Proseminar Datenbanksysteme

Universität Innsbruck — Institut für Informatik



Bottesch R., Hupfauf B., Kelter C., Mayerl M., Moosleitner M., Peintner A., Zangerl D.

15.11.2022

Übungsblatt 5 – Lösungsvorschlag

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

a) wie werden Relationen und Tupel aus der relationalen Algebra in einem relationalen Datenbanksystem dargestellt?

Lösung



Relationen werden als Tabellen (Tables) dargestellt und Tupel (die Elemente der Relationen) sind dann die Zeilen (Rows) der Tabellen.

b) | ** Übersetzen Sie das unten angeführte SQL-Statement in die relationale Algebra

```
SELECT
1
2
    FROM
                    Player, Club AS c
    INNER JOIN
3
                    Event
    ON
4
                    Event.playerId = Player.id
                    c.country = 'Grenada'
    WHERE
5
                    Player.clubId = c.id
6
    AND
```

Lösung



"Wortwörtliche" Übersetzung:

```
sigma (c.country = 'Grenada') and (Player.clubId = c.id)

(
(Player cross join (rho c Club))
join Event.playerId = Player.id Event
)
```

SQL ist deklarativ bzw. es wird die Abfrage dahingehend optimiert, dass zuerst die WHERE Clause bzw. die Selektion und dann der Join durchgeführt wird. Kreuzprodukte werden vom Optimierer (soweit möglich) in Joins übersetzt. Deshalb ist die exakte Übersetzung die folgende:

```
sigma (c.country = 'Grenada') (rho c Club)

join (Player.clubId = c.id) Player
```

join (Player.id = Event.playerId) Event

c) dibt es Fälle, in denen eine Abfrage aus der relationalen Algebra ein anderes Ergebnis als liefert die entsprechende SQL-Abfrage? Wie kann die SQL-Abfrage angepasst werden, damit die Ergebnisse übereinstimmen? *Hinweis:* In der relationalen Algebra ist das Ergebnis einer Abfrage eine Relation, also eine Menge im mathematischen Sinn.

Lösung

~

In SQL können Ergebnisse Duplikate enthalten, die bei der relationale Algebra Query entfernt werden würden. Mit dem DISTINCT Befehl können in SQL Duplikate entfernt werden.

Als Beispiel sei die folgende Relation **Person** gegeben:

id	name
1	Anna
2	Oskar
3	Anna

Die Abfrage $\pi_{name} Person$ würde zu diesem Ergebnis führen:



Die "entsprechende" SQL Query

- 1 SELECT name
- 2 FROM Person

würde allerdings dieses Ergebnis liefern:



In diesem Falle müsste die SQL Query mit DISTINCT ergänzt werden um ein äquivalentes Ergebnis zu erzielen:

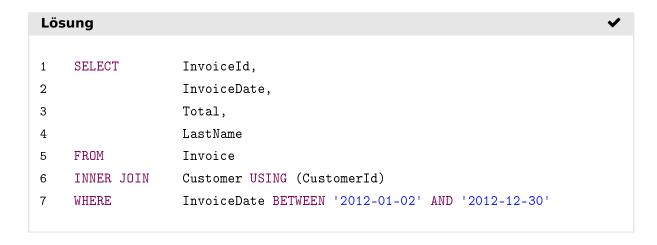
- 1 SELECT DISTINCT name
- 2 FROM Person

Invoice (<u>InvoiceId</u>, CustomerId, InvoiceDate, Total)

 π InvoiceId,InvoiceDate,Total,LastName

 $(\sigma_{\text{InvoiceDate}} > '2012 - 01 - 01' \land \text{InvoiceDate} < '2012 - 12 - 31')$ (Invoice))

M_{Invoice.CustomerId=Customer.CustomerId} (Customer))



e) st es in SQL möglich Tabellen zu erstellen, wo mehrere Spalten die identische Bezeichnung haben?

