeiter - in diesem Fall – eine Gehaltsreduktion um 5% 17](#_Toc472257430)

[3. Erstellen Sie eine Funktion, die die Daten Location/Department/Employees korrekt anlegen kann. 17](#_Toc472257431)

[4. Schreiben Sie eine Funktion, die für einen Mitarbeiter ermittelt, ob dieser einen Job Wechsel hatte. Falls das der Fall war, überprüfen Sie, ob sich der Job Title geändert hat. Geben Sie diese beiden Job Title durch „,“ getrennt von der Funktion zurück. 17](#_Toc472257432)

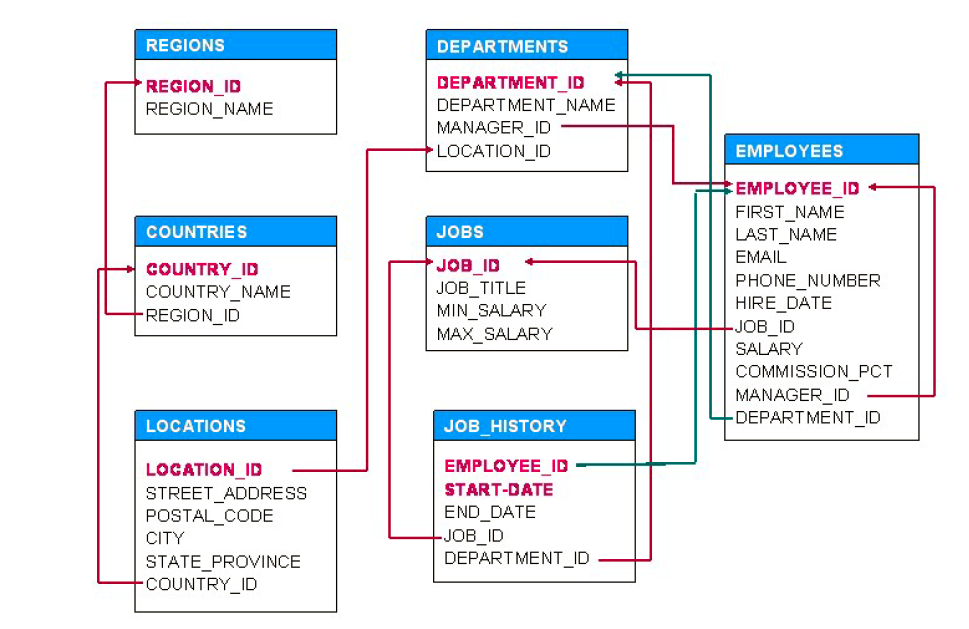
[**PLÜF von der AHIF** 17](#_Toc472257433)

[1. Implementieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie die Hälfte der Max\_Salary seiner Job\_ID (update) 17](#_Toc472257434)

[2. Schreiben Sie eine Funktion, die die Email Adresse eines Mitarbeiters (falls leer) ändert auf <Vorname>@<Familienname>.<Department\_Name>“ 17](#_Toc472257435)

[3. Erstellen Sie eine Prozedur, die den Gehalts- Mittelwert ( (min+max)/2 ) einer gegebenen Job\_ID ermittelt. Suchen Sie einen Mitarbeiter mit der gegebenen JOB\_ID, dessen Salary möglichst gleich diesem Mittelwert ist. 17](#_Toc472257436)

[4. Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann. 17](#_Toc472257437)



