Ausarbeitung Übung 04

Table of Contents

[Hierarchische Abfragen (HR) 1](#__RefHeading___Toc1377_1340376544)

[Hierarchische Abfragen (Sakila) 3](#__RefHeading___Toc1379_1340376544)

[PIVOT und UNPIVOT 4](#__RefHeading___Toc1381_1340376544)

[Analytische Abfragen 6](#__RefHeading___Toc1383_1340376544)

# Hierarchische Abfragen (HR)

*-- 1.1a*

**SELECT employee\_id**,

**last\_name**,

**hire\_date**,

**salary**

**FROM** employees

*-- basically WHERE manager\_id = 102;*

**START WITH manager\_id** = 102

**CONNECT BY PRIOR employee\_id** = **manager\_id**

**AND** *level* = 1;



*-- 1.1b*

**SELECT employee\_id**,

**last\_name**,

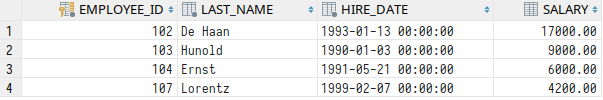
**hire\_date**,

**salary**

**FROM** employees

**START WITH employee\_id** = 102

**CONNECT BY PRIOR employee\_id** = **manager\_id**;



*-- 1.2*

**SELECT employee\_id**,

**last\_name**,

**hire\_date**,

**salary**

**FROM** employees

**WHERE** *level* = 3

**START WITH employee\_id** = 102

**CONNECT BY PRIOR employee\_id** = **manager\_id**;



*-- 1.3*

**SELECT employee\_id**,

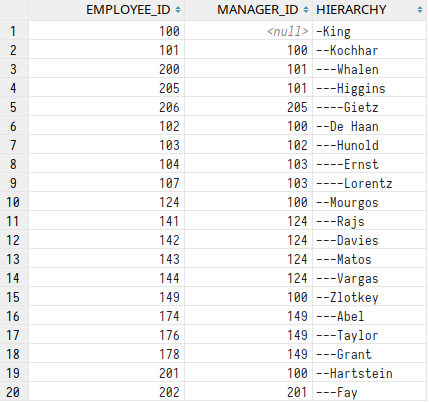
**manager\_id**,

*LPAD*(**last\_name**, *LENGTH*(**last\_name**) + *level*, **'-'**) **AS hierarchy**

**FROM** employees

**START WITH manager\_id IS NULL**

**CONNECT BY PRIOR employee\_id** = **manager\_id**;



*-- 1.4*

**WITH** superiors **AS** (**SELECT CONNECT\_BY\_ROOT first\_name AS** first\_name,

**CONNECT\_BY\_ROOT last\_name AS** last\_name

**FROM** employees

**WHERE** *level* > 1

**CONNECT BY PRIOR employee\_id** = **manager\_id**)

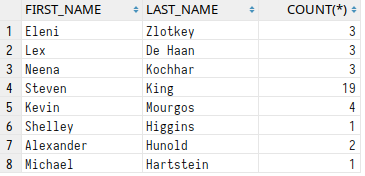
**SELECT** first\_name,

last\_name,

*COUNT*(*\**)

**FROM** superiors

**GROUP BY** first\_name, last\_name;



# Hierarchische Abfragen (Sakila)

*-- 2*

**CREATE OR REPLACE VIEW** partners **AS**

**SELECT** a1.**actor\_id AS** actor\_id,

a2.**actor\_id AS** partner\_id,

a1.**film\_id AS** film\_id

**FROM** film\_actor a1

**INNER JOIN** film\_actor a2 **ON** a1.**actor\_id** != a2.**actor\_id AND**

a1.**film\_id** = a2.**film\_id**

**WHERE** a1.**film\_id** <= 13;

*-- Nick Wahlberg is has actor\_id 2!*

**SELECT DISTINCT partner\_id,**

**last\_name**,

**first\_name**

**FROM** partners p

**INNER JOIN** actor **a ON** p.**partner\_id** = **a**.**actor\_id**

**WHERE partner\_id NOT IN**

(**SELECT partner\_id**

**FROM** partners

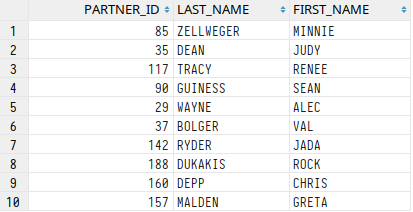
**WHERE actor\_id** = 2)

**START WITH** p.**actor\_id** = 2

**CONNECT BY NOCYCLE PRIOR partner\_id** = p.**actor\_id**

**AND partner\_id** != 2

**AND** *level* = 2;



# PIVOT und UNPIVOT

*-- 3.1*

**SELECT** *\**

**FROM**

(**SELECT staff\_id**,

**name AS category**

**FROM** rental

**INNER JOIN** inventory **USING** (**inventory\_id**)

**INNER JOIN** film **USING** (**film\_id**)

**INNER JOIN** film\_category **USING** (**film\_id**)

**INNER JOIN category USING** (**category\_id**))

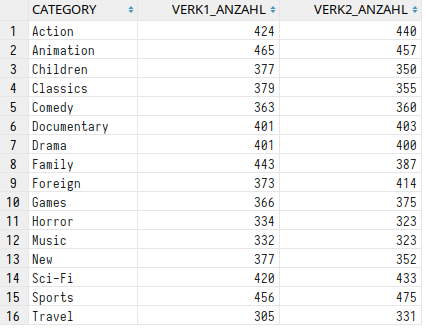
**PIVOT**

(*COUNT*(*\**) **AS** anzahl

**FOR staff\_id**

**IN** (1 **AS** verk1, 2 **AS** verk2))

**ORDER BY category ASC**;



*-- 3.2*

***SELECT name****,*

*ROUND*(family, 2),

*ROUND*(children, 2),

*ROUND*(animation, 2)

**FROM**

(**SELECT** lang.**name**,

cat.**name AS category**,

**length**

**FROM category** cat

**INNER JOIN** film\_category **USING** (**category\_id**)

**INNER JOIN** film **USING** (**film\_id**)

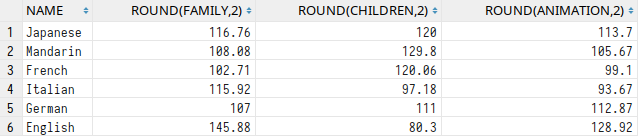
**INNER JOIN language** lang **USING** (**language\_id**))

**PIVOT**

(*AVG*(**length**)

**FOR category**

**IN** (**'Family' AS** family, **'Children' AS** children, **'Animation' AS** animation));

**

*-- 3.3*

**SELECT** *\**

**FROM**

(**SELECT title**,

**l**.**name AS** lang,

ol.**name AS** original

**FROM** film f

**INNER JOIN language l ON** f.**language\_id** = **l**.**language\_id**

**INNER JOIN language** ol **ON** f.**original\_language\_id** = ol.**language\_id**

**WHERE release\_year** = 1983)

**UNPIVOT**

(**language**

**FOR kind**

**IN** (lang **AS 'L'**, original **AS 'OL'**))

**ORDER BY title ASC**;



# Analytische Abfragen

*-- 4.1*

**SELECT title**,

**rental\_date**,

*RANK*() **OVER** (**PARTITION BY title ORDER BY rental\_date**) **AS** rank

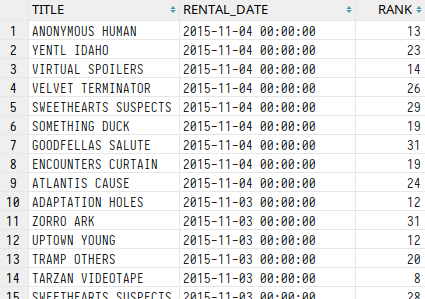
**FROM** rental

**INNER JOIN** inventory **USING** (**inventory\_id**)

**INNER JOIN** film **USING** (**film\_id**)

**ORDER BY rental\_date DESC**

**FETCH FIRST** 10 **ROWS WITH TIES**;



*-- 4.2*

**SELECT name**,

**title**,

**release\_year**

**FROM**

(**SELECT name**,

**title**,

**release\_year**,

*ROW\_NUMBER*() **OVER** (**PARTITION BY category\_id ORDER BY release\_year**) **AS** rn

**FROM** film\_category

**INNER JOIN** film **USING** (**film\_id**)

**INNER JOIN category USING** (**category\_id**))

**WHERE** rn < 4;



*-- 4.3*

*-- I tried to partition by customer id*

*-- and then let a window of size 2 compare*

*-- 2 adjacent dates recursively but calculating*

*-- the date difference turned out to be quite*

*-- tricky so I used regular joins. :)*

**WITH** dates **AS** (**SELECT customer\_id**,

**rental\_date**,

*ROW\_NUMBER*() **OVER** (**PARTITION BY customer\_id ORDER BY rental\_date ASC**) **AS** rn

**FROM** rental)

**SELECT** d1.**customer\_id**,

**c**.**last\_name**,

*ROUND*(*AVG*(d2.**rental\_date** - d1.**rental\_date**)) **AS** average\_rental\_interval

**FROM** dates d1

**INNER JOIN** dates d2 **ON** d1.**customer\_id** = d2.**customer\_id AND**

d1.rn + 1 = d2.rn *-- CONNECT BY (...) AND PRIOR rn + 1 = rn*

**INNER JOIN** customer **c ON** (d1.**customer\_id** = **c**.**customer\_id**)

**GROUP BY** d1.**customer\_id**, **c**.**last\_name**;