Lösung

•

Nein, ist nicht möglich (ERROR: column "column1" specified more than once, SQL state: 42701).

f) st es in SQL möglich eine Abfrage zu schreiben, in deren Ergebnis mehrere Spalten die identische Bezeichnung haben?

Lösung Ja, das ist möglich. Ein Beispiel wäre: 1 SELECT first_name AS name, last_name AS name 2 FROM actor

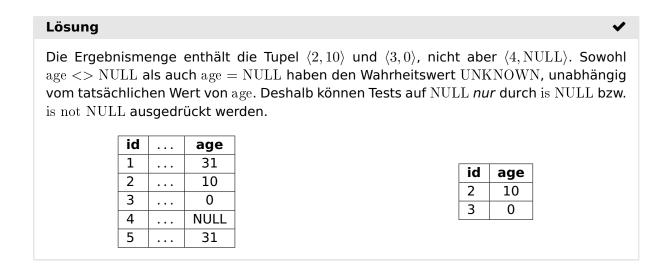
- - 1 SELECT id, age
 - 2 INTO result
 - 3 FROM person
 - 4 WHERE age <> 31;

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

id	age

Tabelle 1: person

Tabelle 2: result



Hausaufgabenteil (Zuhause zu lösen; Abgabe nötig)

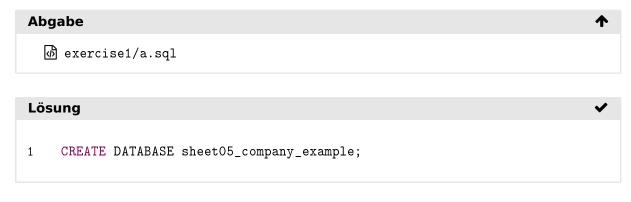
Aufgabe 1 (SQL DDL)

[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 PostgreSQL 13.4 ausgeführt werden können.

a) | 0.5 Punkte | Erstellen Sie mittels SQL-Statement die Datenbank sheet05_company_example.



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

```
employee (employee_id, firstname, lastname, main_location)
project (project_id, name, main_location)
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.



```
Lösung
     CREATE TABLE project (
1
2
       project_id SERIAL PRIMARY KEY,
3
       name VARCHAR(255),
4
       main_location VARCHAR(255)
5
     );
6
7
     CREATE TABLE employee (
8
       employee_id SERIAL PRIMARY KEY,
9
       firstname VARCHAR(255),
10
       lastname VARCHAR(255),
       main_location VARCHAR(255)
11
12
     );
13
14
     CREATE TABLE working (
15
       employee_id INT REFERENCES employee(employee_id),
       project_id INT REFERENCES project(project_id),
16
17
       start_date TIMESTAMP,
       CONSTRAINT pk_working PRIMARY KEY (employee_id, project_id)
18
19
     );
```

c) O.5 Punkte Fügen Sie in die Relationen Employee und Project die Mitarbeiterin Erika Mustermann und das Projekt project 2 ein. Fügen Sie weiters mindestens zwei weitere Mitarbeiter und zwei weitere Projekte mit sinnvollen Testdaten ein.



```
Lösung
1
     INSERT INTO employee (firstname, lastname, main_location)
2
     VALUES
3
       ('Donald', 'Duck', 'Chicago'),
4
       ('Erika', 'Mustermann', 'Innsbruck'),
       ('Benjamin', 'Murauer', 'Innsbruck');
5
6
7
     INSERT INTO project (name, main_location)
8
     VALUES
       ('project1', 'Chicago'),
9
       ('project2', 'Berlin'),
10
       ('project3', 'Innsbruck');
11
```

d) 1 Punkt Konstruieren Sie ein SQL-Statement, das folgenden Eintrag in die Relation working einfügt: employee_id ist die ID der Mitarbeiterin Erika Mustermann, projekt_id ist die ID des Projekts projekt2 und start_date ist 11.11.2021. Die IDs sollen dabei in dem SQL-Statement nicht fest kodiert sein, sondern aus der Datenbank gelesen werden.

Hinweis

Sie können innerhalb von INSERT-Statements auch SELECT-Statements verwenden.



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

(SELECT employee_id FROM employee
WHERE firstname = 'Erika' AND lastname = 'Mustermann'),
(SELECT project_id FROM project WHERE name = 'project2'),
DATE('2021-11-11')
);
```