# Übungszettel 1 – Einführung HR

## Schreiben Sie ein Programm, welches das Einkommen der Mitarbeiter 119 und 124 vertauscht (1L)

**create** **or** **replace** **PROCEDURE** SWAP\_SALARY2 **(**

p\_emp\_1 **IN** employees**.**employee\_id**%type,**

p\_emp\_2 **IN** employees**.**employee\_id**%type**

**)** **AS**

v\_sal\_1 employees**.**salary**%type;**

v\_sal\_2 employees**.**salary**%type;**

**BEGIN**

**update** employees **set** salary **=** get\_salary\_by\_id**(**p\_emp\_2**)**

**where** employee\_id **=** p\_emp\_1**;**

**update** employees **set** salary **=** get\_salary\_by\_id**(**p\_emp\_1**)**

**where** employee\_id **=** p\_emp\_2**;**

**END** SWAP\_SALARY2**;**

Dazugehörige Funktion

**create** **or** **replace** **FUNCTION** GET\_SALARY\_BY\_ID

**(**

EMP **IN** employees**.**employee\_id**%type**

**)** **RETURN** employees**.**salary**%type** **AS**

v\_sal employees**.**salary**%type;**

**BEGIN**

**select** salary

**into** v\_sal

**from** employees

**where** employee\_id **=** emp**;**

**RETURN** v\_sal**;**

**END** GET\_SALARY\_BY\_ID**;**

## Erhöhen Sie das Einkommen der Mitarbeiter 110 wie folgt:

- die Firmenzugehörigkeit ist mehr als 7 Jahre -> 25%

- die Firmenzugehörigkeit ist mehr als 4 Jahre -> 12%

- in jedem anderen Fall: -> 9%

## Der Provisionsprozentsatz für den Mitarbeiter 148 soll berechnet werden. Das geht so:

- wenn das Gehalt mehr als 9000 ist -> 0.45%

- wenn das Gehalt < 9000, aber die Firmenzugehörigkeit > 7 Jahre ist -> 0.32%

- wenn das Gehalt <= 2500 -> 0.24%

- in jedem anderen Fall -> 0.09%.

## ermitteln Sie den Vorgesetzten des Mitarbeiters 103. Geben Sie den Namen und die Abteilung aus, wo dieser Vorgesetzte arbeitet.

## Suche Sie nach Lücken und den Mitarbeiter IDs (1L)

**create** **or** **replace** **PROCEDURE** CHECK\_IDS **AS**

v\_uid1 employees**.**Employee\_ID**%Type;**

v\_uid2 employees**.**Employee\_ID**%Type;**

v\_uidmax employees**.**Employee\_ID**%Type;**

v\_uidmin employees**.**Employee\_ID**%Type;**

**BEGIN**

**select** **max(**Employee\_ID**),** **min(**Employee\_ID**)**

**into** v\_uidmax**,** v\_uidmin

**from** employees**;**

**while(**v\_uidmin **<** v\_uidmax**)**

**loop**

**select** Employee\_ID

**into** v\_uid1

**from** employees

**where** Employee\_ID **=** v\_uidmin**;**

**select** Employee\_ID

**into** v\_uid2

**from** employees

**where** Employee\_ID **=** v\_uidmin **+** 1**;**

**if(**v\_uid2 **-** v\_uid1 **!=** 1**)**

**then** dbms\_output**.**put\_line**(**'Eine Lücke zwischen' **||** v\_uid1 **||** ' und ' **||** v\_uid2 **);**

**end** **if;**

v\_uidmin **:=** v\_uidmin +1**;**

**end** **loop;**

**NULL;**

**END** CHECK\_IDS**;**

## Ermitteln Sie das Jahr, in dem die meisten Mitarbeiter eingestellt worden sind. Zeigen Sie - je Monat- die Anzahl der aufgenommenen Mitarbeiter an.

# Übungszettel 2 – Cursor HR

## Zeigen Sie den Job Title und den Namen des Mitarbeiters an, der am längsten in der Firma arbeitet (1L)

**CREATE** **OR** **REPLACE** **PROCEDURE** UE2\_A1 **AS**

v\_FIRST\_NAME EMPLOYEES**.**FIRST\_NAME**%type;**

v\_LAST\_NAME EMPLOYEES**.**LAST\_NAME**%type;**

v\_JOB\_TITLE JOBS**.**JOB\_TITLE**%type;**

**BEGIN**

**select** FIRST\_NAME**,** LAST\_NAME**,** JOB\_TITLE

**into** v\_FIRST\_NAME**,** v\_LAST\_NAME**,** v\_JOB\_TITLE

**from** **(select** FIRST\_NAME**,** LAST\_NAME**,** JOB\_TITLE

**from** EMPLOYEES

**inner** **join** JOBS **on** **(** EMPLOYEES**.**JOB\_ID **=** JOBS**.**JOB\_ID**)**

**order** **by** hire\_date**)**

**where** **ROWNUM** **=** 1**;**

DBMS\_OUTPUT**.**put\_line**(**v\_FIRST\_NAME **||** ' ' **||** v\_LAST\_NAME **||** ', ' **||** v\_JOB\_TITLE**);**

