Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Zadanie 2

Databázové systémy

Adam Jurčišin

Dátum : 10. 3. 2024

Obsah

Adam Jurčišin
ID: 116203

1. Úvod

V tomto zadaní máme za úlohu vytvoriť 5 úloh, ktoré vykonávajú SQL dopyty na PostgreSQL databázu.

Zadanie budeme vytvárať v programovacom jazyku Python a SQL.

2. 1. Úloha

V prvej úlohe sme implementovali takýto endpoint:

```python
@router.get("/v2/posts/{post_id}/users")
async def posts(post_id: int):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
        host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
```

Tento endpoint vykonáva nasledujúci SQL dopyt, ktorý pracuje s tabuľkami user, a comments kde sa pomocou JOIN tieto tabulky spoja na základe userid. Následne aplikujeme podmienku WHERE, ktorá vyberá iba tie riadky ktore majú postid rovnaké ako je zadaná hodnota v endpointe. Výsledky sú zoradené v zostupnom poradí.

```
    cursor = connection.cursor()
    cursor.execute("""
        SELECT u.id, u.reputation, u.creationdate, u.displayname, u.lastaccessdate,
               u.websiteurl, u.location, u.aboutme, u.views, u.upvotes, u.downvotes,
               u.profileimageurl, u.age, u.accountid
        FROM users AS u
        JOIN comments AS c ON c.userid = u.id
        WHERE c.postid = %s
        ORDER BY c.creationdate DESC
    """, (post_id,))
    db_data = cursor.fetchall()
    cursor.close()
    connection.close()
    items = []
    for row in db_data:
        item = {
                "id": row[0],
                "reputation": row[1],
                "creationdate": row[2].isoformat() if row[2] else None,
                "displayname": row[3],
                "lastaccessdate": row[4].isoformat() if row[4] else None,
                "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]
            }
```

Príklad volania endpointu:

```
GET /v2/posts/1819157/users
```

Výsledok z databázy:


```
{
  "items": [
    {
      "aboutme": null,
      "accountid": 30035903,
      "age": null,
      "creationdate": "2023-11-30T23:05:24.337000+00:00",
      "displayname": "TomR.",
      "downvotes": 0,
      "id": 1866388,
      "lastaccessdate": "2023-12-03T05:18:19.607000+00:00",
      "location": null,
      "profileimageurl": null,
      "reputation": 1,
      "upvotes": 0,
      "views": 1,
      "websiteurl": null
    }
  ]
}
```

3. 2. Úloha

V druhej úlohe sme implementovali takýto endpoint:

```python
@router.get("/v2/users/{user_id}/friends")
async def users(user_id: int):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
        host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
```

Tento endpoint vykonáva nasledujúci SQL dopyt, ktorý pracuje s tabuľkami users posts a comments. Prvá časť spojí tabuľky users a posts na základne owneruserid v tabuľke posts a id v tabuľke users a následne tieto tabuľky spoji s tabuľkou comments podľa postid. Potom sa aplikuje WHERE ktorá vyberá iba tie riadky, ktoré majú postid rovnaké ako je zadaná hodnota v endpointe.

```
cursor = connection.cursor()
cursor.execute("""
    SELECT u.id, u.reputation, u.creationdate, u.displayname, u.lastaccessdate,
       u.websiteurl, u.location, u.aboutme, u.views, u.upvotes, u.downvotes,
       u.profileimageurl, u.age, u.accountid
    FROM users AS u
    JOIN posts AS p ON p.owneruserid = %s
    JOIN comments AS c ON c.postid = p.id AND u.id = c.userid
    UNION
    SELECT u.id, u.reputation, u.creationdate, u.displayname, u.lastaccessdate,
       u.websiteurl, u.location, u.aboutme, u.views, u.upvotes, u.downvotes,
       u.profileimageurl, u.age, u.accountid
    FROM users AS u
    JOIN comments AS c ON c.userid = u.id
    WHERE c.postid IN (
        SELECT postid
        FROM comments
        WHERE userid = %s
    )
    ORDER BY creationdate ASC;
""", (user_id, user_id,))
```

Príklad volania endpointu:

```
GET /v2/users/1076348/friends
```

Výsledok z databázy:


```
{
  "items": [
    {
      "aboutme": null,
      "accountid": 2968677,
      "age": null,
      "creationdate": "2015-08-11T15:42:36.267000+00:00",
      "displayname": "DrZoo",
      "downvotes": 46,
      "id": 482362,
      "lastaccessdate": "2023-12-03T05:41:11.750000+00:00",
      "location": null,
      "profileimageurl": null,
      "reputation": 10581,
      "upvotes": 555,
      "views": 1442,
      "websiteurl": null
    },
```


