

21.11.2023

Übungsblatt 6 – 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 – d. h. Tables – dargestellt und Tupel als die Elemente der Relationen sind dann die Zeilen – d. h. Rows – der Tabellen.

- b) ☐ ★★ Übersetzen Sie folgende SQL-Abfrage in einen zu ihr äquivalenten Relationenalgebra-Ausdruck.

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

Lösung



Naive, „Wortwörtliche“ Übersetzung:

```
1  sigma (c.country = 'Grenada') and (Player.clubId = c.id)
2  (
3  (Player cross join (rho c Club))
4  join Event.playerId = Player.id Event
5  )
```

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:

```
1  sigma (c.country = 'Grenada') (rho c Club)
2  join (Player.clubId = c.id) Player
```

```
3 join (Player.id = Event.playerId) Event
```

- c) ☐ ★ Gibt es Fälle, in denen ein Relationenalgebra-Ausdruck ein anderes Ergebnis liefert die zu ihr äquivalente SQL-Abfrage? Wie kann die SQL-Abfrage angepasst werden, damit die Ergebnisse übereinstimmen?

Lösung



Hinweis



In der relationalen Algebra ist das Ergebnis einer Abfrage eine Relation, also eine Menge im mathematischen Sinn.

In SQL können Ergebnisse Duplikate enthalten, die bei der relationalen Algebra nicht auftreten. 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 $\pi_{\text{name}} \text{Person}$ liefert

name
Anna
Oskar

Die zu ihr äquivalente SQL-Abfrage

```
1 SELECT name
2 FROM Person
```

liefert allerdings

name
Anna
Oskar
Anna

In diesem Falle müsste die SQL-Abfrage mit `DISTINCT` ergänzt werden, um ein äquivalentes Ergebnis zu erzielen:

```
1 SELECT DISTINCT name
2 FROM Person
```

- d) ☐ ★ Gegeben sei ein Relationenmodell mit Relationenschemata `Customer(CustomerId, FirstName, LastName, Address, Email)` und `Invoice(InvoiceId, CustomerId, InvoiceDate,`

Total). Übersetzen Sie folgenden Relationenalgebra-Ausdruck in einen zu ihm äquivalente SQL-Abfrage.

$$\pi_{\text{InvoiceId, InvoiceDate, Total, LastName}} \\ (\sigma_{\text{InvoiceDate} > '2012-01-01' \wedge \text{InvoiceDate} < '2012-12-31'} (\text{Invoice})) \\ \bowtie_{\text{Invoice.CustomerId} = \text{Customer.CustomerId}} (\text{Customer})$$

Lösung



```
1  SELECT      InvoiceId,
2              InvoiceDate,
3              Total,
4              LastName
5  FROM        Invoice
6  INNER JOIN  Customer USING (CustomerId)
7  WHERE      InvoiceDate BETWEEN '2012-01-02' AND '2012-12-30'
```

- e) ☐ ★ Ist es in SQL möglich Tabellen zu erstellen, in denen mehrere Spalten die identische Bezeichnung haben?

Lösung



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

- f) ☐ ★ Ist 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
```

- g) ☐ ★ Führen Sie die folgende SQL-Abfrage „händisch“ aus und vervollständigen Sie die Tabelle result.

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

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

result	
id	age

Lösung



Die Ergebnismenge enthält die Tupel $\langle 2, 10 \rangle$ und $\langle 3, 0 \rangle$, nicht aber $\langle 4, \text{NULL} \rangle$. Sowohl $\text{age} \neq \text{NULL}$ als auch $\text{age} = \text{NULL}$ haben den Wahrheitswert UNKNOWN, unabhängig vom tatsächlichen Wert von age. Deshalb können Tests auf NULL *nur* durch `is NULL` bzw. `is not NULL` ausgedrückt werden.

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

id	age
2	10
3	0

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 15.4 ausgeführt werden können.

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

Abgabe



exercise1/a.sql

Lösung



```
1 CREATE DATABASE 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.

