Projekt BoardBase

Sprawozdanie z Etapu 1 - Zaawansowane systemy baz danych

Krzysztof Dąbrowski 293101

Spis treści

Spis rycin

Spis tabel

Kod projektu

Kod projektu i sprawozdań jest dostępny na GitHub BoardBase

1 Zmiany w diagramie ERD

Poprawiłem diagram Entity Relationship Diagram (ERD) Rysunek 1, odnosząc się do uwag do *Case Study*. Wprowadziłem następujące zmiany:

- Usunąłem tablice słownikowe zastępując je bezpośrednimi atrybutami w tabelach,
- Usunąłem tabele *Collection* tworząc bezpośrednią relację N:N między *User* a *GameRelease*. Niestety straciłem w ten sposób informacje o nazwie kolekcji, ale nie wiem jak to rozwiązać,
- Dodałem tablicę *Location*, do monitorowania gdzie miały miejsce rozgrywki.
- Dodałem tablicę *Price*, pozwalającą śledzić ceny danego wydania. Tabela zawiera datę startu i końca obowiązywania ceny, żebyś śledzić jej zmiany w czasie,
- Dodałem tabelę *Award*, aby móc przechowywać informacje o nagrodach zdobytych przez gry planszowe.

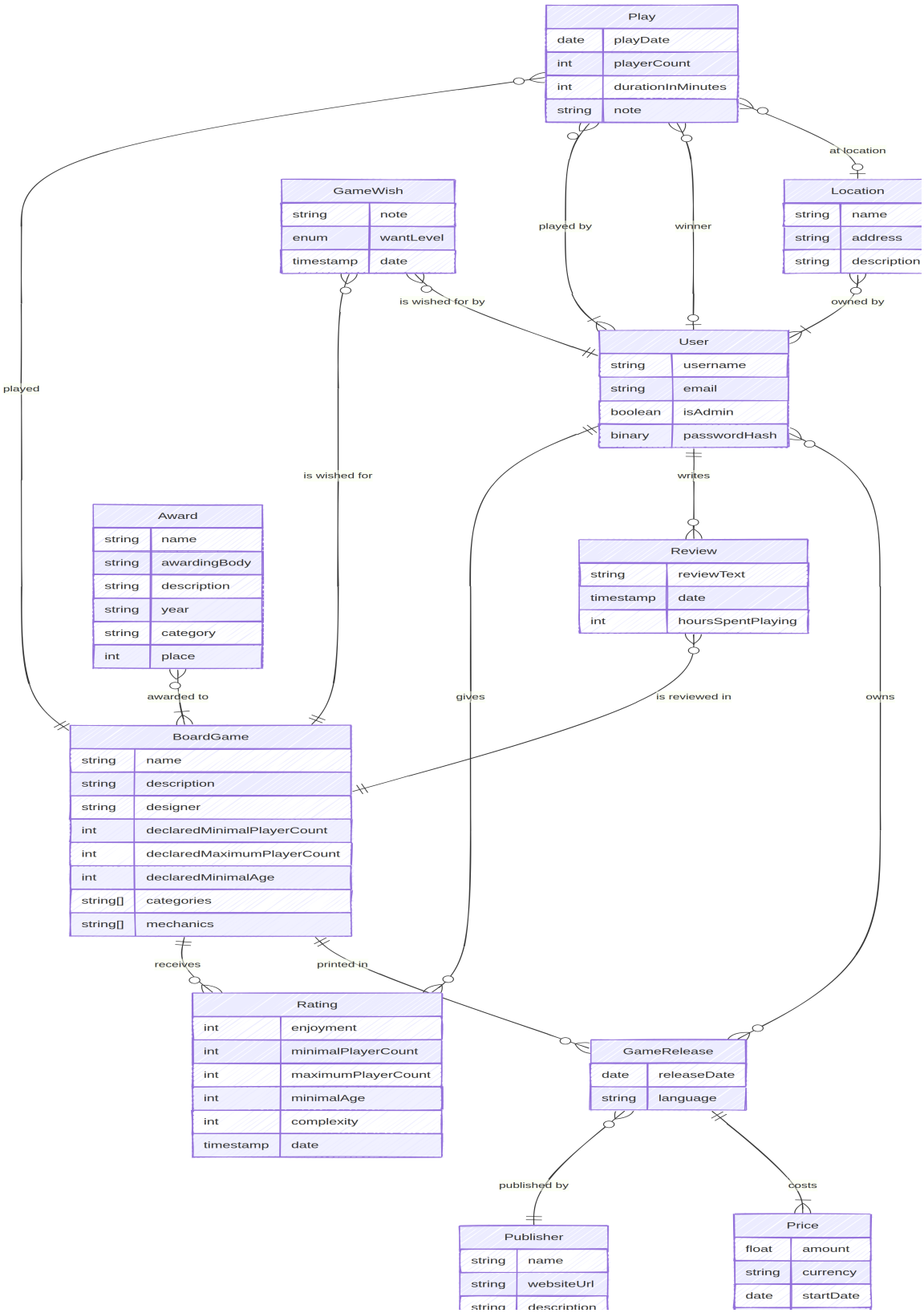
2 Utworzenie bazy danych

Wybrałem System zarządzania bazami danych (SZBD) PostgreSQL. Opis mojego wyboru i konfiguracji SZBD przedstawiłem w Case Study.

2.1 Schemat bazy danych

Sama baza danych `boardbase` została utworzona automatycznie na podstawie konfiguracji kontenera Docker.
Przed utworzeniem tabel utworzyłem schemat bazodanowy `mian` z autoryzacją dla domyślnego użytkownika `postgres` zgodnie z dokumentacją [1].

Po stworzeniu pierwszych tabl w schemacie `main` zauważyłem, że pisząc zapytania muszę nazwy razem ze schematem. Zastanowiło mnie to, ponieważ w przeszłości wykonywałem zapytania w innych bazach danych podając samą nazwę tabeli. Przeczytałem więcej dokumentacji [2] na temat działania schematów w PostgreSQL i znalazłem informację, że można skonfigurować domyślnie przeszukiwane schematy.



