Zadanie: Indeksy i Widoki w bazie danych `epstein_files`

Wprowadzenie

Baza danych `epstein_files` zawiera prawdziwe dane z akt sprawy Jeffreya Epsteina — loty, e-maile, osoby i powiązania między nimi. Baza zawiera ponad 150 000 wierszy danych rozłożonych na 16 tabel.

Część 1 — Indeksy

Indeks w bazie danych działa jak indeks w książce — zamiast przeszukiwać każdą stronę (wiersz), baza danych może od razu skoczyć do właściwego miejsca. Indeksy przyspieszają zapytania z `WHERE`, `JOIN`, `ORDER BY` i `GROUP BY`, ale spowalniają operacje `INSERT` i `UPDATE` oraz zajmują dodatkowe miejsce na dysku.

Indeks warto tworzyć na kolumnie, gdy:

- często filtrujemy po niej w klauzuli `WHERE`
- używamy jej do łączenia tabel (`JOIN`)
- ma wysoką selektywność (wiele różnych wartości)

Składnia tworzenia indeksu:

```
CREATE INDEX nazwa_indeksu ON nazwa_tabeli(nazwa_kolumny);
```

Dla kolumn tekstowych typu `VARCHAR` o dużej długości podajemy prefiks:

```
CREATE INDEX nazwa_indeksu ON nazwa_tabeli(nazwa_kolumny(100));
```

Sprawdzenie istniejących indeksów:

```
SHOW INDEXES FROM nazwa_tabeli;
```

Sprawdzenie czy zapytanie korzysta z indeksu:

```
EXPLAIN SELECT ...;
```

Przykład

Chcemy przyspieszyć wyszukiwanie e-maili po dacie. Najpierw wyszukujemy bez indeksu:

```
SELECT COUNT(*) FROM emails WHERE YEAR(date) = 2016;
```

Tworzymy indeks:

```
CREATE INDEX idx_emails_date ON emails(date);
```

Uruchamiamy zapytanie ponownie i porównujemy czas wykonania. Możemy też sprawdzić plan zapytania:

```
EXPLAIN SELECT COUNT(*) FROM emails WHERE YEAR(date) = 2016;
```

W kolumnie `key` powinien pojawić się nasz indeks `idx_emails_date`.

Zadania

Zadanie 1.1 — Sprawdź istniejące indeksy na tabelach `emails`, `flights` i `persons`. Które kolumny już mają indeksy? Dlaczego właśnie te?

Zadanie 1.2 — Utwórz indeks przyspieszający wyszukiwanie e-maili po identyfikatorze wątku (`thread_id`). Następnie sprawdź czas zapytania:

```
SELECT thread_id, COUNT(*) AS liczba_emaili
FROM emails
WHERE thread_id IS NOT NULL
  AND thread_id NOT LIKE 'db-thread-%'
GROUP BY thread_id
ORDER BY liczba_emaili DESC
LIMIT 10;
```

Zadanie 1.3 — Utwórz indeksy na tabeli `flights` przyspieszające zapytania o trasy lotów (kolumny `origin` i `destination`) oraz o daty lotów. Pamiętaj o użyciu prefiksu dla kolumn tekstowych.

Zadanie 1.4 — Utwórz indeksy na tabeli `persons` dla kolumn `category` i `status`.

Zadanie 1.5 — Utwórz indeks na kolumnie `display_name` w tabeli `email_addresses` oraz indeks na kolumnie `relationship_type` w tabeli `person_connections`.

Część 2 — Widoki

Widok (`VIEW`) to zapisane zapytanie `SELECT`, które zachowuje się jak tabela — można z niego czytać, filtrować i łączyć z innymi tabelami. Widoki:

- upraszczają złożone zapytania (piszesz je raz, używasz wielokrotnie)
- ukrywają złożoność bazy przed użytkownikiem
- nie przechowują danych — za każdym razem wykonują zapytanie na żywo

Składnia tworzenia widoku:

```
CREATE OR REPLACE VIEW nazwa_widoku AS
SELECT ...;
```

Widok odpytujemy jak tabelę:

```
SELECT * FROM nazwa_widoku WHERE warunek;
```

Usunięcie widoku:

```
DROP VIEW nazwa_widoku;
```

Przykład

Tworzymy widok `v_email_full` pokazujący każdy e-mail z rozwiązaną tożsamością nadawcy (imię, adres e-mail, powiązana osoba). Bez widoku musielibyśmy za każdym razem pisać trzy JOINy. Z widokiem wystarczy jedno proste zapytanie.

```
CREATE OR REPLACE VIEW v_email_full AS
SELECT
    e.id             AS email_id,
    e.date,
    e.subject,
    e.thread_id,
    e.snippet,
    e.has_attachment_evidence,
    ea.display_name  AS sender_name,
    ea.email_address AS sender_email,
    p.id             AS sender_person_id,
    p.name           AS sender_person_name,
    p.category       AS sender_category,
    e.body_truncated,
    e.url
FROM emails e
LEFT JOIN email_addresses ea ON e.sender_id  = ea.id
LEFT JOIN persons p          ON ea.person_id = p.id;
```

Teraz możemy np. znaleźć wszystkie e-maile wysłane przez polityków jednym prostym zapytaniem:

```
SELECT email_id, date, subject, sender_person_name
FROM v_email_full
WHERE sender_category = 'politician'
ORDER BY date;
```

Zadania

Zadanie 2.1 — Utwórz widok `v_person_summary` zastępujący usunięte kolumny denormalizowane. Widok powinien pokazywać dla każdej osoby jej dane podstawowe oraz wyliczone: liczbę lotów, liczbę adresów e-mail, liczbę wysłanych e-maili i liczbę połączeń z innymi osobami.

Wskazówka: skorzystaj z tabel `persons`, `flight_passengers`, `email_addresses`, `emails` i `person_connections`. Pamiętaj, że połączenia są dwukierunkowe — osoba może być zarówno `person_id` jak i `target_id`.

Po utworzeniu widoku sprawdź za pomocą `SELECT` (używając tylko widoku):

- Kto odbył najwięcej lotów?
- Kto wysłał najwięcej e-maili?
- Ile lotów odbył Jeffrey Epstein?

Zadanie 2.2 — Utwórz widok `v_flight_manifest` — manifest lotu pokazujący każdy lot razem z danymi pasażerów. Widok powinien zawierać: identyfikator i datę lotu, lotnisko wylotu i przylotu, numer i model samolotu, pilota, nazwę pasażera z bazy oraz dane powiązanej osoby (id, imię, kategorię, status).

Po utworzeniu widoku odpowiedz:

- Wylistuj wszystkich pasażerów lotu o id `f-0001`
- Którym samolotem (numer samolotu) najczęściej latał Jeffrey Epstein?
- Ile lotów odbył Donald Trump?

Zadanie 2.3 — Widok `v_email_full` został już utworzony w przykładzie powyżej. Odpowiedz (`SELECT`) używając tego widoku:

- Ile e-maili wysłał Jeffrey Epstein?
- Jakie są tematy e-maili wysłanych przez osoby z kategorii `legal`?
- Ile e-maili posiada dowody na załączniki (`has_attachment_evidence = 1`)?

Zadanie 2.4 — Utwórz widok `v_person_connections_named` — widok powiązań między osobami z rozwiązanymi nazwami po obu stronach. Tabela `person_connections` jest dwukierunkowa (każda para A→B i B→A istnieje jako osobny wiersz). Widok powinien deduplikować pary — pokazuj każdą parę tylko raz.

Wskazówka: można to osiągnąć warunkiem `pc.person_id < pc.target_id`.

Po utworzeniu widoku odpowiedz:

- Wylistuj wszystkie powiązania Ghislaine Maxwell
- Ile unikalnych par powiązań istnieje w bazie?
- Kto jest powiązany z Billem Clintonem i jakiego typu są te powiązania?

Zadanie 2.5 — Utwórz widok `v_route_stats` — statystyki tras lotów. Widok powinien grupować loty według trasy (origin + destination) i pokazywać: łączną liczbę lotów, datę pierwszego i ostatniego lotu, liczbę unikalnych osób na tej trasie oraz liczbę różnych samolotów.

Po utworzeniu widoku odpowiedz:

- Jaka jest najczęstsza trasa lotów?
- Ile różnych osób leciało trasą Teterboro → Palm Beach?
- Która trasa miała największy rozstęp dat między pierwszym a ostatnim lotem?

Zadanie 2.6 — Utwórz widok `v_location_visitors` — lokalizacje z liczbą odwiedzających. Widok powinien łączyć tabele `locations`, `location_flights`, `flights` i `flight_passengers`, pokazując dla każdej lokalizacji: łączną liczbę lotów, liczbę unikalnych odwiedzających oraz daty pierwszej i ostatniej wizyty.

Po utworzeniu widoku odpowiedz:

- Które lotnisko obsłużyło najwięcej unikalnych pasażerów?
- Kiedy odbyła się pierwsza i ostatnia wizyta na Little St. James Island?
- Ile lokalizacji nie ma żadnych powiązanych lotów?

Część 3 — Zadanie łączone

Używając **tylko widoków** (bez bezpośredniego odwoływania się do tabel bazowych) napisz zapytania odpowiadające na następujące pytania:

1. Wymień 5 osób z największą liczbą lotów spośród tych, które mają status `convicted` lub `deceased`
2. Znajdź wszystkie e-maile wysłane przez osoby, które leciały na Little St. James Island — wyświetl temat, datę i nazwę nadawcy
3. Dla każdego typu relacji (`relationship_type`) podaj średnią wartość `strength` i liczbę par
4. Które osoby leciały zarówno do Paryża (`Paris Le Bourget Airport`) jak i na St. Thomas (`Cyril E. King Airport (St. Thomas)`)?
5. Znajdź osoby, które pojawiają się w danych zarówno jako pasażerowie lotów jak i jako nadawcy e-maili — posortuj wyniki malejąco po łącznej aktywności (loty + e-maile)

Wskazówki ogólne

- Widok można odpytywać jak tabelę: `SELECT * FROM v_person_summary WHERE category = 'politician'`
- Widoki można łączyć ze sobą i z tabelami bazowymi za pomocą `JOIN`
- `SHOW INDEXES FROM nazwa_tabeli` — pokazuje wszystkie indeksy tabeli
- `EXPLAIN SELECT ...` — pokazuje czy MySQL używa indeksu; sprawdź kolumnę `key`
- Jeśli zapytanie jest wolne mimo indeksu, MySQL mógł zdecydować że przeszukanie całej tabeli jest szybsze — dzieje się tak przy małej selektywności kolumny
- Widoki nie przyspieszają zapytań same w sobie — służą czytelności i wielokrotnemu użyciu; indeksy na tabelach bazowych nadal mają znaczenie