

21.11.2023

Übungsblatt 6

Diskussionsteil (im PS zu lösen; keine Abgabe nötig)

- a) ☐ ★ Wie werden Relationen und Tupel aus der relationalen Algebra in einem relationalen Datenbanksystem dargestellt?
- b) ☐ ★★ Übersetzen Sie folgende SQL-Abfrage in einen zu ihr äquivalenten Relationenalgebra-Ausdruck. $\sigma_{c.country='Grenada' \text{ and } Player.clubId=c.id}((player \text{ cross join } (\rho c Club)) \text{ join } Event.playerId=Player.id(Event))$
- | | | |
|---|------------|----------------------------|
| 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 |
- 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? **schlüsselwort distinct entfernt duplicat schlüssel;**
- 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.

```
select InvoiceId, InvoiceDate, Total, LastName,  $\pi_{InvoiceId, InvoiceDate, Total, LastName}$ 
from Invoice
join Customer on
CustomerId=Customer.CustomerId
where InvoiceDate ... and InvoiceDate ...
( $\sigma_{InvoiceDate > '2012-01-01' \wedge InvoiceDate < '2012-12-31'}$  (Invoice))
 $\bowtie_{Invoice.CustomerId=Customer.CustomerId}$  (Customer))
```

- e) ☐ ★ Ist es in SQL möglich Tabellen zu erstellen, in denen mehrere Spalten die identische Bezeichnung haben? **fehlermeldung**
- f) ☐ ★ Ist es in SQL möglich eine Abfrage zu schreiben, in deren Ergebnis mehrere Spalten die identische Bezeichnung haben? **ja ist möglich, zb 2 Spalten die ID heißen !**
- 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

NULL hat wert unknown
 sowohl NULL = NULL als auch
 NULL <> NULL gibt unknown als
 output

result	
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

- 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

- c) 0.5 Punkte Fügen Sie in die Relationen Employee und Project die Mitarbeiterin Erika Mustermann und das Projekt projekt2 ein. Fügen Sie weiters mindestens zwei weitere Mitarbeiter und zwei weitere Projekte mit sinnvollen Testdaten ein.

Abgabe



 exercise1/c.sql

- 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.

Abgabe



 exercise1/d.sql

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 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-schemaOLAT.sql` und die Daten `pagila-insert-dataOLAT.sql`.

Hinweis



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.**

^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`

- 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`

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

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



- 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	↑
 <code>exercise2/c.sql</code>	
 <code>exercise2/c_result.txt</code>	



- 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.

Reihenfolge und Bezeichnung der Ergebnisspalten: `last_name`, `return_date`

Abgabe	↑
 <code>exercise2/d.sql</code>	
 <code>exercise2/d_result.txt</code>	



- 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	↑
 <code>exercise2/e.sql</code>	
 <code>exercise2/e_result.txt</code>	

- 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	↑
 <code>exercise2/f.sql</code>	
 <code>exercise2/f_result.txt</code>	

- 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`

³<https://www.postgresql.org/docs/13/functions-datetime.html>

Abgabe exercise2/g.sql exercise2/g_result.txt



- 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

- 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

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.

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