Abgabe



 exercise1/b.sql

Lösung




```
1  CREATE TABLE project (  
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),  
11     main_location VARCHAR(255)  
12  );  
13  
14  CREATE TABLE working (  
15      employee_id INT REFERENCES employee(employee_id),  
16      project_id INT REFERENCES project(project_id),  
17      start_date TIMESTAMP,  
18      CONSTRAINT pk_working PRIMARY KEY (employee_id, project_id)  
19  );
```

- c) 0.5 Punkte Fügen Sie in die Relationen Employee und Project die Mitarbeiterin Erika Mustermann und das Projekt project2 ein. Fügen Sie weiters mindestens zwei weitere Mitarbeiter und zwei weitere Projekte mit sinnvollen Testdaten ein.

Abgabe



 exercise1/c.sql

Lösung



```
1  INSERT INTO employee (firstname, lastname, main_location)
```

```

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


```

- 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 `project2` 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.

Abgabe

 exercise1/d.sql

Lösung

```

1  INSERT INTO working (employee_id, project_id, start_date)
2  VALUES (
3      (SELECT employee_id FROM employee
4       WHERE firstname = 'Erika' AND lastname = 'Mustermann'),
5      (SELECT project_id FROM project WHERE name = 'project2'),
6      DATE('2021-11-11')
7  );

```

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 Reihenfolge und Bezeichnung der Ergebnisspalten.** Verwenden Sie, wenn notwendig, SQL-Alias¹

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

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 15.4 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.OLATsql` und die Daten `pagila-insert-data.OLATsql`.

Hinweis



Importieren Sie die Daten mit einer **sauberen** Lösung, zum Beispiel mit `psqla`. Läuft das DBMS in einem Docker Container, so können die Befehle an das DBMS im laufenden Container mit `docker execb` 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.**

^a<https://www.postgresql.org/docs/13/app-psql.html>


^b<https://docs.docker.com/engine/reference/commandline/exec/>


- 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`

Abgabe



 `exercise2/a.sql`

 `exercise2/a_result.txt`

Lösung



Query

```
1  SELECT DISTINCT  title, length
2  FROM             actor
3  INNER JOIN       film_actor
4  ON               actor.actor_id = film_actor.actor_id
5  INNER JOIN       film
6  ON               film_actor.film_id = film.film_id
7  WHERE            actor.first_name = 'AUDREY';
```

Result

title	length
SIDE ARK	52
NEWTON LABYRINTH	75

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

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		140
KNOCK WARLOCK		71
KANE EXORCIST		92
PRESIDENT BANG		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
USUAL UNTOUCHABLES		128
POTTER CONNECTICUT		115
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

Abgabe



 exercise2/b.sql

 exercise2/b_result.txt

Lösung



Query

```
1  SELECT      title, category.name AS category
2  FROM        film
3  INNER JOIN  film_category
4  ON          film.film_id = film_category.film_id
5  INNER JOIN  category
6  ON          film_category.category_id = category.category_id
7  WHERE       (category.name = 'Documentary' OR category.name = 'Comedy')
8  AND         film.replacement_cost < 10.00;
```

Result

title	category
CIDER DESIRE	Documentary
CONTROL ANTHEM	Comedy
DELIVERANCE MULHOLLAND	Documentary
LIFE TWISTED	Comedy
NORTH TEQUILA	Documentary
NOTORIOUS REUNION	Documentary
THIN SAGEBRUSH	Documentary
YOUNG LANGUAGE	Documentary


(8 rows)


- 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

Abgabe



 exercise2/c.sql

 exercise2/c_result.txt

Lösung



Query

```
1  SELECT      concat(first_name, ' ', last_name) AS name
2  FROM        customer
3  INNER JOIN   address
4  ON          customer.address_id = address.address_id
5  INNER JOIN   city
6  ON          address.city_id = city.city_id
7  INNER JOIN   country
8  ON          city.country_id = country.country_id
9  WHERE       country.country LIKE '%land';
```