Aufgabe 2 (SQL DQL)

[7 Punkte]

Bei den folgenden Aufgaben sollten Sie jeweils Ihr SQL-Statement sowie das Ergebnis als Textdatei abgeben. Halten Sie sich unbedingt an die in der Aufgabenstellung angegebene Reihen-

folge und Bezeichnung der Ergebnisspalten. Verwenden Sie, wenn notwendig, SQL-Aliasse¹ um die vorgegebene Bezeichnung der Spalten zu generieren. Wenn Sie die Spalten in der falschen Reihenfolge ausgeben, werden Ihre Ergebnisse von unserem Bewertungs-Skript als falsch gewertet. Achten Sie zudem darauf, dass die Dateien UTF-8 kodiert sind.

Verwenden Sie das in Übungsblatt 1 aufgesetzte DBMS und einen SQL-Client Ihrer Wahl und stellen Sie sicher, dass Ihre abgegebenen SQL-Dateien auf PostgreSQL 13.4 ausgeführt werden können. Die Aufgaben sollten auf der Pagila Datenbank² ausgeführt werden. Diese Datenbank müssen Sie erst einrichten (ähnliche Vorgehensweise wie bereits in Übungsblatt 1 geübt). Deshalb müssen Sie als erstes das ZIP-File der Datenbank (Link in Fußnote) herunterladen und entpacken. Erstellen Sie anschließend über Ihren SQL-Client eine neue Datenbank, importieren Sie das Schema pagila-schema. Sql und die Daten pagila-insert-data. Sql .

Hinweis A

Importieren Sie die Daten mit einer **sauberen** Lösung, zum Beispiel mit psql^a. Läuft das DBMS in einem Docker Container, so können die Befehle an das DBMS im laufenden Container mit docker exec^b ausgeführt werden, so wie es im Übungsblatt 1 gezeigt wurde. Natürlich können Sie auch die Import-Funktionen Ihres SQL Clients verwenden. **Nicht erwünscht ist das banale Kopieren und Einfügen des Dateiinhaltes**.

```
ahttps://www.postgresql.org/docs/13/app-psql.html
```

a) 0.5 Punkte Geben Sie den Titel und die Länge aller Filme aus, in denen eine Schauspielerin mit dem Vornamen AUDREY mitgespielt hat.