**END** UE2\_A1**;**

-- Steven King, President

## Zeigen Sie den 2. bis zum 9 Mitarbeiter in der Mitarbeiter Tabelle an. (3L)

**CREATE** **OR** **REPLACE** **PROCEDURE** UE2\_A2 **AS**

**cursor** employee\_cur **is**

**select** FIRST\_NAME**,** LAST\_NAME

**from** employees

**where** **rownum** **between** 2 **and** 9**;**

v\_employee employee\_cur**%ROWTYPE;**

**BEGIN**

**OPEN** employee\_cur**;**

**LOOP**

**FETCH** employee\_cur **INTO** v\_employee**;**

**EXIT** **WHEN** employee\_cur**%**NOTFOUND**;**

DBMS\_OUTPUT**.**put\_line**(**v\_employee**.**FIRST\_NAME **||** ' ' **||** v\_employee**.**LAST\_NAME **);**

**END** **LOOP;**

**CLOSE** employee\_cur**;**

**END** UE2\_A2**;**

**create** **or** **replace** **PROCEDURE** SHOW\_EMP\_2\_9 **AS**

**CURSOR** c\_emp **is**

**select** employee\_id

**from** employees

**order** **by** 1**;**

**BEGIN**

**FOR** v\_emp\_record **IN** c\_emp

**LOOP**

**EXIT** **WHEN** c\_emp**%**ROWCOUNT **>** 9**;**

**if** **(**c\_emp**%**ROWCOUNT **>** 1**)** **then**

DBMS\_OUTPUT**.**PUT**(**c\_emp**%**ROWCOUNT**);**

DBMS\_OUTPUT**.**PUT**(**', '**);**

DBMS\_OUTPUT**.**PUT\_LINE**(**v\_emp\_record**.**employee\_id**);**

**end** **if;**

**END** **LOOP;**

**END** SHOW\_EMP\_2\_9**;**

**create** **or** **replace** **PROCEDURE** ZEIGEMITARBEITER **AS**

v\_id employees**.**employee\_id**%type;**

**cursor** cur **is**

**select** employees**.**employee\_id **from** employees

**where** **rownum** **>** 2**;**

**BEGIN**

**open** cur**;**

**loop**

**fetch** cur

**into** v\_id**;**

**exit** **when** cur**%**RowCount **>** 9**;**

dbms\_output**.**put\_line**(**v\_id**);**

**end** **loop;**

**close** cur**;**

**NULL;**

**END** ZEIGEMITARBEITER**;**

## Geben Sie den Mitarbeitern eine Gehaltserhöhung: (1L)

- wenn die Abteilung 40 ist -> 9 %

- wenn die Abteilung 70 ist -> 17%

- wenn der Provisionsprozentsatz (commission\_prct) > 0,35% ist -> 4%

- in jedem anderen Fall ->11%

**CREATE** **OR** **REPLACE** **PROCEDURE** UE2\_A3 **AS**

**cursor** employee\_cur **is**

**select** **\***

**from** employees**;**

v\_employee employee\_cur**%ROWTYPE;**

**BEGIN**

**OPEN** employee\_cur**;**

**LOOP**

**FETCH** employee\_cur **INTO** v\_employee**;**

**EXIT** **WHEN** employee\_cur**%**NOTFOUND**;**

**if** v\_employee**.**DEPARTMENT\_ID **=** 40 **then**

SET\_SALARY**(** v\_employee**.**employee\_id**,** v\_employee**.**salary **\*** 1.09**);**

**elsif** v\_employee**.**DEPARTMENT\_ID **=** 70 **then**

SET\_SALARY**(** v\_employee**.**employee\_id**,** v\_employee**.**salary **\*** 1.17**);**

**elsif** v\_employee**.**COMMISSION\_PCT **>** 0.35 **then**

SET\_SALARY**(** v\_employee**.**employee\_id**,** v\_employee**.**salary **\*** 1.04**);**

**else**

SET\_SALARY**(** v\_employee**.**employee\_id**,** v\_employee**.**salary **\*** 1.11**);**

