Proseminar Datenbanksysteme

Universität Innsbruck — Institut für Informatik Frontull S., Ipek Y., Mayerl M., Vötter M., Zangerle E.



21.11.2019

Übungsblatt 6 - Lösungsvorschlag

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

a) ★ Wie werden Sets und Tuples aus der relationalen Algebra in einem relationalen Datenbanksystem umgesetzt?

Lösung



Sets werden in Form von Tabellen (Tables) repräsentiert und die Tupel in den Sets sind dann die Zeilen (Rows) in den Tabellen.

- 1 SELECT *
- 2 FROM Player, Matches
- 3 INNER JOIN Club
- 4 ON Player.clubId = Club.id
- 5 WHERE country = 'Grenada'
- 6 AND Matches.player1Id = Player.id

Gibt es Bedingungen, weshalb die relationale Algebra Query ein anderes Ergebnis liefern könnte als die gegebene SQL Query? Falls ja, könnten Sie die SQL Query anpassen, damit das nicht der Fall ist?

Lösung



SQL ist deklarativ bzw. es wird die Abfrage dahingehen optimiert, dass zuerst die WHERE Clause bzw. die Selektion und dann der Join durchgeführt wird. Kreuzprodukte werden vom Optimierer (soweit möglich) in Joins übersetzet. DISTINCT macht den Output unique, es muss eventuell eine Duplikatsuche durchgeführt werden.

- sigma (Club.country = 'Grenada') Club
- join (p1.clubId = Club.id) (rho p1 Player)
- 3 join (p1.id = Matches.player1Id) Matches
- c) \blacksquare Übersetzen Sie die gegebene Abfrage in relationaler Algebra in eine SQL Abfrage.

```
Customer (CustomerId, FirstName, LastName, Address, Email) Invoice (InvoiceId, CustomerId, InvoiceDate, Total)  \frac{\pi_{\text{InvoiceId,InvoiceDate,Total,LastName}}{\sigma_{\text{InvoiceDate}>'2012-01-01'\land\text{InvoiceDate}<'2012-12-31'}} \text{(Invoice)} \\ \bowtie_{\text{Invoice.CustomerId}=\text{Customer.CustomerId}} \text{(Customer)})
```

```
Lösung
    SELECT
                   InvoiceId,
2
                   InvoiceDate,
3
                   Total,
                   LastName
4
                   Invoice
5
    FROM
                   Customer USING (CustomerId)
6
    INNER JOIN
    WHERE
                   InvoiceDate BETWEEN '2012-01-01' AND '2012-12-31'
```

d) RDBS verwenden eine dreiwertige (ternäre) Logik um zu kennzeichnen, dass aktuelle Werte eines Attributes nicht bekannt sind. Führen Sie die gegebene SQL Abfrage "händisch" aus und vervollständigen Sie die result Tabelle anhand der dreiwertigen Logik.

1	SELECT	id,
2		age
3	INTO	result
4	FROM	person
5	WHERE	age = 0:

id	 age
1	 31
2	 10
3	 0
4	 NULL
5	 31

Tabelle 1: person

id	age

Tabelle 2: result

Lösung

Die Ergebnismenge enthält das Tupel < 3, 0 >.

id	 age
1	 31
2	 10
3	 0
4	 NULL
5	 31

Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

Mit dem auf Python 3 basierenden Tool check-files. py (siehe Ordner Kursinformation) und der Datei sheet_06_required_files. yml können Sie überprüfen, ob Sie alle Dateien richtig benannt haben und ob alle geforderten Dateien vorhanden sind. Führen Sie dazu folgenden Befehl im Verzeichnis mit den abzugebenden Dateien aus:

python3 check-files.py sheet_06_required_files.yml

Hinweis

A

Es ist möglich, dass Sie den Befehl anpassen müssen. Der gegebene Befehl ist für ein Linux-System und geht davon aus, dass check-files.py und sheet_06_required_files.yml im aktuellen Verzeichnis liegen.

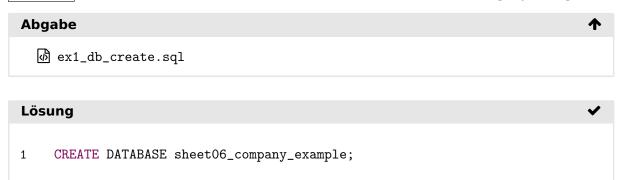
Aufgabe 1 (SQL DDL)

[3 Punkte]

In dieser Aufgabe werden 3 Relationen (Employee, Working, Project) mit Hilfe von SQL in einer Datenbank angelegt.

Verwenden Sie dafür das in Blatt 1 aufgesetzte DBMS und einen SQL-Client ihrer Wahl und stellen Sie sicher, dass Ihre abgegebenen SQL-Dateien auf Postgres 11.5 ausgeführt werden können.

a) | 0.5 Punkte | Erstellen Sie mittels SQL-Statement die Datenbank sheet06_company_example.



b) 1 Punkt Schreiben Sie SQL-Statements, die die folgenden drei Relationen in einer Datenbank anlegen.

```
employee (employee_id, firstname, lastname, country)
project (project_id, name, country)
working (employee_id, project_id, start_date)
```

Beachten Sie dabei auch, dass die Fremdschlüssel richtig referenziert werden. Für Textspalten reicht es aus, wenn 255 Zeichen gespeichert werden können.

```
Abgabe

© ex1_table_create.sql
```

```
Lösung
     CREATE TABLE project (
2
       project_id SERIAL PRIMARY KEY,
       name VARCHAR(255),
3
       country VARCHAR(255)
4
5
     );
6
7
     CREATE TABLE employee (
8
       employee_id SERIAL PRIMARY KEY,
       firstname VARCHAR(255),
9
10
       lastname VARCHAR(255),
       country VARCHAR(255)
11
12
     );
13
14
     CREATE TABLE working (
       employee_id INT NOT NULL REFERENCES employee(employee_id),
15
       project_id INT NOT NULL REFERENCES project(project_id),
16
       start_date TIMESTAMP
17
18
     );
```

c) 0.5 Punkte Fügen Sie in die Relationen Employee und Project den Mitarbeiter Max Mustermann und das Projekt project2 ein. Fügen Sie weiters mindestens drei weitere Mitarbeiter und drei weitere Projekte mit sinnvollen Testdaten ein.

```
Abgabe

© ex1_table_insert.sql
```

```
Lösung

1   INSERT INTO employee (firstname, lastname, country)
2   VALUES
3   ('Donald', 'Duck', 'USA'),
4   ('Max', 'Mustermann', 'Austria'),
5   ('Benjamin', 'Murauer', 'Austria');
```

d) I Punkt Finden Sie einen Weg, um einen Eintrag in die Relation working einzufügen, die den Mitarbeiter Max Mustermann mit dem Projekt project2 verknüpft, und speichern Sie als Datum den 02.02.2019. Greifen Sie dabei nicht auf fest kodierte IDs der betroffenen Tupel aus employee und project zurück, sondern lesen sie diese aus der Datenbank aus.



```
INSERT INTO working (employee_id, project_id, start_date)
VALUES (

(SELECT employee_id FROM employee
WHERE firstname = 'Max' AND lastname = 'Mustermann'),
(SELECT project_id FROM project WHERE name = 'project2'),
DATE('2019-02-02')
);
```

Aufgabe 2 (SQL DQL)

[7 Punkte]

