Peter Brenkus, ID: 120762

Databázové systémy
Dokumentácia
Zadanie č. 3

Čas cvičení: Streda 16:00
Vedúci cvičení: Ing. Ján Balažia, PhD.

Peter Brenkus, ID: 120762

Table of Contents

Outputy volaní pre toto zadanie sú v priečinku /v3moje_jsony/ v samostatných súboroch pomenovaných po jednotlivých endpointoch. Priečinok je v projektovom repozitári na github pod cestou /jsony/ alebo aj v aise v mieste odovzdania pre zadanie 3.

Peter Brenkus, ID: 120762

HTTP endpoint 1

Zadanie: Pre vybraného používateľa uskutočnite analýzu jeho získaných odznakov (badges) a to tak, že vo výstupe budú uvedené všetky získané odznaky spolu s predchádzajúcou správou, ktorú autor napísal pred samotným získaním odznaku. Ak získal odznak a pred daným odznakom nebola poslaná žiadna správa, tak sa vo výstupe takýto odznak nezobrazí. Ak získal napríklad 2 odznaky a predtým bolo poslaných viacero správ, tak vo výstupe je zobrazený iba prvý odznak s tým, že sa uvedie posledná správa, ktorá mu predchádzala.

Volanie: GET /v3/users/:user id/badge history

SQL dopyt: *(komentár treba čítať od najvnútornejšieho subquery smerom nahor)*

```
SELECT -- tuto uz iba formatujem vysledne udaje
    id,
    title,
    CASE    -- toto mi prepise "aaapost" na "post"
        WHEN type = 'aaapost' THEN 'post'
        WHEN type = 'badge' THEN 'badge'
    END AS type,
    TO_CHAR(created_at AT TIME ZONE 'UTC', --a toto urobi format casu jaky je v zadani
        'YYYY-MM-DD"T"HH24:MI:SS.FF3+00') AS created_at,
    position
FROM (
    SELECT -- tuto im pridelim poziciu podla zadania
        id, -- row_number im ju da podla typu, cize postom a odznakom zvlast
        title, -- lenze "partition by type" funkcia mi prehodi poradie post a badge dvojic.
        type,   -- cize urobi mi najprv badge potom post atd. preto mam posty najprv pomenovane s prefixom "aaa"
        created_at,-- a na konci v order by mam ako sekundarku "type", aby zostali na prvej pozicii v pare podla zadania
        row_number() OVER(PARTITION BY type ORDER BY created_at) AS position
    FROM (
        SELECT * -- tuto uz vyberiem finalne riadky
        FROM (
            SELECT  -- prida k tabulke stlpce s lag a lead hodnotami "type"
                *,  -- tie mi umoznia vyfiltrovat dvojice post+odznak podla zadania
                lag(type) OVER() AS type_before,
                lead(type) OVER() AS type_after
            FROM (
                SELECT  -- vyberie vsetky posty, ktore dany user vytvoril
                    p.id AS id,
                    p.title AS title,
                    'aaapost' AS type, -- "aaa" vyuzijem neskor na zachovanie poradia podla example
                    p.creationdate AS created_at
                FROM posts p
                WHERE owneruserid = 120

                UNION   -- spojim ich do jednej tabulky

                SELECT DISTINCT -- vyberie kazdy odznak co user ziskal
                    b.id AS id,
                    b.name AS title,
                    'badge' AS type,
                    b.date AS created_at
                FROM badges b
                WHERE b.userid = 120
                ORDER BY created_at
            ) AS sub1
        ) AS sub2
        WHERE type = 'badge' AND type_before = 'aaapost' OR -- vyberie odznak, za kt. ide post
              type = 'aaapost' AND type_after = 'badge' -- vyberie post, za ktorym ide odznak
        ORDER BY created_at
    ) AS sub3
) AS sub4
ORDER BY position, type; -- zoradi primarne podla posicie a potom prehodi posty pred odznaky
```

Príklad: http://127.0.0.1:8000/v3/users/120/badge_history
Output tohto volania je v e1.json

Čas cvičení: Streda 16:00
Vedúci cvičení: Ing. Ján Balažia, PhD.

Peter Brenkus, ID: 120762

HTTP endpoint 2

Zadanie: Pre zadaný tag vypočítajte pre jednotlivé posty (posts), ktoré majú viac ako zadaný počet komentárov (zadané v rámci API endpointu), priemernú dobu odpovede medzi jednotlivými komentármi v rámci daného postu. Vo výpise uveďte ako sa jednotlivá priemerná doba odpovede menila s pribúdajúcimi komentármi.

Volanie: GET /v3/tags/:tag/comments?count=:count

SQL dopyt: *(komentár treba čítať od najvnútornejšieho subquery smerom nahor)*

```
SELECT -- tuto vyberem vsetko z predosleho query, naformatujem casy a pridam este
    final.postid AS postid, -- priemerny cas
    final.title AS title,
    final.displayname AS displayname,
    final.text AS text,
    TO_CHAR(final.post_created_at AT TIME ZONE 'UTC',
            'YYYY-MM-DD"T"HH24:MI:SS.FF3+00') AS post_created_at,
    TO_CHAR(final.created_at AT TIME ZONE 'UTC',
            'YYYY-MM-DD"T"HH24:MI:SS.FF3+00') AS created_at,
    TO_CHAR(final.diff, 'HH24:MI:SS.FF6') AS diff,
    TO_CHAR(AVG(final.diff) OVER (PARTITION BY final.postid ORDER BY final.created_at),
            'HH24:MI:SS.FF6') AS avg
FROM (
    SELECT *, -- tuto znovu vyberiem vsetky stlpce
        CASE  -- a vypocitam cas od predosleho komentara/postu
            WHEN data.previous IS NOT NULL THEN
                CAST(data.created_at - data.previous AS INTERVAL)
            ELSE
                CAST(data.created_at - data.post_created_at AS INTERVAL)
        END AS diff
    FROM (
        SELECT -- vyberie data z comments, ktore potrebujeme
            p.id AS postid, -- a este aj z users a posts, dole to joinujem
            p.title AS title,   -- a pridam este informaciu o predoslom riadku na vypocet intervalov
            u.displayname AS displayname,
            query_comms.text AS text,
            p.creationdate AS post_created_at,
            query_comms.creationdate AS created_at,
            LAG(query_comms.creationdate) OVER(PARTITION BY p.id) AS previous
        FROM (
            SELECT -- tu uz sa vyberu vsetky komenty z postov ktore maju viac ako x komentov
                c.*
            FROM comments c
            WHERE c.postid IN (
                SELECT DISTINCT
                    postid -- vyberie tie posty, ktore maju viac ako zadany pocet komentov
                FROM (
                    SELECT -- vyberie vsetky komenty z tych postov
                        c.*, -- a prida k nim row number podla postid
                        ROW_NUMBER() OVER(PARTITION BY c.postid) AS comm_no
                    FROM comments c
                    WHERE c.postid IN (
                        SELECT DISTINCT p.id -- vyberie vsetky posty s danym tagom
                        FROM posts p
                        JOIN post_tags pt ON pt.post_id = p.id
                        JOIN tags t ON t.id = pt.tag_id
                        WHERE t.tagname = %s
                    )
                    ORDER BY c.creationdate -- zoradi podla datumu
                ) AS sub1
                WHERE comm_no > %s -- podmienka na pocet komentov
            )
            ORDER BY c.creationdate -- zoradi podla datumu
        ) AS query_comms
        JOIN posts p ON p.id = postid -- joinem to s datami z posts
        LEFT JOIN users u ON u.id = userid -- a aj z users (left join je pre pripad vymazaneho usera)
    ) AS data
) AS final;
```

Príklad: http://127.0.0.1:8000/v3/tags/networking/comments?count=40
Output tohto volania je v e2.json

Čas cvičení: Streda 16:00
Vedúci cvičení: Ing. Ján Balažia, PhD.

Peter Brenkus, ID: 120762

HTTP endpoint 3

Zadanie: Vrátte komentáre pre príspevky s tagom :tagname, ktoré boli vytvorené ako k-te v poradí (:position) zoradených podľa dátumu vytvorenia postup s limitom :limit.

