Institut für Informatik - Lehrstuhl Datenbanken

Einführung in Datenbanken

Wintersemester 22/23 - Prof. Dr. Stefan Brass, PD Dr. Alexander Hinneburg

Übung 8: SQL-Anfragen Nicht-Monotone Anfragen, Aggregationsfunktionen
Abgabe bis Mo. 19.12.2022, 18:00 Uhr,
Übungsplattform: Studip → Einführung in Datenbanken → Übungsplattform

Aufgabe 8.1: 10 Punkte

Das folgende relationale Schema beschreibt eine Film-Verleih-Kette. Das Datenbank-Schema wurde 2006 von MySQL zu Demonstrationszwecken entwickelt. Eine Beschreibung finden Sie unter `http://dev.mysql.com/doc/sakila/en/index.html`. Loggen Sie sich in die PostgreSQL-Datenbank über StudIP (Einführung in Datenbanken → Informationen → Postgres-DB) ein und wählen Sie links das Schema `sakila_public`. Mit Link SQL-Command (links) kommen Sie zur Eingabe von SQL-Statements.

Actor (actor_id, first_name, last_name, last_update)

Address (address_id, address, address2°, district, city_id→City, postal_code°, phone, last_update)

Category (category_id, name, last_update)

City (city_id, city, country_id→Country, last_update)

Country (country_id, country, last_update)

Customer (customer_id, store_id→Store, first_name, last_name, email°, address_id→Address, active°, create_date, last_update°)

Film (film_id, title, description°, release_year°, language_id→Language, original_language_id° →language, rental_duration, rental_rate, length°, replacement_cost, rating°, special_features°, last_update)

Film_Actor (actor_id→Actor, film_id→Film, last_update)

Film_Category (film_id→Film, category_id→Category, last_update)

Film_Text (film_id, title, description)

Inventory (inventory_id, film_id→Film, store_id→Store, last_update)

Language (language_id, name, last_update)

Payment (payment_id, customer_id→Customer, staff_id→Staff, rental_id→Rental, amount, payment_date, last_update)

Rental (rental_id, rental_date, inventory_id→Inventory, customer_id→Customer, return_date°, staff_id→Staff, last_update)

Staff (staff_id, first_name, last_name, address_id→Address, picture°, email°, store_id→Store, active, username, password°, last_update)

Store (store_id, manager_staff_id→Staff, address_id→Address, last_update)

Geben Sie für die folgenden Anfragen entsprechende SQL-Statements an. Für jede Anfrage soll nur ein SQL-Statement angegeben werden. Falls die Tutoren eine SQL-Anfrage nicht schnell verstehen können, kann es Punktabzug geben. Sie können das durch gute Formatierung und hilfreiche Kommentare (wo nötig) vermeiden. Beachten Sie aber, dass die Tutoren SQL natürlich sehr gut kennen. Vermeiden Sie triviale Kommentare. SQL-Statements mit Syntax-Fehlern werden mit Null Punkten bewertet. Geben Sie jede Teilaufgabe als separate Text-Datei mit der Endung .sql ab (kein Word). Unter Windows können Sie Text-Dateien mit Notepad editieren. Das Programm Notepad++ (`http://notepad-plus-plus.org`) bietet mehr Funktionen.

Beachten Sie, dass die SQL Anfragen keine Duplikate ausgeben sollen, d.h. verwenden Sie DISTINCT, wenn es notwendig ist. Für vergessenes oder überflüssiges DISTINCT werden Punkte abgezogen.

a Welche Filme haben mit einem weiteren Film mindestens drei verschiedene Schauspieler gemeinsam und die Vermietungskosten (`rental_rate`) für den auszugebenden Film sind mindestens doppelt so hoch wie die des weiteren Films? Geben Sie `film_id` und `title` dieses Films aus.

Benutzen Sie nicht `Group By` und keine Aggregationsfunktionen, wie z.B. `count`.

b) Welche Filme (film_id, title) sind ausschließlich in den Kategorien 'Children' oder 'Comedy' und in keinen anderen Kategorien? Geben Sie `film_id` und `title` dieser Filme aus.

Treffen Sie keine vereinfachenden Annahmen, wie z.B. dass Filme nur einer Kategorie zugeordnet sind. Das Datenbankschema erlaubt es, dass Filme mehreren Kategorien zugeordnet sein können.

c) Für welche Filme (film_id, title, replacement_cost) in der Kategorie 'Children' muss am meisten bezahlt werden, wenn sie kaputt gehen? Geben Sie `film_id`, `title` und `replacement_cost` dieser Filme aus.

Benutzen Sie nicht `Group By` und keine Aggregationsfunktionen, wie z.B. `max`. Benutzen Sie auch kein `Order By` in Kombination mit `limit`.

d) Geben Sie alle Filme aus, in denen JOE SWANK und EWAN GOODING nicht gleichzeitig mitspielen, sondern immer nur einer von beiden. Geben Sie ebenfalls aus, wer von den beiden Schauspielern mitspielt. Geben Sie `film_id` und `title` dieser Filme und `first_name` und `last_name` des Schauspielers oder Schauspielerin aus, die mitspielt.

Ordnen Sie die Filme nach alphabetisch aufsteigendem Titel.

e) Geben Sie aus wieviele Kinderfilme (Kategorie 'Children') die Videothek im Angebot hat. Nennen Sie die auszugebende Spalte `no_child_films`.