Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Databázové Systémy

Zadanie č.2

Protokol k riešeniu

Akademický rok 2022/2023

Meno: Ján Ágh
Cvičiaci: Ing. Martin Binder

Dátum: 10.3.2023
Počet strán: 13

Obsah

1 Zoznam spolucestujúcich

Query:

```
SELECT array_to_json(array_agg(passengers))
    from (
      SELECT
        t.passenger_id AS id,
        t.passenger_name AS name,
        COUNT(t.passenger_id) AS flights_count,
        ARRAY_AGG(tf.flight_id ORDER BY tf.flight_id ASC) AS flights
      from bookings.tickets AS t
      INNER JOIN bookings.ticket_flights AS tf
      ON t.ticket_no = tf.ticket_no
      WHERE t.passenger_id != '{ passenger_id }' AND tf.flight_id IN (
        SELECT inner_tf.flight_id
        from bookings.tickets AS inner_t
        INNER JOIN bookings.ticket_flights AS inner_tf
        ON inner_t.ticket_no = inner_tf.ticket_no
        WHERE inner_t.passenger_id = '{ passenger_id }'
      )
      GROUP BY t.passenger_id, t.passenger_name
      ORDER BY flights_count DESC, id ASC
    ) passengers;
```

Opis:

Začíname s tabuľkou *tickets*, ku ktorej pomocou INNER JOIN pripojíme tabuľku *ticket_flights* podľa hodnoty atribútu ticket_no. Následne vovnútri WHERE filtrujeme záznamy, pričom zaujímajú nás spolucestujúci daného pasažiera – čiže všetky záznamy, kde sa passenger_id nerovná jeho id (nemôže byť svoj vlastný spolucestujúci) a zároveň id letu sa musí nachádzať v zozname letov, ktoré daný pasažier využil (zoznam sa vráti zo subquery). Uvedené subquery nám zo zjednotenia tabuliek tickets a ticket_flights vráti zoznam všetkých id letov, ktoré náš zvolený pasažier využil. Potom využitím GROUP BY zoskupíme záznamy podľa id pasažiera aj mena pasažiera, aby sme odstránili duplikátne záznamy pre jendého konkrétneho spolucestujúceho, a v rámci SELECT zvolíme potrebné stĺpce, spočítame počet spolucestujúcich a pomocou ARRAY_AGG vytvoríme zoznam hodnôt, ktorý naplníme id spoločných letov a zoradíme vzostupne. Nakoniec celý zoznam zoradíme podľa počtu spoločných letov zostupne a druhotne podľa id pasažiera vzostupne. Databáza daný výstup automaticky konvertuje do json formátu (pomocou array_to_json() funkcie zobrazenej modrou).

Volanie endpointu z prehliadača a výsledok:

127.0.0.1:8000/v1/passengers/8360%20311602/companions


```
{
  "results": [
    {
      "id": "0020 018674",
      "name": "VLADIMIR NIKOLAEV",
      "flights_count": 1,
      "flights": [
        187570
      ]
    },
    {
      "id": "0096 299351",
      "name": "IGOR KOROLEV",
      "flights_count": 1,
      "flights": [
        187570
      ]
    },
```

2 Detail letu

Query:

```
SELECT DISTINCT
        b.book_ref AS id,
        b.book_date,
        t.ticket_no AS bp_id,
        t.passenger_id,
        t.passenger_name,
        bp.boarding_no,
        f.flight_no,
        bp.seat_no AS seat,
        f.aircraft_code,
        f.arrival_airport,
        f.departure_airport,
        f.scheduled_arrival,
        f.scheduled_departure
    from bookings.bookings AS b
    INNER JOIN bookings.tickets AS t
    ON b.book_ref = t.book_ref
    INNER JOIN bookings.boarding_passes AS bp
    ON t.ticket_no = bp.ticket_no
    INNER JOIN bookings.flights AS f
    ON bp.flight_id = f.flight_id
    WHERE b.book_ref = '{ booking_id }'
    ORDER BY bp_id ASC, bp.boarding_no ASC;
```

Opis:

V prvom kroku je potrebné pomocou INNER JOIN spojiť viacero tabuliek – tabuľku *bookings* postupne zjednotíme s tabuľkou *tickets* podľa atribútu book_ref, s tabuľkou *boarding_passes* podľa atribútu ticket_no a nakoniec s tabuľkou *flights*, kde porovnávame s atribútom flight_id z tabuľky boarding_passes. Takto dostaneme v jednom celku všetky informácie, ktoré potrebujeme vrátiť. Následne v rámci WHERE vyfiltrujeme iba záznamy, ktorých book_ref sa zhoduje so zadanou hodnotou, v SELECT si zvolíme potrebné stĺpce a odstránime duplikáty a výsledok zoradíme najprv podľa ticket_no vzostupne, potom podľa boarding_no vzostupne. Spracovanie výstupu do json formátu prebieha v endpointe.

Volanie endpointu z prehliadača a výsledok:

127.0.0.1:8000/v1/bookings/000012


```
{
  "result": {
      "id": "000012",
      "book_date": "2017-07-14T08:02:00+02:00",
      "boarding_passes": [
          {
              "id": "0005432527326",
              "passenger_id": "9091 269355",
              "passenger_name": "TAMARA ZAYCEVA",
              "boarding_no": 130,
              "flight_no": "PG0224",
              "seat": "28C",
              "aircraft_code": "773",
              "arrival_airport": "AER",
              "departure_airport": "SVO",
              "scheduled_arrival": "2017-07-28T17:50:00+02:00",
              "scheduled_departure": "2017-07-28T16:05:00+02:00"
          },
          {
              "id": "0005432527326",
              "passenger_id": "9091 269355",
              "passenger_name": "TAMARA ZAYCEVA",
              "boarding_no": 171,
              "flight_no": "PG0013",
              "seat": "13G",
              "aircraft_code": "773",
              "arrival_airport": "SVO",
              "departure_airport": "AER",
              "scheduled_arrival": "2017-08-09T19:00:00+02:00",
              "scheduled_departure": "2017-08-09T17:15:00+02:00"
          }
      ]
  }
}
```

3 Neskoré odlety

Query:

```
SELECT array_to_json(array_agg(late))
    from (
      SELECT DISTINCT
        f.flight_id,
        f.flight_no,
        FLOOR(
          EXTRACT(
            epoch from (f.actual_departure - f.scheduled_departure)
          ) / 60
        ) AS delay
      from bookings.flights AS f
      WHERE CAST(
        EXTRACT(epoch from (f.actual_departure - f.scheduled_departure)) AS INT
      ) / 60 >= { delay }
      ORDER BY delay DESC, f.flight_id ASC
    ) late;
```

Opis:

V prvom kroku z tabuľky *flights* načítame všetky lety, ktoré vyhovujú podmienke v časti WHERE – meškali minimálne zadané množstvo minút. Podmienka je realizovaná nasledovne: pomocou funkcie EXTRACT dokážeme zistiť konkrétnu časť dátumu alebo realizovať výpočty s dátumami. V tomto prípade vovnútri EXTRACT voláme funkciu epoch, pomocou ktorej vypočítame rozdiel medzi reálnym odletom a plánovaným odletom v sekundách. Túto hodnotu konvertujeme na INT a delíme 60, aby sme dostali počet minút. Výsledok následne porovnáme so zadanou hodnotou. Následne v rámci SELECT si zvolíme potrebné stĺpce a znova vypočítame meškanie letu v minútach, ktoré v tomto prípade zaokrúhlime na celé čísla nadol. Výsledné hodnoty zoradíme podľa meškania zostupne a druhotne podľa id letu. Databáza daný výstup automaticky konvertuje do json formátu (pomocou array_to_json() funkcie zobrazenej modrou).