Volanie: GET /v3/tags/:tagname/comments/:position?limit=:limit

SQL dopyt: *(komentár treba čítať od najvnútornejšieho subquery smerom nahor)*

```
SELECT -- tuto uz vyberiem iba potrebne udaje z danych riadkov
    querycomments.commentid AS id,
    u.displayname AS displayname,
    querycomments.body,
    querycomments.text,
    querycomments.score,
    querycomments.position
FROM (
    SELECT -- z toho nasledne vyberiem iba tie riadky kde je pozicia ktora je zadana v url
        *   -- a vyberiem ich iba tolko kolko je limit zadany v url
    FROM (
        SELECT -- vyberie stlpce zo suquery a prida stlpce z komentov
            c.id AS commentid, -- plus este ocisluje komenty pod kazdym postom od najstarsieho
            c.userid AS userid, -- to cislovanie zabezpecuje row_number
            oldest_posts.id AS postid,  -- partition by deli cislovanie podla postov
            oldest_posts.body AS body,  -- a order by zabezpeci ze najstarsi koment pod postom ma 1.
            c.text AS text,
            c.score AS score,
            row_number() OVER (PARTITION BY oldest_posts.id ORDER BY c.creationdate) AS position
        FROM (
            SELECT -- vyberie potrebne stplce z postov s tagom linux
                p.id,
                p.creationdate,
                p.body
            FROM posts p
            JOIN post_tags pt ON p.id = pt.post_id
            JOIN tags t ON pt.tag_id = t.id
            WHERE -- podmienka na tag
                t.tagname = %s
            ORDER BY -- zoradi podla datumu
                p.creationdate
        ) AS oldest_posts
        JOIN comments c ON c.postid = oldest_posts.id -- joinem to s komentami
        ORDER BY oldest_posts.creationdate, c.creationdate  -- zoradim podla datmu
    ) AS sub1
    WHERE position = %s
    LIMIT %s
) AS querycomments
JOIN users u ON u.id = querycomments.userid; -- joinem to s usermi aby som dostal displayname
```

Príklad: http://127.0.0.1:8000/v3/tags/linux/comments/2?limit=1
Output tohto volania je v e3.json

Čas cvičení: Streda 16:00
Vedúci cvičení: Ing. Ján Balažia, PhD.

Peter Brenkus, ID: 120762

HTTP endpoint 4

Zadanie: Výstupom je zoznam o veľkosti :limit vlákna pre príspevok (post) s ID postid. Vlákno začína príspevkom postid a pokračuje príspevkami, kde postid je parentid zoradený podľa dátumu vytvorenia od najstaršieho.

Volanie: GET /v3/posts/:postid?limit=:limit

SQL dopyt: *(komentár treba čítať od najvnútornejšieho subquery smerom nahor)*

```
SELECT -- vyberie potrebne udaje z postu, ktoreho id je zadane v url
    u.displayname,
    p.body,
    TO_CHAR(p.creationdate AT TIME ZONE 'UTC', 'YYYY-MM-DD"T"HH24:MI:SS.FF3+00') AS created_at
FROM
    posts p
JOIN users u ON p.owneruserid = u.id -- joinem to s users aby som mohol vybrat displayname
WHERE
    p.id = %s
UNION -- spojim to do jednej tabulky
SELECT -- vyberiem potrebne udaje z postov, ktore maju zadany post v url ako parent
    u.displayname,
    p.body,
    TO_CHAR(p.creationdate AT TIME ZONE 'UTC', 'YYYY-MM-DD"T"HH24:MI:SS.FF3+00') AS created_at
FROM
    posts p
JOIN users u ON p.owneruserid = u.id -- joinem s usermi aby som dostal displayname zasa
WHERE
    parentid = %s -- podmienka na parent id
ORDER BY          -- zoradim podla datumu
    created_at ASC
LIMIT %s; -- vyberiem iba zadany limit riadkov z tohto query
```

Príklad: http://127.0.0.1:8000/v3/posts/2154?limit=2
Output tohto volania je v e4.json

Čas cvičení: Streda 16:00
Vedúci cvičení: Ing. Ján Balažia, PhD.