Rysunek 1: Zaktualizowany diagram ERD

Dowiedziałem się, że rekomendowaną praktyką jest dodanie schematu do domyślnego wyszukiwania, tak więc zrobiłem.

Po pewnym czasie zorientowałem się, że `SET search_path` działa tylko w biedzącej sesji. Ponieważ za każdym raziem chciałem korzystać ze schematu `main` zmieniłem `search_path` na poziomie bazy danych zgodnie z artykułem *How Do I Set/Change the Default Schema in PostgreSQL* [3].

2.2 Tabele

Na podstawie koncepcyjnego diagramu ERD Rysunek 1 stworzyłem odpowiednie tabele.
Do przechowania informacji o tym jak bardzo ktoś chce zagrać w daną grę ze swojej listy życzeń utworzyłem `ENUM` [4]. Do przechowania kategorii i mechanik używanych w grach użyłem tablic słownikowych.

Rozważałem czy utworzyć oddzielne tabele dla języków i walut. Zdecydowałem się odejść od Third Normal Form (Trzecia postać normalna) (3NF) w tym przypadku, ponieważ nazwy języków i walut praktycznie się nie zmieniają, a dodanie kolejnych tabel spowolniłoby odczytywanie danych.

Do przechowania danych tekstowych używam `VARCHAR` z ograniczeniem długości. Długości te wybrałem empirycznie na podstawie przykładowych wartości.
Żeby dbać o poprawność wprowadzanych danych użyłem instrukcji `CONSTRAINT`, `UNIQUE` oraz kontroli "nullowości".
Dla zależnych tabeli rozważyłem co powinno się stać, gdy zależne dane zostaną usunięte. Ustawiłem odpowiednie zachowania instrukcjami `ON DELETE`.
Dla tabeli realizujących relacje N:N użyłem kluczy złożonych kierując się dyskusją na temat ze stackoverflow [5].

Po utworzeniu tabel wygenerowałem schemat logiczny bazy danych Rysunek 2, za pomocą narzędzia pgAdmin [6].

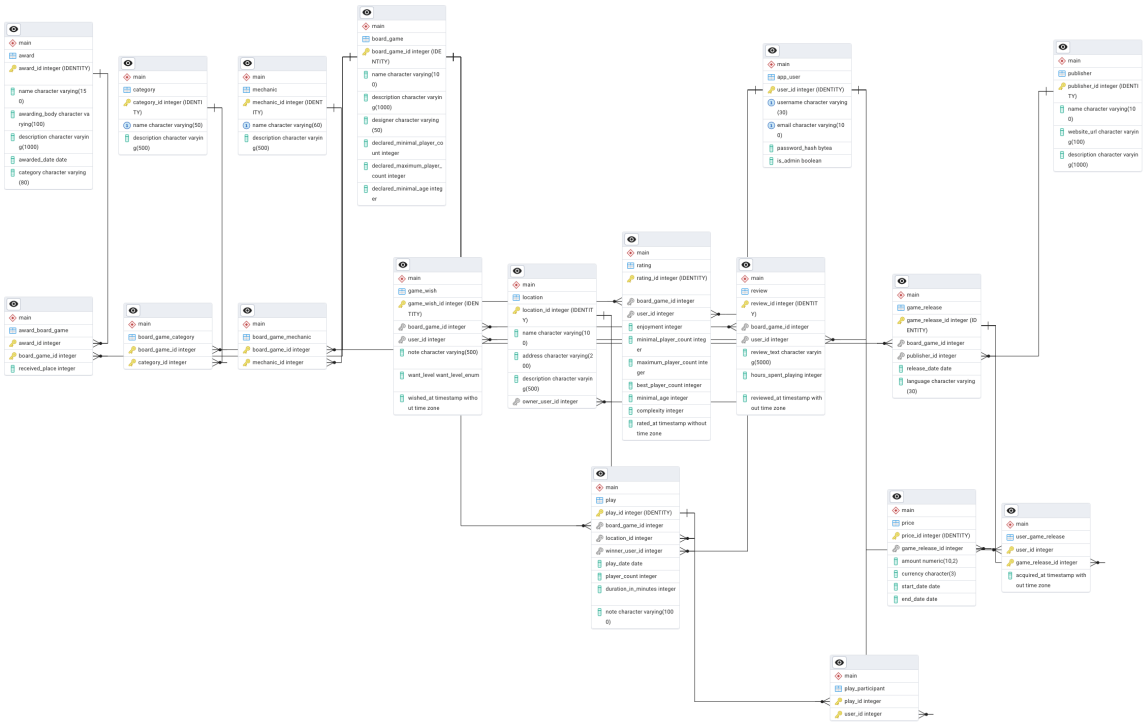
3 Wypełnie bazy danymi

Do wygenerowania przykładowych danych użyłem Large Language Model (LLM) Claude Sonet 4.5.

3.1 Generowanie przykładowych danych

Najpierw przygotowałem plik z informacjami o strukturze bazy danych za pomocą narzędzia GitHub Copilot [7] zintegrowanego z VS Code. Prompt którego użyłem:

> I'll need to generate real looking data for my database. Your task is to crate a LLM readable file describing the database scheema so the LLM can use it to create the sample data. See the #file:schema files to learn about the database



Rysunek 2: Logiczny schemat bazy danych

Na tej podstawie LLM wygenerował plik SCHEMA_DESCRIPTION.md, który dostosowałem do moich potrzeb.
Następnie na stronie https://claude.ai/ wygenerowałem pliki z przykładowymi danymi. Tak wyglądała moja konwersacja z agentem Artificial Intelligence (AI).

3.2 Wczytywanie danych

Przy próbie wczytania danych napotkałem problemy ze wstawieniem danych do tabeli `award`. Sprawdziłem, że ta tabela nie została utworzona. Okazało się, że zrobiłem drobne błędy składniowe i nie zauważyłem błędu podczas tworzenia tabel. Poprawiłem błędy i ponownie utworzyłem tabele.

Napotkałem kolejny problem z wczytaniem haszy haseł. Były one zaczytywane z notacji heksadecymalnej, która ma dość sztywny format. LLM wygenerował niepoprawne wartości. Wygenerowałem haszhe ręcznie używając narzędzia online [8].
Wiem, że hash sha256 nie jest najbezpieczniejszym rozwiązaniem. Mimo tego, do projektu zdecydowałem się go użyć ze względu na szybkość działania.

Przy kolejnej próbie wczytania danych napotkałem problem z kluczami głównymi w tabeli `user`. Ponieważ użyłem automatycznej sekwencji do generowania kluczy, to przy usunięciu i ponownym wczytaniu danych zostały one ponumerowane kolejnymi wartościami sekwencji,

a dane wygenerowane przez LLM zakładały numerację od 1. Rozwiązałem ten problem resetując sekwencję razem z usuwaniem danych zgodnie z odpowiedzią na forum [9].

Po tych poprawach udało mi się wczytać całość wygenerowanych danych.

4 Użytkownicy i uprawnienia

Do zarządzania uprawnieniami użyłem roli [10].
Zacząłem od utworzenia roli grupowych `admins`, `players` i `guests`. Przypisałem im uprawnienia do odpowiednich typów zapytań w zależności od tabeli poleceniem `GRANT` [11].

W kolejnym kroku utworzyłem użytkowników, po jednym dla każdej roli grupowej.

4.1 Testy uprawnień

Zmieniłem rolę na `guest` i spróbowałem wstawić nową grę.

```
SET role guest;
SELECT current_user, session_user;

INSERT INTO board_game (name, description, designer,
 ↪  declared_minimal_player_count, declared_maximum_player_count,
 ↪  declared_minimal_age) VALUES
('Catan', 'Trade, build, and settle the island of Catan.', 'Klaus Teuber',
 ↪  3, 4, 10);
```

W odpowiedzi otrzymałem błąd `permission denied for table board_game`.

Przetestowałem też czy użytkownik `guest` może odczytać dane z tabeli `board_game`. Zapytanie wykonało się poprawnie.

Następnie przetestowałem czy użytkownik `casual_gamer` może dodać ocenę do gry.

```
SET role casual_gamer;
SELECT current_user, session_user;

INSERT INTO rating (user_id, board_game_id, enjoyment) VALUES
((SELECT user_id FROM app_user WHERE username = 'casual_gamer'),
 (SELECT game_id FROM board_game WHERE name = 'Catan'),
 8);
```

Zapytanie wykonało się poprawnie.

Na koniec przetestowałem czy użytkownik `god` może usunąć dopier co dodaną ocenę.

```sql
SET role god;
SELECT current_user, session_user;

DELETE FROM rating WHERE rating_id = 12;
```

Zapytanie wykonało się poprawnie.

Po tych testach spodziewam się, że uprawnienia zostały poprawnie skonfigurowane.

5 Przykładowe zapytania

Wykonałem kilka przykładowych zapytań, aby przetestować działanie bazy danych.

5.1 Użytkownicy z ponad przeciętną liczbą gier

Zastanawiałem się ilu jest użytkowników, którzy mają ponad przeciętną liczbę gier. Napisałem zapytanie, które znajduje takich użytkowników i zwraca ich statystki.

```sql
SELECT
    u.username,
    COUNT(DISTINCT ugr.game_release_id) AS game_count,
    (
        SELECT ROUND(AVG(game_count), 2)
        FROM (
            SELECT COUNT(*) AS game_count
            FROM main.user_game_release
            GROUP BY user_id
        ) AS user_collections
    ) AS avg_games_per_user,
    COUNT(DISTINCT r.rating_id) AS ratings_given,
    COUNT(DISTINCT pp.play_id) AS sessions_played,
    MIN(ugr.acquired_at) AS first_game_acquired
FROM main.app_user u
INNER JOIN main.user_game_release ugr ON u.user_id = ugr.user_id
LEFT JOIN main.rating r ON u.user_id = r.user_id
LEFT JOIN main.play_participant pp ON u.user_id = pp.user_id
GROUP BY u.user_id, u.username, u.email
HAVING COUNT(DISTINCT ugr.game_release_id) > (
    SELECT AVG(game_count)
    FROM (
        SELECT COUNT(*) AS game_count
        FROM main.user_game_release
        GROUP BY user_id
```

```
    ) AS collections
)
ORDER BY game_count DESC;
```

Tabela 1: Użytkownicy z ponad przeciętną liczbą gier

game_name	designer	main_category	avg_rating	rating_count	category_avg_rating
Gloomhaven	Isaac Childres	Adventure	9.67	3	9.29
Gloomhaven	Isaac Childres	Cooperative	9.67	3	9.13
Gloomhaven	Isaac Childres	Strategy	9.67	3	8.31
Brass: Birmingham	Martin Wallace	Strategy	9.33	3	8.31
Brass: Birmingham	Martin Wallace	Economic	9.33	3	8.24
Scythe	Jamey Stegmaier	Economic	9.00	4	8.24
Scythe	Jamey Stegmaier	Strategy	9.00	4	8.31
Wingspan	Elizabeth Hargrave	Family	8.80	5	8.03
Wingspan	Elizabeth Hargrave	Strategy	8.80	5	8.31
Pandemic	Matt Leacock	Family	8.80	5	8.03
Pandemic	Matt Leacock	Strategy	8.80	5	8.31
Terraforming Mars	Jacob Fryxelius	Strategy	8.75	4	8.31
Terraforming Mars	Jacob Fryxelius	Economic	8.75	4	8.24

5.2 Najlepsze gry w danej kategorii

Chciałem znaleźć wyjątkowo dobre gry. Pomyślałem, że sprawiedliwie będzie porównywać je w ramach danej kategorii. Napisałem zapytanie, które dla każdej kategorii znajduje gry z ponad przeciętną oceną względem swojej kategorii.

```
SELECT
    bg.name AS game_name,
    bg.designer,
    c.name AS main_category,
    ROUND(AVG(r.enjoyment), 2) AS avg_rating,
    COUNT(r.rating_id) AS rating_count,
    (
        SELECT ROUND(AVG(r2.enjoyment), 2)
        FROM main.rating r2
        INNER JOIN main.board_game_category bgc2
            ON r2.board_game_id = bgc2.board_game_id
```

```sql
        WHERE bgc2.category_id = c.category_id
    ) AS category_avg_rating
FROM main.board_game bg
INNER JOIN main.rating r ON bg.board_game_id = r.board_game_id
INNER JOIN main.board_game_category bgc ON bg.board_game_id =
 ↪ bgc.board_game_id
INNER JOIN main.category c ON bgc.category_id = c.category_id
GROUP BY
    bg.board_game_id,
    bg.name,
    bg.designer,
    c.category_id,
    c.name
HAVING AVG(r.enjoyment) > (
    SELECT AVG(r3.enjoyment)
    FROM main.rating r3
    INNER JOIN main.board_game_category bgc3
        ON r3.board_game_id = bgc3.board_game_id
    WHERE bgc3.category_id = c.category_id
)
ORDER BY avg_rating DESC, rating_count DESC
```

Tabela 2: Gry z oceną powyżej średniej w swojej kategorii

username	game_count	avg_games_per_user	ratings_given	sessions_played	first_game_acquired
board_master_42	9	4.50	9	17	2005-08-10 12:00:00
meeple_collector	6	4.50	6	17	2010-05-20 17:45:00
strategy_queen	6	4.50	8	18	2008-01-20 14:15:00
euro_enthusiast	5	4.50	6	16	2011-03-15 18:00:00

6 Perspektywy

Utworzyłem 5 widoków dostępnych dla różnych ról użytkowników:

1. **game_catalog** – Publiczny katalog wszystkich gier planszowych z agregowanymi informacjami: średnimi ocenami, liczbą odgrywek, średnim czasem trwania oraz połączonymi listami kategorii i mechanik. Dostępny dla wszystkich użytkowników.
2. **game_prices_current** – Widok aktualnych cen wydań gier. Dostępny dla wszystkich.
3. **user_game_collection** – Filtruje dane po nazwie aktualnie zalogowanego użytkownika i wyświetla jego kolekcję gier.

4. **user_play_history** – Filtruje dane po nazwie aktualnie zalogowanego użytkownika i pokazuje historię jego rozgrywek.
5. **games_missing_metadata** – Widok administratorski wyświetlający gry bez przypisanych kategorii lub mechanik.

Tak przykładowo wygląda użycie widoku `user_game_collection`, którego wynik przedstawia Tabela 3.:

```
SET ROLE casual_gamer;

SELECT
    game_name,
    publisher_name,
    language,
    current_price,
    currency,
    years_owned,
    times_played
FROM main.user_game_collection;
```

Tabela 3: Kolekcja gier użytkownika casual_gamer

game_name	publisher_name	language	current_price	currency	years_owned	times_played
Catan	KOSMOS	English	42.00	USD	9	0
Ticket to Ride	Days of Wonder	English	44.99	USD	10	4
Pandemic	Z-Man Games	English	39.99	USD	11	4
Carcassonne	Z-Man Games	German	29.99	EUR	15	2

7 Indeksy

Żeby poprawnie przetestować działanie indeksów wygenerowałem znaczną ilość danych testowych za pomocą narzędzia mockaroo [12].
Wygenerowałem po 1000 rekordów dla tabeli `board_game` i `app_user`.