Volanie endpointu z prehliadača a výsledok:

127.0.0.1:8000/v1/flights/late-departure/280


```
{
  "results": [
    {
      "flight_id": 157571,
      "flight_no": "PG0073",
      "delay": 303
    },
    {
      "flight_id": 186524,
      "flight_no": "PG0040",
      "delay": 284
    },
    {
      "flight_id": 126166,
      "flight_no": "PG0533",
      "delay": 282
    },
    {
      "flight_id": 56731,
      "flight_no": "PG0132",
      "delay": 281
    },
    {
      "flight_id": 102938,
      "flight_no": "PG0531",
      "delay": 281
    }
  ]
}
```

4 Linky, ktoré obslúžili najviac pasažierov

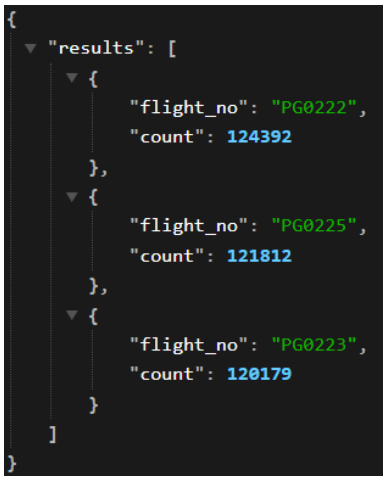
Query:

```
SELECT array_to_json(array_agg(top))
    from (
        SELECT
            f.flight_no,
            COUNT(bp.flight_id) AS count
        from bookings.flights AS f
        INNER JOIN bookings.boarding_passes AS bp
        ON f.flight_id = bp.flight_id
        WHERE f.status = 'Arrived'
        GROUP BY f.flight_no
        ORDER BY count DESC, f.flight_no ASC
        LIMIT { limit }
    ) top;
```

Opis:

V prvok kroku vykonáme INNER JOIN tabuliek *flights* a *boarding passes* podľa atribútu flight_id, čím dostaneme spojenú tabuľku, kde sú jednotlivé boarding_passes namapované na konkrétne flights. Následne v rámci WHERE aplikujeme filter, podľa ktorého nás zaujímajú iba flights, ktorých status je Arrived (keďže musíme pracovať iba s už dokončenými letmi). V ďalšom kroku dáta zoskupíme (GROUP BY) podľa čísla linky, čím dostaneme jedno entry pre každú linku a vykonáme SELECT čísel liniek spolu s počtom flight_id nachádzajúcich sa v boarding passes, podľa čoho vieme jednoznačne určiť počet cestujúcich pre danú linku. V poslednom kroku výsledky zoradíme zostupne podľa počtu pasažierov a v prípade rovnosti podľa čísla linky a aplikujeme LIMIT zadaný v dopyte. Databáza daný výstup automaticky konvertuje do json formátu (pomocou array_to_json() funkcie zobrazenej modrou).

Volanie endpointu z prehliadača a výsledok:



5 Naplánované linky

Query:

```
SELECT array_to_json(array_agg(departures))
    from (
      SELECT
        f.flight_id,
        f.flight_no,
        f.scheduled_departure
      from bookings.flights AS f
      WHERE
        f.status = 'Scheduled' AND
        f.departure_airport = '{ airport }' AND
        EXTRACT(isodow from f.scheduled_departure) = { day }
      ORDER BY f.scheduled_departure ASC, f.flight_id ASC
    ) departures;
```

Opis:

V prvom kroku načítame z tabuľky *flights* všetky entries, ktoré vyhovujú nasledovnej podmienke: sú ešte len naplánované (Scheduled), odchádzajú zo zadaného letiska a odchádzajú v konkrétny deň v týždni (od 1 po 7). Určenie, ktoré dátumy spadajú na konkrétny zvolený deň je riešené pomocou EXTRACT (dokáže vrátiť časť dátumu, napr. deň, mesiac, hodinu…) v kombinácii s funkciou isodow (vráti konkrétny deň v týždni podľa ISO formátu – od pondelka (1) do nedele (7)). Následne sa z vyhovujúcich dát pomocou SELECT zvolia potrebné stĺpce a výsledok je zoradený podľa času odchodu od najbližších a druhotne podľa id letu. Databáza daný výstup automaticky konvertuje do json formátu (pomocou array_to_json() funkcie zobrazenej modrou).