**end** **if;**

**END** **LOOP;**

**CLOSE** employee\_cur**;**

**NULL;**

**END** UE2\_A3**;**

# Übungszettel 3 – Functions HR

## Erstellen Sie eine Funktion, die als Parameter die Abteilungsnummer hat und den Namen des Managers liefert. (1L)

**CREATE** **OR** **REPLACE** **FUNCTION** "5BHIF20167\_04"**.**"FUNCTIONS03\_1\_GREILINGER" **(**

v\_dep\_id **IN** **INT**

**)RETURN** **VARCHAR2** **AS**

v\_fn **VARCHAR2(**20**);**

v\_ln **VARCHAR2(**20**);**

**BEGIN**

**SELECT** EMPLOYEES**.**LAST\_NAME**,** EMPLOYEES**.**FIRST\_NAME

**INTO** v\_ln**,** v\_fn

**FROM** EMPLOYEES **INNER** **JOIN** DEPARTMENTS **ON** EMPLOYEES**.**EMPLOYEE\_ID **=** DEPARTMENTS**.**MANAGER\_ID

**WHERE** DEPARTMENTS**.**DEPARTMENT\_ID **=** v\_dep\_id**;**

**RETURN** v\_ln **||** ' '**||** v\_fn**;**

**END** FUNCTIONS03\_1\_GREILINGER**;**

## Erstellen Sie eine Funktion, die die MitarbeiterNr bekommt und die Anzahl seiner Jobs liefert (2L)

**create** **or** **replace** **FUNCTION** ANZAHL\_JOBS

**(**

P\_EMP **IN** employees**.**employee\_id**%type**

**)** **RETURN** **NUMBER** **AS**

v\_anzahl\_jobs **number;**

v\_emp employees**.**employee\_id**%type;**

**BEGIN**

v\_anzahl\_jobs **:=** 0**;**

**if** **(**p\_emp **is** **NULL)** **then**

**NULL;**

**else**

/\* hatte der Mitarbeiter bereits Jobwechsel? \*/

**select** **count(\*)**

**into** v\_anzahl\_jobs

**from** job\_history

**where** employee\_id **=** p\_emp**;**

**select** employee\_id

**into** v\_emp

**from** employees

**where** employee\_id **=** p\_emp**;**

v\_anzahl\_jobs **:=** v\_anzahl\_jobs +1**;**

**RETURN** v\_anzahl\_jobs**;**

**end** **if;**

**Exception**

**when** NO\_DATA\_FOUND **THEN**

RAISE\_APPLICATION\_ERROR**(**-20202**,** 'Employee '**||**p\_emp**||**' does not exist'**);**

**when** **others** **then**

**NULL;**

**END** ANZAHL\_JOBS**;**

**CREATE** **OR** **REPLACE** **FUNCTION** "5BHIF20167\_04"**.**"FUNCTIONS03\_2\_GREILINGER" **(**

emp\_id **IN** **INT**

**)** **RETURN** **VARCHAR2** **AS**

v\_emp\_name employees**.**first\_name**%TYPE;**

v\_numb **INT;**

**BEGIN**

**SELECT**

EMPLOYEES**.**FIRST\_NAME**,**

**COUNT(**JOB\_HISTORY**.**EMPLOYEE\_ID**)**

**INTO** v\_emp\_name**,** v\_numb

**FROM**

EMPLOYEES **INNER** **JOIN** JOB\_HISTORY **ON** JOB\_HISTORY**.**EMPLOYEE\_ID **=** EMPLOYEES**.**EMPLOYEE\_ID

**WHERE**

EMPLOYEES**.**EMPLOYEE\_ID **=** emp\_id

**GROUP** **BY**

EMPLOYEES**.**FIRST\_NAME**;**

**RETURN** 'Employee ' **||** v\_emp\_name **||** ' <id: ' **||** emp\_id **||** '> number of jobs: ' **||** v\_numb**;**

**END** FUNCTIONS03\_2\_GREILINGER**;**

## Erstellen Sie eine Funktion, der die ManagerID mitgegeben wird. Diese Funktion soll eine Liste aller seiner Mitarbeiter liefern. (1L)

**CREATE** **OR** **REPLACE** **FUNCTION** "5BHIF20167\_04"**.**"FUNCTIONS03\_3\_GREILINGER" **(**

v\_man\_id **IN** **INT**

**)** **RETURN** **VARCHAR2** **AS**

v\_names **VARCHAR2(**200**);**

**BEGIN**

v\_names **:=** 'employees for manager with id: ' **||** v\_man\_id **||** ':'**;**

**FOR** emp\_name **IN**

**(**

**SELECT**

EMPLOYEES**.**LAST\_NAME

**FROM**

EMPLOYEES

**WHERE**

EMPLOYEES**.**MANAGER\_ID **=** v\_man\_id

**)**

**LOOP**

v\_names **:=** v\_names **||** ' ' **||** emp\_name**.**LAST\_NAME**;**

**END** **LOOP;**

**RETURN** v\_names**;**

**END** FUNCTIONS03\_3\_GREILINGER**;**

# Übungszettel 4 – Exceptions HR

## Ändern Sie das Gehalt des Mitarbeiters 129: (2L)

- auf das Gehalt des Mitarbeiters mit dem Vornamen 'Peter'.