Result

```
      name
-----
KATHERINE RIVERA
GAIL KNIGHT
BRIAN WYMAN
SIDNEY BURLESON
SUSAN WILSON
MARJORIE TUCKER
LEAH CURTIS
WADE DELVALLE
RUBEN GEARY
SHAWN HEATON
GERTRUDE CASTILLO
ERIKA PENA
CAROLYN PEREZ
JOHNNIE CHISHOLM
JACQUELINE LONG
RUSSELL BRINSON
JIMMIE EGGLESTON
(17 rows)
```


- 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 Operations³ 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

Abgabe



 exercise2/d.sql

 exercise2/d_result.txt

Lösung



Query

```
1  SELECT      customer.last_name, rental.return_date
2  FROM        customer
3  INNER JOIN   rental
4  ON          customer.customer_id = rental.customer_id
5  INNER JOIN   staff
6  ON          rental.staff_id = staff.staff_id
7  WHERE       staff.last_name = 'Stephens'
8  AND         DATE(rental.rental_date) = '2005-05-24';
```

Result

last_name	return_date
PURDY	2005-06-03 01:43:41
WALTERS	2005-05-29 20:34:53
ROMERO	2005-05-27 23:33:46


(3 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.

Reihenfolge und Bezeichnung der Ergebnisspalten: email

Abgabe



 exercise2/e.sql

 exercise2/e_result.txt

Lösung



Query

```
1  SELECT DISTINCT customer.email
2  FROM customer
3  INNER JOIN address
4  ON customer.address_id = address.address_id
5  INNER JOIN city
```

```

6      ON          address.city_id = city.city_id
7      INNER JOIN   rental
8      ON          customer.customer_id = rental.customer_id
9      INNER JOIN   staff
10     ON          rental.staff_id = staff.staff_id
11     INNER JOIN   address AS staff_address
12     ON          staff.address_id = staff_address.address_id
13     INNER JOIN   city AS staff_city
14     ON          staff_address.city_id = staff_city.city_id
15     WHERE        city.country_id = staff_city.country_id;

```

Result

```

-----
email
-----
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

```

1  SELECT      customer.first_name || ' ' || customer.last_name
2              AS customer_name,
3              staff.first_name || ' ' || staff.last_name
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)

```

```

9    GROUP BY    customer_id, staff_id
10   ORDER BY    total_amount DESC
11   FETCH       FIRST 1 ROWS ONLY;

```

Result

customer_name	staff_name	total_amount
JUNE CARROLL	Mike Hillyer	126.74

(1 row)

- 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

Abgabe



exercise2/g.sql
 exercise2/g_result.txt

Lösung



Query

```

1    SELECT      COUNT(DISTINCT film.rating) AS different_ratings
2    FROM        film
3    WHERE       'Deleted Scenes' = ANY(special_features);

```

Result

different_ratings
5

(1 row)

- 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

Abgabe



exercise2/h.sql
 exercise2/h_result.txt

⁴<https://www.postgresql.org/docs/13/functions-datetime.html>

Lösung**Query**

```
1  SELECT      COUNT(*) AS returned_friday_13
2  FROM        rental
3  WHERE       EXTRACT(ISODOW FROM rental.return_date) = 5
4  AND         EXTRACT(DAY FROM rental.return_date) = 13;
```



Result

```
returned_friday_13
-----
0
(1 row)
```

- 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

Abgabe

 exercise2/i.sql
 exercise2/i_result.txt

Lösung**Query**

```
1  SELECT      language.name
2  FROM        language
3  LEFT JOIN   film
4  ON          language.language_id = film.language_id
5  WHERE       film.film_id IS NULL
6  ORDER BY    language.name;
```

Result

```
name
-----
French
German
Italian
Japanese
Mandarin
(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 16:00 ab.