Volanie endpointu z prehliadača a výsledok:


```
{
  "results": [
    {
      "flight_id": 173390,
      "flight_no": "PG0328",
      "scheduled_departure": "2017-08-19T01:45:00+02:00"
    },
    {
      "flight_id": 172707,
      "flight_no": "PG0660",
      "scheduled_departure": "2017-08-19T02:55:00+02:00"
    },
    {
      "flight_id": 172468,
      "flight_no": "PG0201",
      "scheduled_departure": "2017-08-19T11:10:00+02:00"
    },
```

6 Všetky destinácie zo zadaného letiska

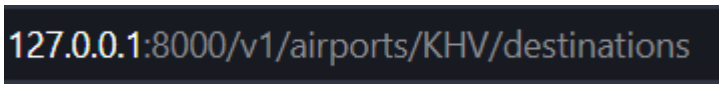
Query:

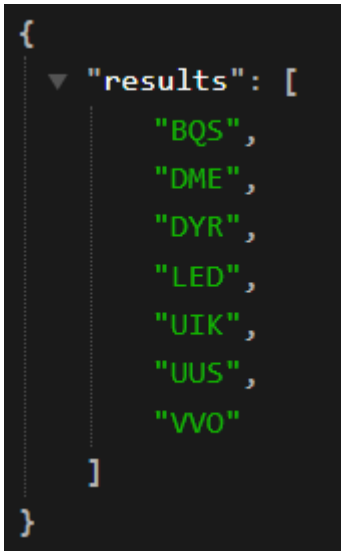
```
SELECT DISTINCT
        f.arrival_airport
    from bookings.flights AS f
    WHERE f.departure_airport = '{ airport }'
    ORDER BY f.arrival_airport ASC;
```

Opis:

Pri tejto query sa pracuje iba s dátami z tabuľky *flights*, keďže každý let obsahuje údaj o tom, z ktorého letiska vychádza a na ktorom letisku pristáva. V rámci WHERE sa zvolia všetky lety, kde kód východzieho letiska sa rovná zadanému kódu. Týmto získame všetky lety, ktoré z daného letiska odchádzajú. Následne v rámci SELECT už iba stačí zvoliť si stĺpec obsahujúci cieľové letiská a odstrániť duplikáty a máme zoznam destinácií. V poslednom kroku sa vovnútri ORDER BY zoradia kódy letísk podľa abecedy. Spracovanie výstupu do json formátu prebieha v endpointe.

Volanie endpointu z prehliadača a výsledok:





7 Vyťaženosť letov pre konkrétnu linku

Query:

```
SELECT array_to_json(array_agg(airlines))
    from (
      WITH aircraft_detail(code, seats) AS (
            SELECT
                  s.aircraft_code,
                  COUNT(s.seat_no)
            from bookings.seats AS s
            GROUP BY s.aircraft_code
      )
      SELECT
        f.flight_id AS id,
        COUNT(DISTINCT tf.ticket_no) AS load,
      (
            SELECT ad.seats
            FROM aircraft_detail AS ad
            WHERE ad.code = f.aircraft_code
      ) AS aircraft_capacity,
        ROUND(
            CAST(COUNT(DISTINCT tf.ticket_no) AS numeric) /
            CAST((
                  SELECT ad.seats
                  FROM aircraft_detail AS ad
                  WHERE ad.code = f.aircraft_code) AS numeric) * 100,
          2
        )::REAL AS percentage_load
      from bookings.flights AS f
      LEFT JOIN bookings.ticket_flights AS tf
      ON f.flight_id = tf.flight_id
      WHERE f.flight_no = '{ flight_no }'
      GROUP BY f.flight_id, f.aircraft_code
      ORDER BY id ASC
    ) airlines;
```

Opis:

Pri tejto query si pomocou WITH vytvoríme pomocnú tabuľku *aircraft_detail*, ktorá bude obsahovať kód každého typu lietadla a k nemu priradenú kapacitu (načítame tabuľku sedadiel *seats* a zoskupíme ju podľa kódu lietadla). Následke k tabuľke *flights* pomocou LEFT JOIN pripojíme tabuľku *ticket_flights* podľa flight_id. LEFT JOIN sa používa z dôvodu, aby sme po zjednoteniach mali stále k dispozícii celý zoznam z tabuľky flights. Nasledovne vo WHERE špecifikujeme, že nás zaujímajú iba lety, ktoré patria pod zvolenú linku. Potom pomocou GROUP BY zoskupíme záznamy podľa flight_id, keďže potrebujeme pre jeden let len jedno entry, a aircraft_code, lebo danú hodnotu potrebujeme v select statemente. V rámci SELECT