Laden Sie Ihre SQL-Statements und die Ergebnisse in einem Textfile (UTF-8 kodiert) in OLAT hoch. Beachten Sie dabei die geforderten Namen der Dateien. Nützen Sie ab und zu die Umbenennungsfunktion für Ergebnisspalten (zur besseren Les- und Erkennbarkeit der Ergebnisse). **Halten Sie sich unbedingt an die in der Aufgabenstellung angegebene Reihenfolge der Ergebnisspalten. Wenn Sie die Spalten in der falschen Reihenfolge ausgeben, werden Ihre Ergebnisse von unserem Bewertungs-Skript als falsch gewertet.** Die Aufgaben sollten auf der Pagila Datenbank¹ ausgeführt werden. Deshalb müssen Sie als erstes das ZIP-File der Datenbank (Link in Fußnote) herunterladen und entpacken. Importieren Sie anschließend pagila-schema. sql und pagila-insert-data. sql in ihren Postgres-Server.

 $^{^{1} \}verb|https://github.com/devrimgunduz/pagila/archive/2.0.1.zip|$

Verwenden Sie dafür das in Blatt 1 aufgesetzte DBMS und einen SQL-Client Ihrer Wahl und stellen Sie sicher, dass Ihre abgegebenen SQL-Dateien auf Postgres 11.5 ausgeführt werden können.

a) 0.5 Punkte Geben Sie den Titel und das Erscheinungsjahr aller Filme aus, in denen ein Schauspieler mit dem Nachnamen CAGE mitgespielt hat.



Lös	Lösung ✓						
Qu	Query						
1 SELECT title,			,				
2		releas	se_year				
3	FROM	actor					
4	INNER JOIN	film_a	actor				
5	ON		.actor_id = film_actor.actor_id				
6	INNER JOIN	film	.debbl_id				
7	ON		actor.film_id = film.film_id				
8	WHERE	actor.	.last_name = 'CAGE'				
Re	sult						
	title		release_year				
	 NYON STOCK		+				
	NCES NONE		2006				
	CINO ELF		2006				
EN	DING CROWDS		2006				
GA	NDHI KWAI		2006				
	DFATHER DIARY		2006				
	NDICAP BOONDOCE	ζ	2006				
	NEY TIES		2006				
HORN WORKING			2006				
IMAGE PRINCESS			2006				
JERSEY SASSY			2006				
LOSER HUSTLER			2006				
MEET CHOCOLATE			2006 2006				
MOD SECRETARY MOONWALKER FOOL			2006				
OLEANDER CLUE			2006				
RACER EGG			2006				
	ORY SIDE		2006				
STRANGERS GRAFFITI			2006				
THIN SAGEBRUSH			2006				
	OTSIE PILOT		2006				
UF	TOWN YOUNG		2006				
VELVET TERMINATOR			2006				

WEST LION	1	2006		
WORKER TARZAN	1	2006		
ACADEMY DINOSAUR	1	2006		
ALAMO VIDEOTAPE	1	2006		
ARABIA DOGMA	1	2006		
BUNCH MINDS	1	2006		
CATCH AMISTAD	1	2006		
CLYDE THEORY	1	2006		
CONNECTICUT TRAMP		2006		
DESIRE ALIEN	1	2006		
DISCIPLE MOTHER	1	2006		
FLYING HOOK	1	2006		
GRAFFITI LOVE	1	2006		
HAMLET WISDOM	1	2006		
HANGING DEEP	1	2006		
INSTINCT AIRPORT	1	2006		
INTOLERABLE INTENTIONS	5	2006		
KARATE MOON	1	2006		
LIES TREATMENT	1	2006		
REIGN GENTLEMEN		2006		
ROCK INSTINCT	1	2006		
ROOTS REMEMBER	1	2006		
ROXANNE REBEL	1	2006		
RUSHMORE MERMAID	1	2006		
SIMON NORTH	1	2006		
SPY MILE	1	2006		
SUPERFLY TRIP	1	2006		
SUSPECTS QUILLS	1	2006		
THIEF PELICAN	1	2006		
VAMPIRE WHALE	1	2006		
VELVET TERMINATOR	1	2006		
54 rows)				

b) 0.5 Punkte Geben Sie den Titel all jener Filme aus, die in der Category New sind und weniger als 15.00 kosten, wenn man den Film ersetzten muss.





```
4
    ON
                   film.film_id = film_category.film_id
5
    INNER JOIN
                   category
6
                   film_category.category_id = category.category_id
7
    WHERE
                   category.name = 'New'
    AND
                   film.replacement_cost < 15.00</pre>
Result
        title
 AMISTAD MIDSUMMER
 APOCALYPSE FLAMINGOS
 ATTRACTION NEWTON
 BOULEVARD MOB
 EAGLES PANKY
 ENDING CROWDS
 FRONTIER CABIN
 GODFATHER DIARY
 HOURS RAGE
 JUMANJI BLADE
 JUNGLE CLOSER
 MAIDEN HOME
 MINE TITANS
 NUTS TIES
 PLATOON INSTINCT
 PLUTO OLEANDER
 STING PERSONAL
 STOCK GLASS
 VAMPIRE WHALE
 VARSITY TRIP
(20 rows)
```

c) 0.5 Punkte Geben Sie die E-Mail Adresse aller Kunden aus, die in einem Land leben, dessen Name mit Au beginnt.





```
4
    ON
                   customer.address_id = address.address_id
5
    INNER JOIN
                   city
6
                   address.city_id = city.city_id
    INNER JOIN
7
                   country
8
    ON
                   city.country_id = country.country_id
                   country.country LIKE 'Au%'
9
    WHERE
Result
              email
 AUDREY.RAY@sakilacustomer.org
 JILL.HAWKINS@sakilacustomer.org
 NORA.HERRERA@sakilacustomer.org
(3 rows)
```

d) 0.5 Punkte Geben Sie den Vor- und Nachnamen aller Kunden an, die einen Film am 14.06.2005 bei dem Mitarbeiter, dessen Benutzernamen Mike ist, ausgeliehen haben. Sie können dafür die Date/Time Functions and Operations von Postgres verwenden.

```
Abgabe

© ex2_customer_rental.sql

= ex2_customer_rental_result.txt
```

```
Lösung
Query
    SELECT
                  customer.first_name,
2
                  customer.last_name
3
    FROM
                  customer
    INNER JOIN
4
                  rental
5
    ON
                  customer.customer_id = rental.customer_id
    INNER JOIN
6
                  staff
7
    ON
                  rental.staff_id = staff.staff_id
                  staff.username = 'Mike'
8
    WHERE
    AND
                  DATE(rental_rental_date) = '2005-06-14'
Result
 first_name | last_name
_____
 ELMER
            I NOE
 MIRIAM
            | MCKINNEY
 DANIEL
            | CABRAL
```

²https://www.postgresql.org/docs/11/functions-datetime.html

```
TERRANCE | ROUSH

JOYCE | EDWARDS

CATHERINE | CAMPBELL

HERMAN | DEVORE

CHARLES | KOWALSKI

(8 rows)
```

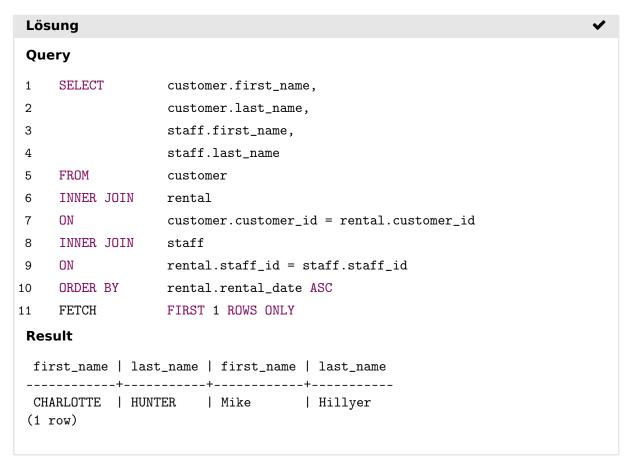
e) 1 Punkt Geben Sie die E-Mail Adresse aller Kunden aus, die im selben Land leben, wie der Mitarbeiter bei dem sie einen Film ausgeliehen haben. Achten Sie darauf, dass jede E-Mail Adresse im Ergebnis nur einmal vorkommt.