- wenn 'Peter' nicht in der Datenbank gespeichert ist, dann ermitteln sie das Durchschnittsgehalt aller Mitarbeiter

- wenn es mehr als einen 'Peter' gibt, dann ermitteln Sie das kleinste Einkommen aller 'Peter'

Verwenden Sie Exceptions!

**CREATE** **OR** **REPLACE** **PROCEDURE** UE4\_A1 **AS**

v\_salary employees**.**salary**%TYPE;**

**BEGIN**

**select** **min(**salary**)**

**into** v\_salary

**from** EMPLOYEES

**where** FIRST\_NAME**=**'Peter'

**group** **by** first\_name**;**

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

**select** **avg(**salary**)**

**into** v\_salary

**from** EMPLOYEES**;**

**update** EMPLOYEES

**set** SALARY**=**v\_salary

**where** employee\_id**=**1**;**

**END** UE4\_A1**;**

**set** define **off;**

**CREATE** **OR** **REPLACE** **PROCEDURE** "5BHIF20167\_04"**.**"EXCEP\_04\_01" **AS**

v\_salary employees**.**salary**%TYPE;**

**BEGIN**

v\_salary **:=** EXCEP\_04\_01\_HELP**(**'Peter'**);**

--DBMS\_OUTPUT.PUT\_LINE(v\_salary);

**UPDATE** employees

**SET** salary **=** v\_salary

**WHERE** employee\_id **=** 129**;**

**END** EXCEP\_04\_01**;**

Dazugehörige Funktion

**CREATE** **OR** **REPLACE** **FUNCTION** "5BHIF20167\_04"**.**"EXCEP\_04\_01\_HELP"

**(**v\_emp\_fn **IN** EMPLOYEES**.**FIRST\_NAME**%TYPE)RETURN** EMPLOYEES**.**SALARY**%TYPE** **AS**

v\_salary EMPLOYEES**.**SALARY**%TYPE;**

**BEGIN**

**SELECT** salary **INTO** v\_salary

**FROM** employees

**WHERE** first\_name **=** v\_emp\_fn**;**

**RETURN** v\_salary**;**

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

--DBMS\_OUTPUT.PUT\_LINE('no data');

**SELECT** **AVG(**salary**)** **INTO** v\_salary

**FROM** employees**;**

**WHEN** TOO\_MANY\_ROWS **THEN**

--DBMS\_OUTPUT.PUT\_LINE('too many');

**SELECT** **MIN(**salary**)** **INTO** v\_salary

**FROM** employees

**WHERE** first\_name **=** v\_emp\_fn**;**

**RETURN** v\_salary**;**

**WHEN** **OTHERS** **THEN**

--DBMS\_OUTPUT.PUT\_LINE('!WARNING! OTHER EXCEPTION');

**RETURN** **null;**

**END** EXCEP\_04\_01\_HELP**;**

## Erstellen Sie eine Prozedur, der die Abteilungsnummer mitgegeben wird. Suchen Sie den Mitarbeiter mit dem höchsten Einkommen und machen Sie diesen zum Manager der Abteilung. (2L)

Programmieren Sie sicher und verwenden Sie Exceptions. Geben Sie Beispiele an, wo die Verarbeitung funktioniert und wo es Probleme gibt.

**CREATE** **OR** **REPLACE** **PROCEDURE** UE4\_A2 **AS**

v\_manager\_id employees**.**employee\_id**%type;**

v\_manager\_dep employees**.**department\_id**%type;**

**BEGIN**

**select** employee\_id**,** department\_id

**into** v\_manager\_id**,** v\_manager\_dep

**from** **(**

**select** employee\_id**,** department\_id

**from** EMPLOYEES

**where** FIRST\_NAME**=**'Peter'

**order** **by** salary**)**

**where** **ROWNUM** **=** 1**;**

**update** EMPLOYEES

**set** MANAGER\_ID**=**V\_MANAGER\_ID

**where** department\_id**=**v\_manager\_dep**;**

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

dbms\_output**.**put\_line**(**'No Data was returned'**);**

**RAISE;**

**WHEN** TOO\_MANY\_ROWS **THEN**

dbms\_output**.**put\_line**(**'Multiple Employees with the same salary has been found'**);**