4. 3. Úloha

Tretia úloha má aplikovaný nasledujúci endpoint:

```python
@router.get("/v2/tags/{tag_name}/stats")
async def tags(tag_name: str):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
        host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
```

Tento SQL dopyt vytvára JSON objekt obsahujúci percentuálny podiel príspevkov s daným štítkom pre každý deň v týždni. Vnútri dotazu sú poddotazy, ktoré počítajú počet príspevkov s daným tagom pre každý deň v týždni. Výsledok je zoskupený do jedného JSON objektu s kľúčmi pre každý deň v týždni a príslušnými percentuálnymi hodnotami.

```
    cursor.execute("""
    SELECT
        json_build_object(
            'monday', COALESCE(monday_percentage, 0),
            'tuesday', COALESCE(tuesday_percentage, 0),
            'wednesday', COALESCE(wednesday_percentage, 0),
            'thursday', COALESCE(thursday_percentage, 0),
            'friday', COALESCE(friday_percentage, 0),
            'saturday', COALESCE(saturday_percentage, 0),
            'sunday', COALESCE(sunday_percentage, 0)
        ) AS result
    FROM (
        SELECT
            ROUND(monday_tag_count * 100.0 / NULLIF(monday_count, 0), 2) AS monday_percentage,
            ROUND(tuesday_tag_count * 100.0 / NULLIF(tuesday_count, 0), 2) AS tuesday_percentage,
            ROUND(wednesday_tag_count * 100.0 / NULLIF(wednesday_count, 0), 2) AS wednesday_percentage,
            ROUND(thursday_tag_count * 100.0 / NULLIF(thursday_count, 0), 2) AS thursday_percentage,
            ROUND(friday_tag_count * 100.0 / NULLIF(friday_count, 0), 2) AS friday_percentage,
            ROUND(saturday_tag_count * 100.0 / NULLIF(saturday_count, 0), 2) AS saturday_percentage,
            ROUND(sunday_tag_count * 100.0 / NULLIF(sunday_count, 0), 2) AS sunday_percentage
        FROM (
            SELECT
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 1) AS monday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 1 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 2) AS tuesday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 2 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 3) AS wednesday_count
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 3 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 4) AS thursday_count
        FROM (
            SELECT
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 1) AS monday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 1 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 2) AS tuesday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 2 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 3) AS wednesday_coun
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 3 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 4) AS thursday_count
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 4 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 5) AS friday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 5 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 6) AS saturday_count
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 6 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 0) AS sunday_count,
                (SELECT COUNT(*) FROM posts AS p WHERE EXTRACT(DOW FROM p.creationdate) = 0 AND p.id IN
                    (SELECT pt.post_id FROM tags AS t JOIN post_tags AS pt ON t.id = pt.tag_id WHERE t.tagname
        ) AS days_counts
    ) AS sql_result;
    """, (tag_name, tag_name, tag_name, tag_name, tag_name, tag_name, tag_name,))
```

Príklad volania endpointu:

```
GET /v2/tags/linux/stats
```

Výsledok z databázy:


```
{
  "result": {
    "friday": 4.7,
    "monday": 4.73,
    "saturday": 4.97,
    "sunday": 4.85,
    "thursday": 4.58,
    "tuesday": 4.68,
    "wednesday": 4.62
  }
}
```


5. 4. Úloha

Štvrtá úloha má aplikovaný nasledujúci endpoint:

```python
@router.get("/v2/posts/")
async def get_posts(limit: int = Query(None), query: str = Query(None), duration: int = Query(None)):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
        host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
```

Tento SQL dopyt pracuje s tabuľkami posts, post_tags a tags. Vyberá atribúty id, creationdate, viewcount, lasteditdate, title, body, answercount, closeddate a lastactivitydate z tabuľky posts. Súčasťou dotazu je aj poddotaz, ktorý spojí tagy priradené k príspevku do jedného reťazca. V podmienke WHERE sa filtrovajú riadky podľa podobnosti s názvom alebo obsahom príspevku s ohľadom na veľkosť písmen a diakritiku. Výsledky sú zoradené zostupne podľa creationdate a limituje sa počet vrátených záznamov.

```
if query is not None:
    cursor = connection.cursor()
    cursor.execute("""
        SELECT p.id, p.creationdate, p.viewcount, p.lasteditdate, p.title, p.body, p.answercount, p.closeddat
        (SELECT STRING_AGG(t.tagname, ', ')
        FROM post_tags AS pt
        JOIN tags AS t ON pt.tag_id = t.id
        WHERE pt.post_id = p.id) AS tags
        FROM posts AS p
        WHERE p.title ILIKE %s OR p.body ILIKE %s
        ORDER BY p.creationdate DESC
        LIMIT %s;
        """, ('%' + query + '%', '%' + query + '%', limit,))
    db_data = cursor.fetchall()
    cursor.close()
    connection.close()
    items = []
    for row in db_data:
        item = {
        "id": row[0],
        "creationdate": row[1],
        "viewcount": row[2],
        "lasteditdate": row[3],
        "title": row[4],
        "body": row[5],
        "answercount": row[6],
        "closeddate": row[7],
        "tags": row[8],
        "lastactivitydate": row[9],
        }
```

Príklad volania endpointu:

```
GET /v2/posts?duration=5&limit=2
```

Výsledok z databázy:


```
{
  "items": [
    {
      "closeddate": "2023-11-30T15:59:23.560000+00:00",
      "creationdate": "2023-11-30T15:55:32.137000+00:00",
      "duration": 3.86,
      "id": 1818849,
      "lastactivitydate": "2023-11-30T15:55:32.137000+00:00",
      "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": "2023-11-27T17:29:18.947000+00:00",
      "creationdate": "2023-11-27T17:26:57.617000+00:00",
      "duration": 2.36,
      "id": 1818386,
      "lastactivitydate": "2023-11-27T17:26:57.617000+00:00",
      "lasteditdate": null,
      "title": "Are there any libraries for parsing DWG files with LGPL, MIT, Apache, BSD?",
      "viewcount": 19
    }
  ]
}
```


6. 5. Úloha

Piata úloha má aplikovaný nasledujúci endpoint:

```python
@router.get("/v2/posts/")
async def get_posts(limit: int = Query(None), query: str = Query(None), duration: int = Query(None)):
    connection = psycopg2.connect(
        dbname=settings.DATABASE_NAME,
        user=settings.DATABASE_USER,
        password=settings.DATABASE_PASSWORD,
        host=settings.DATABASE_HOST,
        port=settings.DATABASE_PORT
    )
```

Tento SQL dopyt pracuje s tabuľkou posts. Vyberá atribúty id, creationdate, viewcount, lasteditdate, lastactivitydate, title, closeddate a vytvára nový stĺpec duration, ktorý reprezentuje čas medzi vytvorením a uzavretím príspevku v minútach s presnosťou na dve desatinné miesta. V podmienke WHERE sa filtrovajú príspevky podľa trvania medzi vytvorením a uzavretím, pričom sa používa vypočítané trvanie v minútach. Výsledky sú zoradené zostupne podľa dátumu uzavretia príspevku a limituje sa počet vrátených záznamov.

```
    elif duration is not None:
        cursor = connection.cursor()
        cursor.execute("""
        SELECT p.id, p.creationdate, p.viewcount, p.lasteditdate, p.lastactivitydate, p.title, p.closeddate,
            ROUND(EXTRACT(EPOCH FROM (p.closeddate - p.creationdate)) / 60.0, 2) AS duration
        FROM posts AS p
        WHERE ROUND(EXTRACT(EPOCH FROM (p.closeddate - p.creationdate)) / 60.0, 2) < %s
        ORDER BY p.closeddate DESC
        LIMIT %s;
        """, (duration, limit,))
        db_data = cursor.fetchall()
        cursor.close()
        connection.close()
        items = []
        for row in db_data:
            item = {
        "id": row[0],
        "creationdate": row[1],
        "viewcount": row[2],
        "lasteditdate": row[3],
        "lastactivitydate": row[4],
        "title": row[5],
        "closeddate": row[6],
        "duration": row[7]
            }
            items.append(item)

        return {"items": items}
```

Príklad volania endpointu:

```
GET /v2/posts?limit=1&query=linux
```

Výsledok z databázy:


```
{
  "items": [
    {
      "answercount": 0,
      "body": "<p>I have recently installed virtualbox on my windows 10 and trying to run Linux Ubuntu and Kali.
      "closeddate": null,
      "creationdate": "2023-12-03T04:22:43.587000+00:00",
      "id": 1819160,

      "lasteditdate": null,

      "tags": "virtual-machine",

      "title": "Keyboard not working on khali linux",
      "viewcount": 7
    }
  ]
}
```

Záver

Implementovali sme úspešne úlohy 1-5, ktoré vykonávajú SQL dopyty ktoré sú im zadané.