Zapytania SQL w projekcie

Tabela: `autor`

```
1       INSERT INTO kategoria (nazwa) VALUES ('Fantasy'), ('Science␣Fiction'), ('Horror'), ('
            Romans'),('Krymina '), ('Thriller'), ('Literatura␣faktu'), ('Biografia'), ('
            Przygodowa'), ('Dla␣dzieci');
```

Listing 1: Tworzenie tabeli `autor`

Na tej samej zasadzie dane zostały wprowadzone do pozostałych tabel w funkcji init2() podczas inicjalizacji bazy danych do testów.

Funkcja uzupełniająca egzemplarze

```
1       CREATE OR REPLACE FUNCTION dodaj_egzemplarze(id_ksiazki INTEGER, id_biblioteki INTEGER
            , ilosc INTEGER) RETURNS VOID AS $$
2       DECLARE
3           i INTEGER := 1;
4       BEGIN
5           WHILE i <= ilosc LOOP
6               INSERT INTO egzemplarz (ksiazka_id, biblioteka_id, stan, data_nabycia)
7               VALUES (id_ksiazki, id_biblioteki, 'Nowy', NOW());
8               i := i + 1;
9           END LOOP;
10      END;
11      $$ LANGUAGE plpgsql;
```

Listing 2: Uzupełnienie egzemplarzy. Wykorzystana została w pętli, co usprawniło przypisywanie

Widok książek

```
1   CREATE OR REPLACE VIEW project.widok_ksiazki AS
2   SELECT
3       ks.ksiazka_id AS ksiazka_id,
4       ks.tytul AS tytul,
5       ks.rok_wydania AS rok_wydania,
6       kat.nazwa AS kategoria,
7       CONCAT(au.imie, '␣', au.nazwisko) AS autor,
8       bib.miejscowosc AS biblioteka,
9       ks.kod AS kod,
10      COUNT(*) AS ilosc_wszystkich,
11      COUNT(CASE WHEN eg.stan = 'Nowy' OR eg.stan = 'zwrocono' THEN 1 ELSE NULL END) AS
            ilosc_dostepnych
12  FROM
13      project.ksiazka ks
14          JOIN
15      project.kategoria kat ON ks.kategoria_id = kat.kategoria_id
16          JOIN
17      project.ksiazka_autor ka ON ks.ksiazka_id = ka.ksiazka_id
18          JOIN
19      project.autor au ON ka.autor_id = au.autor_id
20          JOIN
21      project.egzemplarz eg ON ks.ksiazka_id = eg.ksiazka_id
22          JOIN
23      project.biblioteka bib ON eg.biblioteka_id = bib.biblioteka_id
24  GROUP BY
25      ks.ksiazka_id, ks.tytul, ks.rok_wydania, kat.nazwa, au.imie, au.nazwisko, bib.
            miejscowosc, ks.kod;
```

Listing 3: Widok umożliwiający wyświetlanie książek w aplikacji

Widok wypożyczeń

```sql
CREATE VIEW project.widok_wypozyczen AS
SELECT
    ks.tytul AS tytul,
    kat.nazwa AS kategoria,
    CONCAT(au.imie, ' ', au.nazwisko) AS autor,
    bib.miejscowosc AS biblioteka,
    ks.kod AS kod,
    wyp.start AS start,
    wyp.koniec AS koniec,
    wyp.zwrot AS zwrot,
    wyp.wypozyczenie_id AS wypozyczenie_id,
    uz.nr_karty_bibliotecznej AS numer_karty_bibliotecznej
FROM
    wypozyczenie wyp
JOIN
    egzemplarz e ON wyp.egzemplarz_id = e.egzemplarz_id
JOIN
    ksiazka ks ON e.ksiazka_id = ks.ksiazka_id
JOIN
    kategoria kat ON ks.kategoria_id = kat.kategoria_id
JOIN
    ksiazka_autor ka ON ks.ksiazka_id = ka.ksiazka_id
JOIN
    autor au ON ka.autor_id = au.autor_id
JOIN
    biblioteka bib ON e.biblioteka_id = bib.biblioteka_id
JOIN
    uzytkownik uz ON wyp.uzytkownik_id = uz.uzytkownik_id;
```

Listing 4: Widok umożliwiający wyświetlanie wypożyczeń w aplikacji

Walidacja użytkownika

```sql
      SELECT uzytkownik_id
      FROM project.uzytkownik
      WHERE imie = $1 AND nazwisko = $2 AND nr_karty_bibliotecznej = $3;
```

Listing 5: Sprawdzenie czy użytkownik chcący wypożyczyć książkę istnieje

Walidacja książki

```sql
     SELECT ksiazka_id
      FROM ksiazka
      WHERE kod = $1;
```

Listing 6: Sprawdzenie czy książka o danym kodzie istnieje

Wybranie pojedycznego egzemplarza

```sql
SELECT egzemplarz_id
FROM egzemplarz
WHERE ksiazka_id = $1
  AND biblioteka_id = $2
  AND stan IN ('Nowy', 'zwrocono')
LIMIT 1;
```

Listing 7: Sposrod dostepnych egzemplarzy wybieramy dowolny, dla ktorego zmienimy dane

Wprowadzenie danych do tabeli wypozyczenie

```
INSERT INTO wypozyczenie (egzemplarz_id, uzytkownik_id, start, koniec)
VALUES ($1, $2, CURRENT_DATE, CURRENT_DATE + INTERVAL '30 days')
RETURNING wypozyczenie_id;
```

Listing 8: Wprowadzenie danych do tabeli wypozyczenie, zakladamy wypozyczanie ksiazki na 1 miesiac

Aktualizacja stanu egzemplarza

```
UPDATE egzemplarz
 SET stan = 'Wypozyczony'
 WHERE egzemplarz_id = $1;
```

Listing 9: Aktualizacja stanu egzemplarza ktory zostal wypozyczony na podstawie id

* Cały proces wypożyczania wykonywany jest przy pomocy BEGIN - COMMIT - ROLLBACK

Pobieranie danych z widoku ksiazki

Pobieranie danych z widoku wypozyczen

```
SELECT * FROM project.widok_wypozyczen
WHERE numer_karty_bibliotecznej = $1
```

Listing 10: Pobieranie danych z widoku wypozyczen na podstawie numeru karty bibliotecznej

Pobieranie egzemplarza podczas operacji zwrotu

```
SELECT e.egzemplarz_id, b.biblioteka_id
FROM project.wypozyczenie w
JOIN project.egzemplarz e ON w.egzemplarz_id = e.egzemplarz_id
JOIN project.biblioteka b ON e.biblioteka_id = b.biblioteka_id
WHERE w.wypozyczenie_id = $1
```

Listing 11: Pobieranie egzemplarza podczas operacji zwrotu na podstawie id wypozyczenia, pobierane jest rowniez id biblioteki wymagane do sprawdzenia czy administrator moze ja wypozyczyc

Pobieranie admina

```
SELECT 1
FROM project.administrator a
JOIN project.zarzadzanie z ON a.administrator_id = z.administrator_id
WHERE a.login = $1 AND z.biblioteka_id = $2
```

Listing 12: Sprawdzanie czy aktualnie zalogowany admin jest przypisany do biblioteki do ktorej jest zwracana ksiazka

Aktualizacja daty zwrotu

```
UPDATE project.wypozyczenie
SET zwrot = CURRENT_DATE
WHERE wypozyczenie_id = $1
```

Listing 13: Aktualizacja daty zwrotu wypozyczenia

Rejestracja administratora

```
INSERT INTO project.administrator (login, haslo, email, telefon) VALUES ($1, $2, $3, $4)
    RETURNING administrator_id‘
```

Listing 14: Rejestracja administratora, zwracane id do komunikatu

Przypisanie administratora do bibliotek

```
INSERT INTO project.zarzadzanie (biblioteka_id, administrator_id) VALUES ($1, $2)
```

Listing 15: Uzupelnienie tabeli zarzadzanie w celu przypisania administratora do bibliotek

Pobieranie danych do walidacji logowania

```
SELECT login FROM project.administrator WHERE login = $1 AND haslo = $2
```

Listing 16: Pobieranie danych do walidacji logowania

Zmiana danych uzytkownika

```
      UPDATE project.uzytkownik
      SET imie = $1, nazwisko = $2, data_urodzenia = $3
      WHERE uzytkownik_id = $4
```

Listing 17: Zmiana danych uzytkownika wprowadzana przez administratora na podstawie id uzytkownika

Pobieranie danych uzytkownika

```
      SELECT uzytkownik_id, imie, nazwisko, data_urodzenia, nr_karty_bibliotecznej
      FROM project.uzytkownik
```

Listing 18: Pobieranie danych uzytkownika

Usuwanie uzytkownika

```
      DELETE FROM project.uzytkownik
      WHERE uzytkownik_id = $1
```

Listing 19: Przeprowadzanie operacji usuwania uzytkownika przez administratora

Pobieranie kategorii

```
SELECT
    k.nazwa AS kategoria,
    COUNT(DISTINCT ks.ksiazka_id) AS ilosc_ksiazek,
    (SELECT CONCAT(a.imie, ' ', a.nazwisko)
     FROM project.ksiazka_autor ka
     JOIN project.autor a ON ka.autor_id = a.autor_id
     WHERE ka.ksiazka_id IN (
         SELECT ks.ksiazka_id
         FROM project.ksiazka ks
         WHERE ks.kategoria_id = k.kategoria_id
     )
     GROUP BY a.autor_id
```

```
     ORDER BY COUNT(ka.ksiazka_id) DESC
     LIMIT 1) AS najpopularniejszy_autor
    FROM
        project.kategoria k
    LEFT JOIN
        project.ksiazka ks ON k.kategoria_id = ks.kategoria_id
    GROUP BY
        k.kategoria_id
    ORDER BY
        k.nazwa;
```

Listing 20: Pobieranie danych kategorii do listy kategorii w aplikacji. Dodatkowo wprowadzono najpopularniejszego autora, oraz ilosc ksiazek w danej kategorii

Pobieranie autorów

```
SELECT
       a.autor_id,
       a.imie,
       a.nazwisko,
       a.narodowosc,
       COUNT(ka.ksiazka_id) AS liczba_ksiazek
     FROM
       project.autor a
     LEFT JOIN
       project.ksiazka_autor ka ON a.autor_id = ka.autor_id
     GROUP BY
       a.autor_id, a.imie, a.nazwisko, a.narodowosc
     ORDER BY
       liczba_ksiazek DESC;
```

Listing 21: Pobieranie danych do listy autorów wraz z iloscia ich ksiazek

Wprowadzanie uzytkownika

```
     INSERT INTO project.uzytkownik (imie, nazwisko, data_urodzenia, nr_karty_bibliotecznej
         )
     VALUES ($1, $2, $3, $4)
     RETURNING uzytkownik_id;
```

Listing 22: Wprowadzanie nowego uzytkownika przez administratora

Sprawdzanie istnienia ksiazki

```
     SELECT ksiazka_id
     FROM project.ksiazka
     WHERE tytul = $1 AND rok_wydania = $2 AND kategoria_id = $3;
```

Listing 23: Sprawdzanie istnienia ksiazki podczas wprowadzania nowej

Wprowadzanie ksiazki

```
     INSERT INTO project.ksiazka (tytul, kategoria_id, rok_wydania, kod)
     VALUES ($1, $2, $3, $4)
     RETURNING ksiazka_id;
```

Listing 24: Wprowadzanie nowej ksiazki przez administratora. Zwracanie id w celu komunikatu w aplikacji

Łączenie ksiazki z autorem

```
    INSERT INTO project.ksiazka_autor (ksiazka_id, autor_id)
     VALUES ($1, $2);
```

Listing 25: Po wporwadzeniu nowej ksiazki nastepuje polaczenie jej z autorem z pomoca tablicy asocjacyjnej ksiazka$_a utor$

Wprowadzanie egzemplarzy

```
    INSERT INTO project.egzemplarz (ksiazka_id, biblioteka_id, stan, data_nabycia)
    VALUES ($1, $2, 'Nowy', CURRENT_DATE)
```

Listing 26: Wprowadzanie nowych egzemplarzy po zamowieniu ich przez administratora

Sprawdzenie istnienia autora

```
    SELECT autor_id
    FROM autor
    WHERE imie = $1 AND nazwisko = $2 AND narodowosc = $3;
```

Listing 27: Sprawdzanie czy istnieje autor o podanych danych, przed wprowadzeniem nowego

Wprowadzenie autora

```
    INSERT INTO autor (imie, nazwisko, narodowosc)
    VALUES ($1, $2, $3)
    RETURNING autor_id;
```

Listing 28: Wprowadzenie nowego autora

Pojedyńcze, proste zapytania SELECT zostały pominięte, ponieważ aplikacja zawiera wiele prostych zapytań. Wszystkie najważniejsze zostały przedstawione powyżej