**RAISE;**

**WHEN** **others** **THEN**

dbms\_output**.**put\_line**(**'An error occured'**);**

**RAISE;**

**END** UE4\_A2**;**

**set** define **off;**

**CREATE** **OR** **REPLACE** **PROCEDURE** "5BHIF20167\_04"**.**"EXCEP\_04\_02"

**(**v\_department\_id **IN** departments**.**department\_id**%TYPE)** **AS**

v\_man\_id employees**.**employee\_id**%TYPE;**

**BEGIN**

**SELECT** manager\_id **INTO** v\_man\_id

**FROM** departments

**WHERE** department\_id **=** v\_department\_id**;**

**IF** v\_man\_id **IS** **NULL** **THEN**

**UPDATE** departments

**SET** manager\_id **=** **(**

**SELECT** employee\_id

**FROM** employees

**WHERE** salary **=** **(**

**SELECT** **MAX(**salary**)**

**FROM** employees

**WHERE** department\_id **=** v\_department\_id

**)**

**)**

**WHERE** department\_id **=** v\_department\_id**;**

**ELSE**

DBMS\_OUTPUT**.**PUT\_LINE**(**'manager already set!'**);**

**END** **IF;**

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

DBMS\_OUTPUT**.**PUT\_LINE**(**'department not found or without employees!'**);**

**END** EXCEP\_04\_02**;**

# Übungszettel 5 – Trigger HR

## Stellen Sie sicher, daß an der Mitarbeitertabelle vor 5:00 und nach 23:00 keine Änderungen gemacht werden können. (2L)

**CREATE** **OR** **REPLACE** **TRIGGER** UE5\_A1

**BEFORE** **INSERT** **OR** **UPDATE** **OR** **DELETE** **ON** EMPLOYEES

**DECLARE**

oowhe **exception;**

**BEGIN**

**IF** **CURRENT\_TIMESTAMP** **<** '05:00' **OR** **CURRENT\_TIMESTAMP** **>** '23:00' **THEN**

**RAISE** oowhe**;**

**end** **if;**

**END;**

**CREATE** **OR** **REPLACE** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_1\_GR"

**BEFORE** **UPDATE** **OR** **INSERT** **OR** **DELETE** **ON** employees

**BEGIN**

**IF** **EXTRACT(hour** **FROM** SYSTIMESTAMP**)** **>** 23 **OR**

**EXTRACT(hour** **FROM** SYSTIMESTAMP**)<** 5 **THEN**

RAISE\_APPLICATION\_ERROR**(**-20201**,**'This employee can not be updated during 23:00 and 5:00.'**);**

**END** **IF;**

**END;**

**/**

**ALTER** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_1\_GR" DISABLE**;**

## Stellen Sie sicher, dass das Gehalt eines Mitarbeiters nicht verkleinert werden kann. (1L)

**CREATE** **OR** **REPLACE** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_2\_GR"

**BEFORE** **UPDATE** **OF** salary **ON** employees **FOR** **EACH** **ROW**

**BEGIN**

**IF** **:OLD.**salary **>** **:NEW.**salary **THEN**

RAISE\_APPLICATION\_ERROR**(**-20202**,**Gehalt geht nix kleiner!'**);**

**END** **IF;**

**END;**

**/**

**ALTER** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_2\_GR" DISABLE**;**

## Schreiben Sie einen Trigger, der sicher stellt, dass der Mitarbeiter und der Vorgesetzte zur gleichen Abteilung gehören. (1L)

**CREATE** **OR** **REPLACE** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_3\_GR"

**BEFORE** **INSERT** **OR** **UPDATE** **OF** MANAGER\_ID **ON** EMPLOYEES **FOR** **EACH** **ROW**

**DECLARE**

v\_dep\_id DEPARTMENTS**.**DEPARTMENT\_ID**%TYPE;**

department\_manager\_mismatch **EXCEPTION;**

**BEGIN**

-- change select depending on insert/update

**SELECT** DEPARTMENT\_ID **INTO** v\_dep\_id

**FROM** employees

**WHERE** employee\_id **=** **:new.**manager\_id**;**

-- nachlesen des gleichen datensatzes -> mutating trigger

**IF** v\_dep\_id **!=** **:new.**department\_id **THEN**

**RAISE** department\_manager\_mismatch**;**

**END** **IF;**

**EXCEPTION**

**WHEN** department\_manager\_mismatch **THEN**

RAISE\_APPLICATION\_ERROR**(**-20203**,** 'employee department and manager department mismatch'**);**

**END;**

**/**

**ALTER** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_3\_GR" ENABLE**;**

## Jedes Mal, wenn der Job eines Mitarbeiters geändert wird, soll eine Logging Information (job history) geschrieben werden: (1L)

- Employee ID, alte job ID, neue department ID, Einstellungsdatum, Austrittsdatum.

