Drugi sprawdzian z SQL 25.03.2020

Oczekujemy rozwiązania w postaci pliku zawierającego TREŚCI poleceń SQL, a nie znalezionej odpowiedzi. Nie będą sprawdzane jakiekolwiek zapytania niepoprawne składniowo, sprawdź swoje rozwiązanie używając `\i plik.sql` ! Plik możesz wysyłać wielokrotnie, sprawdzana będzie wyłącznie najnowsza wersja.

Wczytaj do swojej bazy danych plik hsm.dump. Jest to dump bazy hsm.stackexchange.org poświęconej dyskusjom na tematy związane ze historią matematyki i nauki.

Zachęcam do korzystania z dokumentacji PostgreSQL.

Format nazwy pliku z rozwiązaniem: `grupa-imie-nazwisko.sql`, gdzie `grupa` to inicjały prowadzącego Twoją grupę: (`pwi/plg/mpy/rfe/pga`), np. `pwi-Jan-Kowalski.sql`. Wymagany format pliku z rozwiązaniem (tu też podaj swoje imię, nazwisko i grupę):

```
-- Imię Nazwisko, grupa np. Jan Kowalski, pwi
-- Zadanie 1
<zapytanie>                -- zastąp napis `<zapytanie>` swoim zapytaniem :)

-- Zadanie 2
<zapytanie>
...
```

Zadanie 1 (2 pkt.) Przyjmijmy, że post p jest duplikatem jakiegoś posta r jeśli w tabeli `postlinks` istnieje wpis z `postid` równym `id` posta p, `relatedpostid` równym `id` posta r oraz z `linktypeid` wynoszącym 3.

Dla każdego użytkownika wypisz jego `id, displayname, reputation` oraz sumaryczną liczbę jego postów, które są duplikatami innych postów. Uwzględnij wyłącznie użytkowników, dla których powyższa liczba jest większa od zera.

Wyniki posortuj w pierwszej kolejności malejąco względem ostatniej kolumny (tj. wg sumarycznej liczby jego postów, które są duplikatami innych postów), a następnie alfabetycznie wg drugiej kolumny (tj. `displayname` użytkownika). Wypisz nie więcej niż pierwsze 20 wyników.

```
SELECT users.Id, DisplayName, Reputation, COUNT(DISTINCT PostId)
  FROM postlinks
    JOIN posts ON posts.Id = PostId
    JOIN users ON OwnerUserId = users.Id
    JOIN posts AS orig ON orig.Id = RelatedPostId
  WHERE LinkTypeId = 3
  GROUP BY users.Id, DisplayName, Reputation
  ORDER BY 4 DESC, 2
  LIMIT 20;
```

Zadanie 2 (2 pkt.) Dla każdego użytkownika posiadającego odznakę *Fanatic* (wg `badges`) wypisz jego `id, displayname, reputation` oraz sumaryczną liczbę komentarzy

do jego postów oraz średni score tych komentarzy. Zostaw tylko te wyniki, dla których sumaryczna liczba komentarzy nie przekracza 100.

Wyniki posortuj w pierwszej kolejności malejąco względem przedostatniej kolumny (tzn. sumarycznej liczby komentarzy), a następnie alfabetycznie wg drugiej kolumny (tj. `displayname` użytkownika). Wypisz nie więcej niż pierwsze 20 wyników.

```
SELECT users.id, displayname, reputation, count(comments.id), avg(comments.score)
FROM users
 JOIN badges ON users.id=badges.userid
 LEFT JOIN posts ON users.id=posts.owneruserid
 LEFT JOIN comments on posts.id=comments.postid
WHERE badges.name='Fanatic'
GROUP BY users.id, displayname, reputation
HAVING count(comments.id) <= 100
ORDER BY 4 DESC, 2
LIMIT 20
```

Zadanie 3 (2 pkt.)

- Spraw aby atrybut id tabeli users był jej kluczem głównym.
- Dodaj klucz obcy, który wymusi aby w tabeli badges wszystkie niepuste wartości userid występowały jako id w tabeli users.
- Usuń kolumnę viewcount tabeli posts.
- Usuń wszystkie krotki z tabeli `posts` takie, że ich `body` jest pustym napisem lub nullem.

```
ALTER TABLE users
   ADD PRIMARY KEY (id);

ALTER TABLE badges
   ADD FOREIGN KEY (userid)
   REFERENCES users(id);

ALTER TABLE posts
   DROP COLUMN viewcount;

DELETE FROM posts
   WHERE body = '' OR body IS NULL;
```

Zadanie 4 (2 pkt.) Przepisz wszystkie komentarze z tabeli comments do tabeli posts dbając o następujące szczegóły. Nie zmieniaj zawartości tabeli comments ani krotek z obecnego stanu tabeli posts. Dla każdego przepisywanego komentarza

- zadbaj aby jego id było unikalne (a w szczególności różne od id wszystkich dotychczasowych postów) - w tym celu wykorzystaj odpowiednio zdefiniowaną sekwencję, powiąż tę sekwencję z kolumną `posts.id`,

- ustaw posttypeid na 3, a parentid na obecny postid,
- przepisz userid na owneruserid,
- przepisz text na body,
- przepisz bez zmian score oraz creationdate.

Pozostałym atrybutom ustaw wartość NULL.

```
CREATE SEQUENCE aux;

SELECT setval('aux', max(id)) FROM posts;

ALTER TABLE posts
     ALTER COLUMN id
     SET DEFAULT nextval('aux');

ALTER SEQUENCE aux OWNED BY posts.id;

INSERT INTO posts (posttypeid, parentid, owneruserid, body, score, creationdate)
SELECT 3, postid, userid, text, score, creationdate
 FROM comments;
```