si zvolíme potrebné stĺpce, pričom pri load spočítavame iba unikátne hodnoty ticket_no (inak by niektoré lístky mohli byť započítané aj viackrát) a informáciu o kapacite dostaneme z pomocnej tabuľky aircraft_detail (vyberáme z nej hodnotu seats zo záznamu, ktorého aircraft_code sa rovná s aircraft_code súčasného letu). Výslednú obsadenosť v percentách zaokrúhlime na dve desatinné miesta. Dôležité je použitie ::REAL – bez neho by sme v prípade plného obsadenia nedostali výsledok 100, ale 100.0, čo nie je správne. V poslednom kroku výsledky zoradíme vzostupne podľa id letu. Databáza daný výstup automaticky konvertuje do json formátu (pomocou array_to_json() funkcie zobrazenej modrou).

Volanie endpointu z prehliadača a výsledok:

127.0.0.1:8000/v1/airlines/PG0328/load


```
{
  "results": [
    {
      "id": 173062,
      "aircraft_capacity": 97,
      "load": 86,
      "percentage_load": 88.66
    },
    {
      "id": 173063,
      "aircraft_capacity": 97,
      "load": 84,
      "percentage_load": 86.6
    },
    {
      "id": 173064,
      "aircraft_capacity": 97,
      "load": 87,
      "percentage_load": 89.69
    },
```

8 Priemerná vyťaženosť linky

Query:

```
WITH aircraft_detail(code, seats) AS (
        SELECT
           s.aircraft_code,
           COUNT(s.seat_no)
        from bookings.seats AS s
        GROUP BY s.aircraft_code
    ),
    weeks_detail(day, code, avg) AS (
        SELECT
           EXTRACT(isodow from f.scheduled_departure),
           f.aircraft_code,
           CAST(COUNT(tf.ticket_no) AS numeric) /
           CAST(COUNT(DISTINCT f.flight_id) AS numeric)
        from bookings.flights AS f
        INNER JOIN bookings.ticket_flights AS tf
        ON f.flight_id = tf.flight_id
        WHERE f.flight_no = '{ flight_no }'
        GROUP BY EXTRACT(isodow from f.scheduled_departure), f.aircraft_code
    )
    SELECT
        wd.day,
        ROUND((wd.avg / CAST(ad.seats AS numeric)) * 100, 2)::REAL
    from weeks_detail AS wd
    INNER JOIN aircraft_detail AS ad
    ON wd.code = ad.code;
```

Opis:

Základom tejto query sú dve pomocné tabuľky vytvorené operátorom WITH – *aircraft_detail* obsahujúci kód každého typu lietadla spolu s ich maximálnou kapacitou (načítame všetky záznamy tabuľky *seats*, zoskupíme ich podľa kódu lietadla (dostaneme jedno entry pre každý typ lietadla) a spočítame počet priradených sedadiel (takto získame kapacitu)) a *weeks_detail* obsahujúci priemerný počet pasažierov zadanej linky pre každý deň v týždni (pomocou INNER JOIN spojíme tabuľky *flights* a *ticket_flights* podľa flight_id (takto namapujeme na každý let k nemu prislúchajúce lístky), ale iba pre zadanú linku (ošetrené vo WHERE operátore), následne vyhovujúce záznamy zoskupíme podľa dňa v týždni určeného pomocou funkcie EXTRACT v kombinácii s parametrom isodow a podľa aircraft_code a v SELECT statemente vyberáme deň v týždni, kód lietadla (pomocou neho budeme neskôr pripájať tabuľku kapacít) a vypočítame priemerný počet pasažierov v daný deň ako podiel počtu lístkov a počtu letov). Keď sú pomocné tabuľky hotové, v dolnej časti query ich pomocou INNER JOIN spojíme podľa kódu lietadla a následne už iba v SELECT vypočítame pre každý deň priemernú obsadenosť ako podiel priemerného počtu pasažierov (z tabuľky weeks_detail) a kapacity lietadla (z tabuľky aircraft_detail) vynásobený 100 a zaokrúhlený na dve desatinné miesta.

Volanie endpointu z prehliadača a výsledok: