Databázové systémy

Zadanie 2

Základné SQL dotazy

Filip Híreš

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

xhires@stuba.sk

9.3.2024

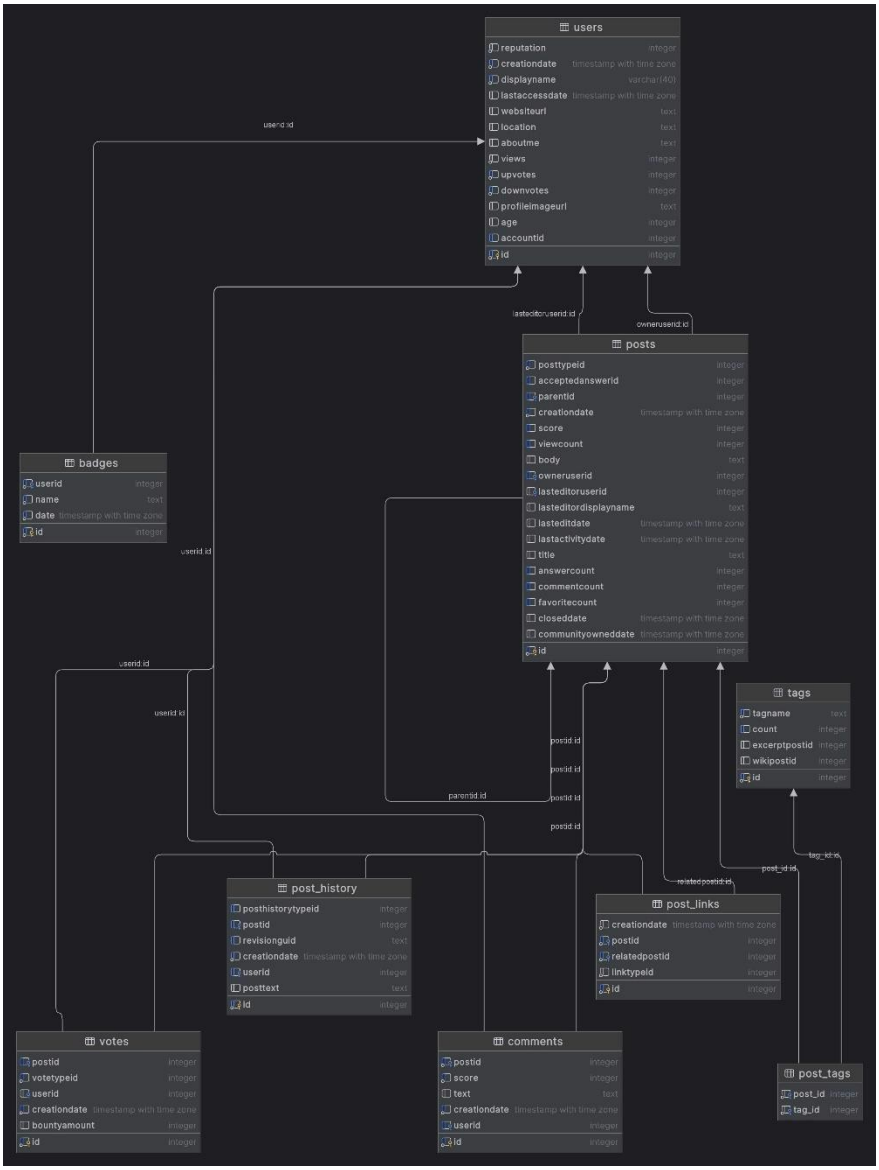
Zadanie:

Zadanie je zamerané na vytvorenie základných SQL dotazov nad priloženou PostgreSQL databázou, ktorá´ a vznikla z Stack Exchange Data Dump Superuser datasetu. Cieľom je realizovať uvedené úlohy ako RESTful endpointy, ktoré sú realizované ako SQL dotazy, transformované do JSON výstupu. Výstup je opísaný ako JSON Schéma. Vstupy na pripojenie k databázovému serveru budú poskytované pomocou environmentálnych premenných). Poradie vo výstupe musí byť zhodné s jeho definíciou pri jednotlivých end-pointoch. Pri realizácii je možné používať iba čisté SQL dopyty a nie je dovolené používať žiadne ORM. Pri odpovediach je potrebné vrátiť časy vo formáte ISO8601 v UTC. Okrem implementovanie samotných endpointov je potrebné vyhotoviť dokumentáciu, ktorá bude obsahovať:

- SQL dopyty s ich popisom
- príklady volania HTTP end-pointu (pre každý endpoint)

Dokumentácia môže byť realizovaná ako PDF alebo Markdown dokumentácia s tým, že sa bude nachádzať v AIS odovzdaní a aj v samotnom GitHub repozitári.

