DES3UE Datenbanksysteme WS 2018 Übung 4

Abgabetermin: 31.10.2018, 13:30 Uhr
Abgabeform elektronisch

☒ **DES31UE Niklas** **Name** PAPESH Konstantin **Aufwand in h** 6
☐ **DES32UE Niklas**
☐ **DES33UE Traxler** **Punkte** _____ **Kurzzeichen Tutor** _____

Hinweise und Richtlinien:

- Übungsausarbeitungen müssen den im elearning angegebenen Formatierungsrichtlinien entsprechen – Nichtbeachtung dieser Formatierungsrichtlinien führt zu Punkteabzug.
  - Treffen Sie, falls notwendig, sinnvolle Annahmen und dokumentieren Sie diese nachvollziehbar in ihrer Lösung!

Ziel dieser Übung ist die Formulierung von hierarchischen Abfragen in SQL, die Manipulation von Zeilen und Spalten mit den Befehlen PIVOT und UNPIVOT und der Einsatz von analytischen Abfragen.

1. Hierarchische Abfragen – Human Resources (5 Punkte – 1+1+1+1+1)

1. Lex De Haan verlässt das Unternehmen. Sein Nachfolger wünscht Berichte über die Angestellten, die De Haan direkt unterstellt sind. Erstellen Sie eine SQL-Anweisung, um die Angestelltennummer, den Nachnamen, das Einstellungsdatum und das Gehalt anzuzeigen, wobei auf Folgendes eingeschränkt werden soll:
   a. Die Angestellten, die De Haan direkt unterstellt sind
   b. Die Organisationsstruktur unter De Haan (Angestelltennummer: 102)
2. Erstellen Sie eine hierarchische Abfrage, um die Angestelltennummer, die Managernummer und den Nachnamen für alle Angestellten unter De Haan anzuzeigen, die sich genau zwei Ebenen unterhalb dieses Angestellten (De Haan, Angestelltennummer: 102) befinden. Zeigen Sie zudem die Ebene des Angestellten an. *(2 Zeilen)*
3. Der CEO benötigt einen hierarchischen Bericht über alle Angestellten. Er teilt Ihnen die folgenden Anforderungen mit: Erstellen Sie einen hierarchischen Bericht, der die Angestelltennummer, die Managernummer, die Pseudospalte `LEVEL` und den Nachnamen des Angestellten anzeigt. Für jede Zeile der Tabelle `EMPLOYEES` soll eine Baumstruktur ausgegeben werden, die den Angestellten, seinen Manager, dessen Manager usw. zeigt. Verwenden Sie Einrückungen (LPAD) für die Spalte `LAST_NAME`.
4. Erstellen Sie einen Bericht, der alle Angestellten enthält (Vor- und Nachname) und bestimmt, von wie vielen Personen er/sie Vorgesetzte/r ist. Zählen Sie nicht nur die direkt unterstellten, sondern die gesamte Hierarchie darunter (zB Steven King ist der Vorgesetzte von 19 Personen). Sie benötigen für diese Aufgabe eine spezielle Pseudo-Spalte, recherchieren Sie diese!

2. Hierarchische Abfragen – Sakila-Datenbank (6 Punkte)

Anmerkungen: Beachten Sie, dass für diese beiden Abfragen nur Filme mit einer ID <= 13 berücksichtigt werden. Für die Hierarchie soll die Bekanntschaftskette nicht innerhalb eines

Filmes durlaufen werden (zusätzliche Bedingung bei CONNECT BY). Verwenden Sie EXECUTE SYS.KILL_MY_SESSIONS() um hängen gebliebene Session/Abfragen zu beenden.

1. Für Schauspielerinnen wird ein soziales Netzwerk aufgebaut. Initial wird davon ausgegangen, dass sich Schauspielerinnen, die gemeinsam in einem Film spielen bereits kennen. Das System soll nun eine Liste von neuen Kontaktvorschlägen generieren, die darauf beruht, jemanden „über (genau) einen gemeinsamen Bekannten" zu kennen.

   Erstellen Sie eine View partners, die Ihnen Schauspielerinnen und Schauspieler-Kollegen ausgibt (wenn sie in irgendeinem Film miteinander gespielt haben). Achten Sie darauf, dass keine Beziehung der SchauspielerInnen auf sich selbst entsteht. Die View soll beide Actor-IDs und die Film-ID enthalten. Damit das Ergebnis überschaubar bleibt, sollen Sie hierfür nur die ersten 13 Filme (film_id <= 13) der Datenbank heranziehen. *(2 Punkte, 438 Zeilen)*

   Erstellen Sie aufbauend auf der obigen View eine Vorschlagsliste für Schauspieler „NICK WAHLBERG". *(4 Punkte, 10 Zeilen)*

3. PIVOT und UNPIVOT (6 Punkte – 2+2+2)

1. Erstellen Sie eine Pivot-Tabelle die angibt, welcher Verkäufer welche Film-Kategorien am besten verleihen kann. Erstellen Sie eine Spalte pro Verkäufer (berücksichtigen Sie nur Verkäufer 1 und 2) und listen Sie die Film-Kategorien zeilenweise auf. Zählen Sie die Anzahl der jeweils verliehenen Filme, sortieren Sie nach Kategorie. *(16 Zeilen)*

Kategorie	Verk1_Anzahl	Verk2_Anzahl
Action	424	440
Animation	465	457
…	…	…

2. Erstellen Sie eine Pivot-Tabelle für die Kategorien „Family", „Children" und „Animation" (Spalten), welche die Durchschnittslänge der Filme pro Sprache (Zeilen) angibt. *(6 Zeilen, 4 Spalten)*
3. Erstellen Sie eine Anfrage, die für jeden Film die Sprache und die Original-Sprache enthält (verwenden Sie nur Filme aus dem Jahr 1983). Erstellen Sie eine Tabelle, die als Spalten den Film-Namen, die Sprache und die Art der Sprache (L, OL) enthält, sortieren Sie nach Film-Titel. *(42 Zeilen)*

Titel	Art	Sprache
BORN SPINAL	L	Japanese
BORN SPINAL	OL	German
…	…	…

4. Analytische Abfragen (7 Punkte – 2+2+3)

Anmerkung: Verwenden Sie für die Lösung der folgenden Aufgaben analytische Funktionen.

1. Geben Sie Titel und Verleihdatum der zuletzt verliehenen Filme aus (zumindest zehn). Enthält das Verleihdatum gleiche Werte (Ties), geben Sie alle Filme mit diesem Datum aus. Ermitteln Sie dazu den Rang der Filme innerhalb des Verleihdatums. (*31 Zeilen)*
2. Geben Sie zu jeder Filmkategorie drei Filme aus, wählen Sie jene Filme, die neueren Datums sind. Geben Sie den Kategorienamen aus, den Filmtitel und das Erscheinungsjahr. Verwenden Sie ROW_NUMBER().
3. Ermitteln Sie, wie viele Tage ein Kunde durchschnittlich zwischen zwei aufeinanderfolgenden Verleihvorgängen verstreichen lässt, dh. wann er den nächsten Film ausleiht. Die Auswertung ist nur möglich, wenn der Kunde bereits mehr als einen Film ausgeborgt hat (beachten Sie NULL-Werte). *(zB Dana Hart leiht durchschnittlich alle 24 Tage einen Film aus)*

1.1a

```
SELECT employee_id, last_name, hire_date, salary
FROM employees
WHERE manager_id = 102;
```

EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
103	Hunold	1990-01-03 00:00:00	9000.00

1.1b

```
SELECT employee_id, last_name, hire_date, salary
FROM employees
START WITH manager_id = 102
CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	LAST_NAME	HIRE_DATE	SALARY
103	Hunold	1990-01-03 00:00:00	9000.00
104	Ernst	1991-05-21 00:00:00	6000.00
107	Lorentz	1999-02-07 00:00:00	4200.00

1.2

```
COLUMN org_chart FORMAT A25
SELECT employee_id, manager_id, last_name, LEVEL
FROM employees
WHERE LEVEL = 2
START WITH manager_id = 102
CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	MANAGER_ID	LAST_NAME	LEVEL
104	103	Ernst	2
107	103	Lorentz	2

1.3

```
SELECT employee_id, manager_id, LEVEL, LPAD(last_name, LENGTH(last_name) +
(LEVEL * 2) - 2, '_') AS last_name
FROM employees
CONNECT BY PRIOR employee_id = manager_id;
```

EMPLOYEE_ID	MANAGER_ID	LEVEL	LAST_NAME
101	100	1	Kochhar
200	101	2	__Whalen
205	101	2	__Higgins
206	205	3	____Gietz
201	100	1	Hartstein
202	201	2	__Fay
149	100	1	Zlotkey
174	149	2	__Abel
178	149	2	__Grant
176	149	2	__Taylor
124	100	1	Mourgos
141	124	2	__Rajs
144	124	2	__Vargas

56 Zeilen

1.4

```
SELECT manager_first_name, manager_last_name, COUNT(employee_id) AS
subordinates
FROM (SELECT employee_id,
             CONNECT_BY_ROOT first_name AS manager_first_name,
             CONNECT_BY_ROOT last_name  AS manager_last_name
      FROM employees
      WHERE level > 1
      CONNECT BY PRIOR employee_id = manager_id)
GROUP BY manager_first_name, manager_last_name
ORDER BY COUNT(employee_id) DESC;
```

MANAGER_FIRST_NAME	MANAGER_LAST_NAME	SUBORDINATES
Steven	King	19
Kevin	Mourgos	4
Lex	De Haan	3
Neena	Kochhar	3
Eleni	Zlotkey	3
Alexander	Hunold	2
Michael	Hartstein	1
Shelley	Higgins	1

2.1

```
CREATE OR REPLACE VIEW partners AS (
SELECT person.actor_id AS actor_id, partner.actor_id AS partner_id,
person.film_id
FROM film_actor person
      INNER JOIN film_actor partner ON person.film_id = partner.film_id AND
person.actor_id != partner.actor_id
WHERE person.film_id <= 13
                                  );

SELECT *
FROM partners;
```

```
SELECT first_name, last_name, actor_id
FROM actor
   WHERE first_name = 'NICK' AND last_name = 'WAHLBERG';

SELECT DISTINCT partner_id
FROM partners
WHERE partner_id NOT IN (SELECT partner_id
                         FROM partners
                        WHERE actor_id = 2)
START WITH actor_id = 2
CONNECT BY NOCYCLE PRIOR partner_id = actor_id AND level = 2 AND partner_id !=
2;
```

PARTNER_ID
29
157
90
85
117
35
37
188
142
160

3.1

```
SELECT *
FROM (SELECT rental_id, name, staff_id
     FROM category
            INNER JOIN film_category c2 ON category.category_id =
c2.category_id
            INNER JOIN film f ON c2.film_id = f.film_id
            INNER JOIN inventory i ON f.film_id = i.film_id
            INNER JOIN rental r ON i.inventory_id = r.inventory_id)
   PIVOT
   (
     COUNT(rental_id)
   FOR staff_id
   IN (1 AS Verk1_Anzahl,
     2 AS Verk2_Anzahl)
   )
ORDER BY name;
```

NAME	VERK1_ANZAHL	VERK2_ANZAHL
Action	424	440
Animation	465	457
Children	377	350
Classics	379	355
Comedy	363	360
Documentary	401	403
Drama	401	400
Family	443	387
Foreign	373	414
Games	366	375
Horror	334	323
Music	332	323
New	377	352

16 Zeilen

3.2

```
SELECT language, ROUND(family) AS Family, ROUND(children) AS Children,
ROUND(animation) AS Animation
FROM (SELECT c1.name, length, l.name AS language
     FROM category c1
            INNER JOIN film_category c2 ON c1.category_id = c2.category_id
            INNER JOIN film f ON c2.film_id = f.film_id
            INNER JOIN language l ON f.language_id = l.language_id)
   PIVOT
   (
     AVG(length)
   FOR name
   IN ('Family' AS Family,
     'Children' AS Children,
     'Animation' AS Animation)
   );
```

LANGUAGE	FAMILY	CHILDREN	ANIMATION
Japanese	117	120	114
Mandarin	108	130	106
French	103	120	99
Italian	116	97	94
German	107	111	113
English	146	80	129

3.3

```
SELECT *
FROM (SELECT title, ol.name AS ol, l.name AS l
     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
   (
```

```
      sprache
   FOR Art
   IN (L, OL)
   )
ORDER BY title;
```

TITLE	ART	SPRACHE
BORN SPINAL	OL	German
BORN SPINAL	L	Japanese
BOWFINGER GABLES	OL	German
BOWFINGER GABLES	L	Italian
BUNCH MINDS	L	English
BUNCH MINDS	OL	Mandarin
CHITTY LOCK	L	Mandarin
CHITTY LOCK	OL	German
CIDER DESIRE	OL	German
CIDER DESIRE	L	Italian
CLOSER BANG	L	Japanese
CLOSER BANG	OL	Mandarin
DIVIDE MONSTER	OL	German

42 Zeilen

4.1

```
SELECT title, rental_date, RANK() OVER (PARTITION BY rental_date ORDER BY
title) AS rank
     FROM film f
            INNER JOIN inventory i ON f.film_id = i.film_id
            INNER JOIN rental r ON i.inventory_id = r.inventory_id
     ORDER BY rental_date DESC
     FETCH FIRST 10 ROWS WITH TIES;
```

TITLE	RENTAL_DATE	RANK
ANONYMOUS HUMAN	2015-11-04 00:00:00	1
ATLANTIS CAUSE	2015-11-04 00:00:00	2
ENCOUNTERS CURTAIN	2015-11-04 00:00:00	3
GOODFELLAS SALUTE	2015-11-04 00:00:00	4
SOMETHING DUCK	2015-11-04 00:00:00	5
SWEETHEARTS SUSPECTS	2015-11-04 00:00:00	6
VELVET TERMINATOR	2015-11-04 00:00:00	7
VIRTUAL SPOILERS	2015-11-04 00:00:00	8
YENTL IDAHO	2015-11-04 00:00:00	9
ADAPTATION HOLES	2015-11-03 00:00:00	1
ARTIST COLDBLOODED	2015-11-03 00:00:00	2
BIRD INDEPENDENCE	2015-11-03 00:00:00	3
CHAMBER ITALIAN	2015-11-03 00:00:00	4

31 Zeilen

4.2

```
SELECT name, title, release_year
FROM (
   SELECT name, title, release_year, ROW_NUMBER() OVER(PARTITION BY name ORDER
BY release_year DESC) AS rn
```

```
    FROM film f
           INNER JOIN film_category fc ON f.film_id = fc.film_id
           INNER JOIN category c ON fc.category_id = c.category_id
    )
WHERE rn <= 3;
```

NAME	TITLE	RELEASE_YEAR
Action	CLUELESS BUCKET	2008
Action	LORD ARIZONA	2008
Action	ANTITRUST TOMATOES	2008
Animation	WAIT CIDER	2008
Animation	ANACONDA CONFESSIONS	2007
Animation	SUGAR WONKA	2007
Children	CROOKED FROGMEN	2008
Children	GRADUATE LORD	2008
Children	GORGEOUS BINGO	2007
Classics	JEOPARDY ENCINO	2008
Classics	STEEL SANTA	2008
Classics	SNATCHERS MONTEZUMA	2008
Comedy	HEAVEN FREEDOM	2008

48 Zeilen

4.3

```
SELECT first_name, last_name, ROUND(AVG(rental_date - date_before)) AS
days_between
FROM (
SELECT first_name, last_name, rental_date,
LAG(rental_date, 1, NULL) OVER(PARTITION BY first_name, last_name ORDER BY
rental_date ASC) AS date_before
FROM rental r INNER JOIN customer c USING(customer_id))
GROUP BY first_name, last_name
ORDER BY last_name ASC;
```

FIRST_NAME	LAST_NAME	DAYS_BETWEEN
RAFAEL	ABNEY	33
NATHANIEL	ADAM	25
KATHLEEN	ADAMS	27
DIANA	ALEXANDER	24
GORDON	ALLARD	22
SHIRLEY	ALLEN	21
CHARLENE	ALVAREZ	26
LISA	ANDERSON	27
JOSE	ANDREW	25
IDA	ANDREWS	29
OSCAR	AQUINO	36
HARRY	ARCE	17
JORDAN	ARCHULETA	22

599 Zeilen