Lö	Lösung ✓					
Qu	Query					
1	SELECT DISTINCT	customer.email				
2	FROM	customer				
3	INNER JOIN	address				
4	ON	<pre>customer.address_id = address.address_id</pre>				
5	INNER JOIN	city				
6	ON	address.city_id = city.city_id				
7	INNER JOIN	rental				
8	ON	<pre>customer.customer_id = rental.customer_id</pre>				
9	INNER JOIN	staff				
10	ON	rental.staff_id = staff.staff_id				
11	INNER JOIN	address as staff_address				
12	ON	<pre>staff.address_id = staff_address.address_id</pre>				
13	INNER JOIN	city as staff_city				
14	ON	<pre>staff_address.city_id = staff_city.city_id</pre>				
15	WHERE	<pre>city.country_id = staff_city.country_id</pre>				
d R	Result					
	email					
DA DE LO CU	ROY.QUIGLEY@sakilac ARRELL.POWER@sakila ERRICK.BOURQUE@saki DRETTA.CARPENTER@sa JRTIS.IRBY@sakilacu rows)	customer.org lacustomer.org kilacustomer.org				

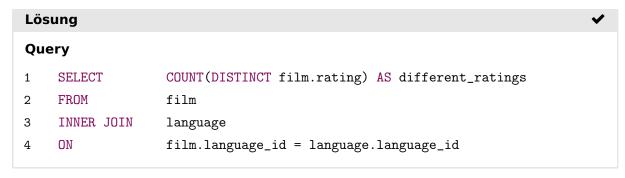
f) 1 Punkt Geben Sie den Vor- und Nachnamen für den Kunden an, der als erster einen Film ausgeliehen hat. Geben Sie auch den Vor- und Nachnamen des zuständigen Mitarbeiters aus.





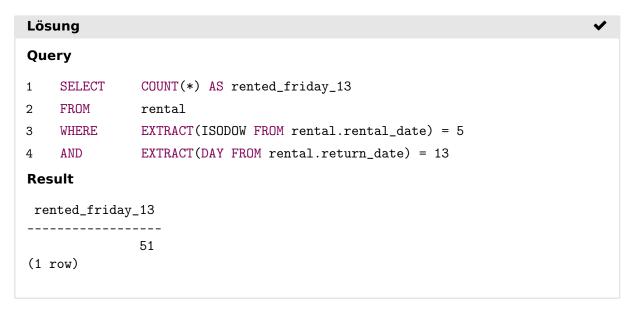
g) 1 Punkt Geben Sie die Anzahl verschiedener Ratings aus, die für Filme English vergeben wurden.





h) 1 Punkt Geben Sie die Anzahl an ausgeliehenen Filmen an, die an einem Freitag den 13. ausgeliehen wurden. Benutzen Sie dafür die Date/Time Functions and Operations³ von Postgres.





i) 1 Punkt Geben Sie den Namen aller Sprachen aus, für die es keine Filme gibt.





 $^{^{3} \}texttt{https://www.postgresql.org/docs/11/functions-datetime.html}$

```
2
    FROM
                 language
3
    LEFT JOIN
                 film
4
                 language.language_id = film.language_id
    ON
5
    WHERE
                 film.film_id IS NULL
Result
        name
_____
Italian
French
German
Mandarin
 Japanese
(5 rows)
```

Wichtig: Laden Sie bitte Ihre Lösung in OLAT hoch und geben Sie mittels der Ankreuzliste auch unbedingt an, welche Aufgaben Sie gelöst haben. Die Deadline dafür läuft am Vortag des Proseminars um 23:59 (Mitternacht) ab.