Schéma databázy:



Implementácia:

Pre implementáciu zadania som si zvolil programovací jazyk Python. Na implementáciu samotných endpointov som využil knižnice psycopg2 a fastapi. Na lokálne testovanie som používal pgAdmin4 a uvicorn.

Endpointy:

Štruktúra endpointov pozostáva z troch hlavných častí:

- **SQL dopyt:**
```
query = """SELECT users.* FROM users
    JOIN comments on users.id = comments.userid
WHERE comments.postid = %s
    GROUP BY users.id
    ORDER BY MAX(comments.creationdate) DESC;"""
```
Premenná query obsahuje samotný kód SQL dopytu.

- **Funkciu na formátovanie JSON výstupu:**
```
@router.get('/v2/posts/{post_id}/users')
async def get_users(post_id):
    postgres_users = get_postgres_users(post_id)
response = [
        {
         "id": row[0],
         "reputation": row[1],
         "creationdate": date_formating(row[2]),
"displayname": row[3],
         "lastaccessdate": date_formating(row[4]),
"websiteurl": row[5],
         "location": row[6],
         "aboutme": row[7],
         "views": row[8],
         "upvotes": row[9],
         "downvotes": row[10],
         "profileimageurl": row[11],
         "age": row[12],
         "accountid": row[13]
         }
        for row in postgres_users
    ]
    return {"items": response}
```
Táto funkcia zaistí, že výstup z databázy bude správne naformátovaný do JSON slovníkov.

- **Funkciu na vyhodnotenie dopytu:**
```
def get_postgres_users(post_id):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
    cursor = connection.cursor()
    cursor.execute(query, (post_id,))
    version = cursor.fetchall()
    connection.close()
    return version
```

Táto funkcia vyhodnocuje pripojenie k databáze a samotný SQL dopyt, ktorý je uložený v premennej query.

Obsah a parametre funkcií a dopytov sa pre každý endpoint líšia.

Endpoint users.py - GET /v2/posts/:post_id/users:

Endpoint vracia zoznam všetkých diskutujúcich príspevku *post_id*, ktoré je zadané na vstupe. Diskutujúci sú zoradení podľa času vytvorenia komentára od najnovšieho po najstarší.

```
SELECT users.* FROM users
JOIN comments on users.id = comments.userid
WHERE comments.postid = %s
GROUP BY users.id
ORDER BY MAX(comments.creationdate) DESC;
```

Na tabuľku users sa pripojí tabuľka comments. Vyfiltrujú sa záznamy, ktoré súvisia s konkrétnym príspevkom. Záznamy sa zoskupia podľa users.id aby sa odstránili duplikáty používateľov. Následne sa používatelia zoradia podľa ich najnovšie pridaného komentára na daný príspevok.

Výstup pre http volanie 127.0.0.1:8000/v2/posts/1819157/users


```
{
    "items": [
        {
            "id": 1866388,
            "reputation": 1,
            "creationdate": "2023-12-01T00:05:24.337+01",
"displayname": "TomR.",
            "lastaccessdate": "2023-12-03T06:18:19.607+01",
"websiteurl": null,
            "location": null,
            "aboutme": null,
            "views": 1,
            "upvotes": 0,
            "downvotes": 0,
            "profileimageurl": null,
            "age": null,
            "accountid": 30035903
        }
    ]
}
```


Endpoint friends.py - GET /v2/users/:user_id/friends:

Endpoint vracia zoznam všetkých diskutujúcich, ktorí komentovali na príspevkoch používateľa *user_id*, ktoré je zadané na vstupe, a používateľov, ktorý založili príspevky, na ktorých daný používateľ komentoval. Používatelia sú zoradení podľa dátumu registrácie od najstarších po najnovších.

```
SELECT DISTINCT users.* FROM (
    SELECT comments.userid
    FROM users
    JOIN posts ON users.id = posts.owneruserid
    JOIN comments ON posts.id = comments.postid
    WHERE posts.owneruserid = %s or comments.userid = %s )
AS first
JOIN users ON first.userid = users.id
ORDER BY creationdate ASC;
```

Najprv v subqery first prepojíme tabuľky users, posts a comments. Potom vyfiltrujeme všetky id používateľov, ako to vyžaduje zadanie. Následne vyfiltrované id prepojíme s údajmi o používateľoch a zoradíme ich.

Výstup pre http volanie 127.0.0.1:8000/v2/users/1076348/friends


```
{
    "items": [
        {
            "id": 482362,
            "reputation": 10581,
            "creationdate": "2015-08-11T17:42:36.267+02",
"displayname": "DrZoo",
            "lastaccessdate": "2023-12-03T06:41:11.75+01",
            "websiteurl": null,
            "location": null,
            "aboutme": null,
            "views": 1442,
            "upvotes": 555,
            "downvotes": 46,
            "profileimageurl": null,
            "age": null,
            "accountid": 2968677
        },
        {
            "id": 1076348,
            "reputation": 1,
            "creationdate": "2019-08-15T16:00:28.473+02",
"displayname": "Richard",
            "lastaccessdate": "2019-09-10T16:57:48.527+02",
"websiteurl": null,
            "location": null,
            "aboutme": null,
            "views": 0,
            "upvotes": 0,
            "downvotes": 0,
            "profileimageurl": null,
            "age": null,
            "accountid": 16514661
        }
    ]
}
```


Endpoint stats.py - GET /v2/tags/:tagname/stats:

Endpoint vracia percentuálne zastúpenie príspevkov s určitým *tagname*, ktoré je zadané na vstupe v rámci celkového počtu príspevkov pre každý deň v týždni zvlášť.

```
WITH temp_posts AS (SELECT
CASE WHEN EXTRACT(DOW FROM posts.creationdate::timestamp) = 0 THEN 7
ELSE EXTRACT(DOW FROM posts.creationdate::timestamp)
END AS number_of_week,
SUM(CASE WHEN tags.tagname = %s THEN 1 ELSE 0 END) AS linux_posts,
COUNT(*) AS total_posts
FROM posts
JOIN post_tags ON posts.id = post_tags.post_id
JOIN tags ON post_tags.tag_id = tags.id
GROUP BY number_of_week)
SELECT ROUND(linux_posts * 100.0 / total_posts, 2) AS percentage FROM


temp_posts
ORDER BY number_of_week;
```

Najprv prepojíme tabuľky posts, post_tags a tags. Následne z dátumu vytvoríme číslo dňa podľa ktorého záznamy zoskupíme. Keďže Nedeľa má číslo 0 zmeníme ho na 7 aby sa po zoradení nachádzala na konci. Pre každý deň spočítame počet záznamov s konkrétnym tagom a počet

všetkých záznamov. Následne z týchto dvoch záznamov vypočítame percentuálne zastúpenie.

Výstup pre http volanie http://127.0.0.1:8000/v2/users/1076348/friends


```
{
    "result": {
        "Monday": 3.97,
        "Tuesday": 4,
        "Wednesday": 3.96,
        "Thursday": 3.92,
        "Friday": 4.01,
        "Saturday": 4.04,
        "Sunday": 3.98
    }
}
```


Posledné dva endpointy sa nachádzajú v rovnakom .py súbore keďže majú rovnakú adresu ktorá sa líši zadanými parametrami

Endpoint posts.py - GET /v2/posts?duration=:duration_in_minutes&limit=:limit:

Endpoint vracia zoznam určitého počtu najnovších vyriešených príspevkov podľa parametru *limit,* ktoré boli otvorené maximálne určitý počet minút podla parametru *duration_in_minutes.* Obidva parametre sú zadané na vstupe.

```
SELECT closeddate, creationdate, ROUND(EXTRACT(EPOCH FROM (closeddate -
creationdate)) / 60, 2) AS duration, id, lastactivitydate, lasteditdate,
title, viewcount FROM posts
WHERE closeddate IS NOT null and ROUND(EXTRACT(EPOCH FROM (closeddate -
creationdate)) / 60, 2) < %s
ORDER BY creationdate DESC
LIMIT %s;
```

Najprv pre každý záznam ktorého hodnota closeddate nie je null (a teda sa jedná o vyriešený príspevok), vypočítame rozdiel medzi dátumami otvorenia a zatvorenia. Následne vyfiltrujeme záznamy, ktorých výsledná hodnota je väčšia ako hodnota na vstupe. Nakoniec ich zoradíme a pomocou funkcie limit vyberieme určitý počet záznamov.

Výstup pre http volanie 127.0.0.1:8000/v2/posts/?duration=5&limit=2


```
{
    "items": [
        {
            "closeddate": "2023-11-30T16:59:23.56+01",
            "creationdate": "2023-11-30T16:55:32.137+01",
            "duration": 3.86,
            "id": 1818849,
            "lastactivitydate": "2023-11-30T16:55:32.137+01",
            "lasteditdate": null,
            "title": "Why is my home router address is 10.x.x.x and not
100.x.x.x which is properly reserved and widely accepted for CGNAT?",
"viewcount": 22924
        },
        {
            "closeddate": "2023-11-27T18:29:18.947+01",
"creationdate": "2023-11-27T18:26:57.617+01",
"duration": 2.36,
            "id": 1818386,

            "lastactivitydate": "2023-11-27T18:26:57.617+01",
            "lasteditdate": null,
            "title": "Are there any libraries for parsing DWG files with
LGPL, MIT, Apache, BSD?",
```

```
            "viewcount": 19
        }
    ]
}
```


Endpoint posts.py - GET /v2/posts?limit=:limit&query=:query:

Endpoint vracia zoznam určitého počtu najnovších príspevkov podľa parametru *limit,* ktoré obsahujú v *posts.title* alebo *posts.body* reťazec z parametru *query.* Obidva parametre sú zadané na vstupe.

```
WITH temp AS (SELECT answercount, body, closeddate, creationdate, id,
lastactivitydate, lasteditdate, title, viewcount FROM posts
WHERE (posts.title ILIKE %s OR posts.body ILIKE %s)
ORDER BY creationdate DESC
LIMIT %s)
SELECT temp.answercount, temp.body, temp.closeddate, temp.creationdate,
temp.id, temp.lastactivitydate, temp.lasteditdate, array_agg(tags.tagname),
temp.title, temp.viewcount FROM temp
JOIN post_tags ON temp.id = post_tags.post_id
JOIN tags ON post_tags.tag_id = tags.id
GROUP BY temp.answercount, temp.body, temp.closeddate, temp.creationdate,
temp.id, temp.lastactivitydate, temp.lasteditdate, temp.title,
temp.viewcount;
```

Najprv záznamy zoradíme a vyberieme prvých N, v ktorých sa nachádza hľadaný reťazec, funkciou limit. Následne ich prepojíme s tabuľkami post_tags a tags a zoskupíme ich aby sme dostali pole všetkých tagov patriacich k danému príspevku.

Výstup pre http volanie 127.0.0.1:8000/v2/posts?limit=1&query=linux


```
{
    "items": [
        {
            "answercount": 0,
            "body": "<p>I have recently installed virtualbox on my windows
10 and trying to run Linux Ubuntu and Kali. Everything working on Ubuntu
without any issue but when I am running kali it is not taking
keyboard(Samsung bluetooth 500) input. Please can anyone help me out here.\
nMany thanks in advance!!</p>\n",
            "closeddate": null,
            "creationdate": "2023-12-03T05:22:43.587+01",
            "id": 1819160,
            "lastactivitydate": "2023-12-03T05:22:43.587+01",
            "lasteditdate": null,
            "tags": [
                "virtual-machine"
            ],
            "title": "Keyboard not working on khali linux",
"viewcount": 7
        }
    ]
}
```