Reihenfolge und Bezeichnung der Ergebnisspalten: title, length





¹https://www.w3schools.com/sql/sql_alias.asp

bhttps://docs.docker.com/engine/reference/commandline/exec/

²https://github.com/devrimgunduz/pagila/archive/2.0.1.zip

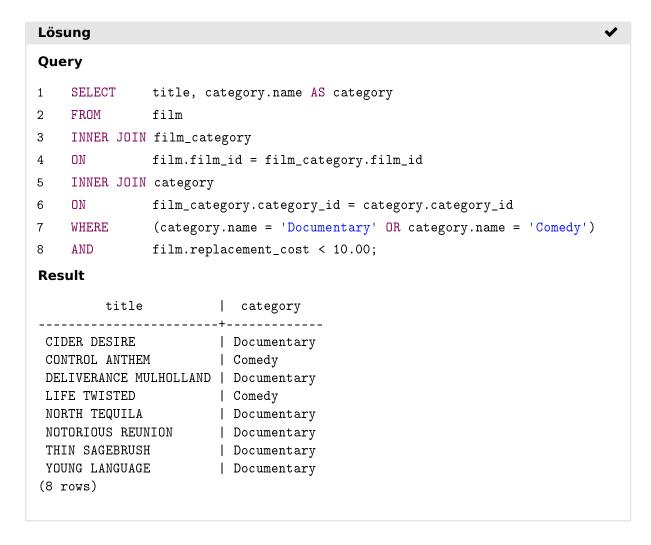
SIDE ARK	52
NEWTON LABYRINTH	75
HEAVENLY GUN	49
CONEHEADS SMOOCHY	112
MAGNOLIA FORRESTER	171
GRAFFITI LOVE	117
HUMAN GRAFFITI	68
BANGER PINOCCHIO	113
STING PERSONAL	93
PEAK FOREVER	80
SKY MIRACLE	132
PURPLE MOVIE	88
FEVER EMPIRE	158
DISTURBING SCARFACE	94
SENSE GREEK	54
VOLUME HOUSE	132
STRANGER STRANGERS	139
SQUAD FISH	136
REDEMPTION COMFORTS	179
DORADO NOTTING	139
ARK RIDGEMONT	68
HOME PITY	185
CONTROL ANTHEM	185
PITTSBURGH HUNCHBACK	134
QUILLS BULL	112
CASSIDY WYOMING	61
BED HIGHBALL	106
PILOT HOOSIERS	50
LOATHING LEGALLY	
KNOCK WARLOCK	140 71
KANE EXORCIST	
PRESIDENT BANG	92
•	144
GUNFIGHTER MUSSOLINI	127
SLEEPY JAPANESE	137
TADPOLE PARK	155
BOULEVARD MOB	63
ELF MURDER	155
MUMMY CREATURES	160
MASKED BUBBLE	151
DRIFTER COMMANDMENTS	61
BOONDOCK BALLROOM	76
SHIP WONDERLAND	104
WHALE BIKINI	109
ATLANTIS CAUSE	170
WARLOCK WEREWOLF	83
FRENCH HOLIDAY	99
CONFESSIONS MAGUIRE	65
ITALIAN AFRICAN	174
	128
USUAL UNTOUCHABLES	120

```
HALLOWEEN NUTS | 47
CAPER MOTIONS | 176
(52 rows)
```

b) 0.5 Punkte Geben Sie den Titel und die Kategorie all jener Filme aus, die in der Kategorie Documentary oder Comedy sind und weniger als 10.00 kosten, wenn man den Film ersetzen muss.

Reihenfolge und Bezeichnung der Ergebnisspalten: title, category





c) 0.5 Punkte Geben Sie die den Namen (aus Vor- und Nachnamen zusammengesetzt) aller Kunden aus, die in einem Land leben, dessen Name mit land endet.

Reihenfolge und Bezeichnung der Ergebnisspalten: name





d) 0.5 Punkte Geben Sie den Nachnamen aller Kunden an, die einen Film am 24.05.2005 bei dem Mitarbeiter, dessen Nachname Stephens lautet, ausgeliehen haben. Sie können dafür die Date/Time Functions and Operations von Postgres verwenden. Geben Sie weiters noch das Rückgabedatum aus.

 $^{^3}$ https://www.postgresql.org/docs/13/functions-datetime.html

Reihenfolge und Bezeichnung der Ergebnisspalten: last_name, return_date

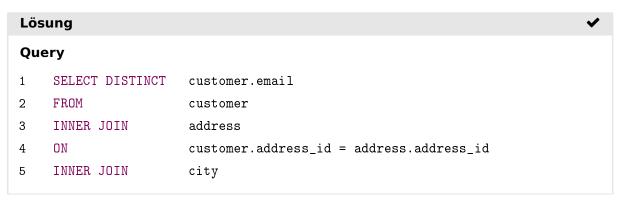




e) 1 Punkt Geben Sie die E-Mail Adresse aller Kunden aus, die im selben Land leben, wie der Mitarbeiter bei dem sie einen Film ausgeliehen haben. Achten Sie darauf, dass jede E-Mail Adresse im Ergebnis nur einmal vorkommt.