- wenn schon ein Logging Datensatz für Mitarbeiter vorhanden ist, dann soll das Eintrittsdatum auf Enddatum+1 gesetzt werden.

**CREATE** **OR** **REPLACE** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_4\_GR"

**AFTER** **UPDATE** **OF** JOB\_ID **ON** EMPLOYEES **FOR** **EACH** **ROW**

**DECLARE**

v\_jobs **number;**

v\_end **number;**

**BEGIN**

**select** **count(\*)** **into** v\_jobs

**from** job\_history

**where** employee\_id **=** **:old.**employee\_id**;**

**if** v\_jobs **>** 0 **then**

v\_end **:=** 1**;**

**elsif** v\_jobs **=** 0 **then**

v\_end **:=** 0**;**

**end** **if;**

**INSERT** **INTO** JOB\_HISTORY

**VALUES** **(:old.**employee\_id**,SYSDATE,SYSDATE+**v\_end**,:old.**job\_id**,** **:new.**department\_id**);**

**END;**

**/**

**ALTER** **TRIGGER** "5BHIF20167\_04"**.**"TRIGGER\_5\_4\_GR" ENABLE**;**

# PLÜFen vom letzten Schuljahr (Teil 1)

## Implmentieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie das Max\_Salary seiner Job\_ID (insert)

## Schreiben Sie einen Trigger, der die Email Adresse eines Mitarbeiters (beim Insert und falls leer) ändert auf <Vorname>,<Familienname>@<Department\_Name>“

## Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters zum Min- bzw. Max\_Salary seiner Job\_ID passt. Geben Sie dem Mitarbeiter eine Gehaltsreduktion um 5%, falls er mehr als das Max\_Salary verdient.

## Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann.

Analysieren Sie die Tabellen employees und job\_history. Sehen sie den Inhalt der Tabellen an!

# PLÜFen vom letzten Schuljahr (Teil 2)

## Implementieren Sie einen INSERT Trigger, der prüft ob MIN\_Salary und MAX\_Salary eines JOBS befüllt ist. Falls eines der beiden leer ist, setzten sie den vorhandenen Wert ein. Falls beide leer sind, erzeugen Sie eine entsprechende Exception.

## Erstellen Sie eine Prozedur, die überprüft ob das Gehalt eines Mitarbeiters grösser als das Durchschnittsgehalt seiner Kollegin in den anderen Abteilung ist. Geben Sie dem Mitarbeiter - in diesem Fall – eine Gehaltsreduktion um 5%

## Erstellen Sie eine Funktion, die die Daten Location/Department/Employees korrekt anlegen kann.

## Schreiben Sie eine Funktion, die für einen Mitarbeiter ermittelt, ob dieser einen Job Wechsel hatte. Falls das der Fall war, überprüfen Sie, ob sich der Job Title geändert hat. Geben Sie diese beiden Job Title durch „,“ getrennt von der Funktion zurück.

# PLÜF von der AHIF

## Implementieren Sie einen Trigger, der sicher stellt, dass ein Mitarbeiter genau so viel verdient, wie die Hälfte der Max\_Salary seiner Job\_ID (update)

## Schreiben Sie eine Funktion, die die Email Adresse eines Mitarbeiters (falls leer) ändert auf <Vorname>@<Familienname>.<Department\_Name>“

## Erstellen Sie eine Prozedur, die den Gehalts- Mittelwert ( (min+max)/2 ) einer gegebenen Job\_ID ermittelt. Suchen Sie einen Mitarbeiter mit der gegebenen JOB\_ID, dessen Salary möglichst gleich diesem Mittelwert ist.

## Erstellen Sie eine Prozedur die einen neuen Mitarbeiter anlegen kann bzw. einen Jobwechsel eines bestehenden Mitarbeiters durchführen kann.