Reihenfolge und Bezeichnung der Ergebnisspalten: email





```
6
     ON
                        address.city_id = city.city_id
7
     INNER JOIN
                        rental
8
     ON
                        customer.customer_id = rental.customer_id
     INNER JOIN
9
                        staff
10
                        rental.staff_id = staff.staff_id
     0N
     INNER JOIN
11
                        address as staff_address
12
     ON
                        staff.address_id = staff_address.address_id
     INNER JOIN
13
                        city as staff_city
14
                        staff_address.city_id = staff_city.city_id
     0 N
15
     WHERE
                        city.country_id = staff_city.country_id;
 Result
                 email
  {\tt TROY.QUIGLEY@sakilacustomer.org}
  DARRELL.POWER@sakilacustomer.org
 DERRICK.BOURQUE@sakilacustomer.org
 LORETTA.CARPENTER@sakilacustomer.org
  CURTIS.IRBY@sakilacustomer.org
 (5 rows)
```

f) 1 Punkt Finden Sie heraus, welcher Mitarbeiter am meisten Geld durch einen Kunden erwirtschaftet hat. Geben Sie dazu sowohl den Namen (wieder aus Vor- und Nachnamen zusammengesetzt) des Kunden als auch des Mitarbeiters an und die insgesamt bezahlte Summe.

Reihenfolge und Bezeichnung der Ergebnisspalten: customer_name, staff_name, total_amount

```
Abgabe

© exercise2/f.sql

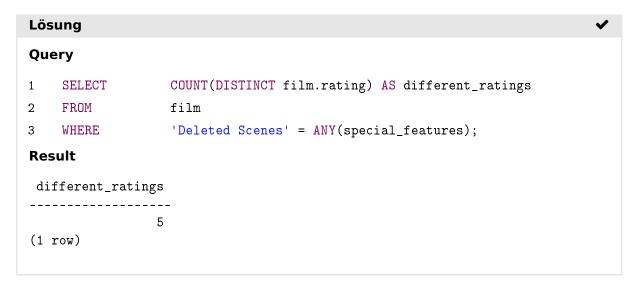
exercise2/f_result.txt
```

```
Lösung
Query
    SELECT
               customer.first_name || ' ' || customer.last_name
1
2
               AS customer_name,
               staff.first_name || ' ' || staff.last_name
3
4
               AS staff_name,
5
               SUM(amount) AS total_amount
6
    FROM
               payment
7
    JOIN
               customer USING(customer_id)
8
    JOIN
               staff USING(staff_id)
```

g) 1 Punkt Geben Sie die Anzahl verschiedener Ratings aus, die für Filme vergeben wurden, die Deleted Scenes als Bonusmaterial (special_features) haben.

Reihenfolge und Bezeichnung der Ergebnisspalten: different_ratings





h) 1 Punkt Geben Sie die Anzahl an ausgeliehenen Filmen an, die an einem Freitag den 13. zurückgegeben wurden. Benutzen Sie dafür die Date/Time Functions and Operations⁴ von Postgres.

Reihenfolge und Bezeichnung der Ergebnisspalten: returned_friday_13

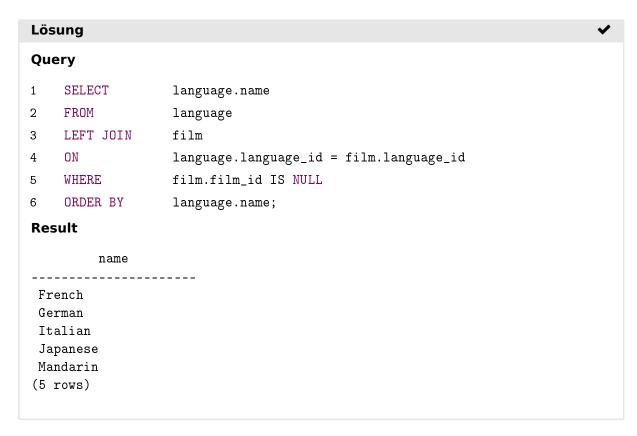


 $^{^{\}bf 4} {\tt https://www.postgresql.org/docs/13/functions-datetime.html}$

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

Reihenfolge und Bezeichnung der Ergebnisspalten: name





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.