Wybierając typy indeksów kierowałem się ich opisami w dokumentacji [13].

Pomiędzy zapytaniami usuwałem i tworzyłem na nowo kontener z bazą danych, żeby pozbyć się cashowania danych.

7.1 Wyszukiwanie użytkownika po nazwie

Próbowałem dodać indeks do kolumny `username` w tabeli `app_user`. Podczas testowania zorientowałem się jednak, że nawet bez tego indeksu zapytania używają indeksu o nazwie `app_user_username_key`.

Z dokumentacji [14] dowiedziałem się, że `UNIQUE` tworzy indeks B-tree automatycznie. Chciałem jednak przetestować jego działanie, więc usunąłem ograniczenie `UNIQUE` z kolumny `username` i utworzyłem indeks ręcznie.

Wybrałem indeks B-tree, ponieważ dobrze nadaje się do indeksowania krótkich tekstów i pozwala szukać nawet tylko po początkowej części nazwy użytkownika, w przeciwieństwie do indeksu Hash.

Bez indeksu zapytanie znalezienia konkretnego użytkownika wykonywało się metodą `Seq Scan on app_user` i trwało

```
Planning Time: 1.994 ms
Execution Time: 0.980 ms
```

Z indeksem zapytanie wykonywało się metodą `Index Scan using username_index on app_user` i trwało

```
Planning Time: 1.588 ms
Execution Time: 0.094 ms
```

Widać wyraźną poprawę czasu wykonania zapytania przy zastosowaniu indeksu.

Z ciekawości przetestowałem też indeks typu Hash. Zapytanie wykonywało się metodą `Index Scan using idx_app_user_username on app_user` i trwało

```
Planning Time: 0.753 ms
Execution Time: 0.067 ms
```

Jest to wynik lekko szybszy niż B-tree. Wymaga on jednak podania dokładnej nazwy użytkownika. Uważam, że w tym przypadku lepiej pozostać przy indeksie B-tree.

7.2 Indeksy na tabeli gier planszowych

Na tabeli `board_game` utworzyłem 4 indeksy:

- idx_board_game_name — indeks B-tree na kolumnie `name` przyspieszający wyszukiwanie po nazwie oraz sortowanie.
- idx_board_game_player_count — indeks B-tree na parach kolumn (`declared_minimal_player_count`, `declared_maximum_player_count`) zoptymalizowany pod zapytania typu "czy N graczy mieści się w zakresie", używany przez warunki typu BETWEEN.
- idx_board_game_min_age — indeks B-tree na kolumnie `declared_minimal_age` przyspieszający filtrowanie po minimalnym wieku.
- idx_board_game_description — indeks GIN przewidziany do wsparcia zapytań pełnotekstowych

Zmierzyłem czas wykonania zapytań wyszukiwania gry pasującej dla określonej liczby graczy.

```
EXPLAIN (ANALYZE, BUFFERS, FORMAT TEXT)
SELECT board_game_id, name,
       declared_minimal_player_count,
       declared_maximum_player_count
FROM board_game
WHERE 4 BETWEEN declared_minimal_player_count AND
 ↪  declared_maximum_player_count;
```

Przed zastosowaniem indeksu

```
Planning Time: 2.095 ms
Execution Time: 9.450 ms
```

Po zastosowaniu indeksu

```
Planning Time: 1.445 ms
Execution Time: 1.514 ms
```

Indeks zdecydowanie poprawił czas wykonania tego typu zapytań.

Przy wyborze indeksów kierowałem się przewidywaniem jakie zlatania będą najczęściej wykonywane. Nie wybierałem kolumn z małą liczbą unikalnych wartości, ponieważ indeksowanie ich prawdopodobnie nie przyniosłoby znaczącego wzrostu wydajności.

Źródła

1. PostgreSQL: Documentation: 18: CREATE SCHEMA. https://www.postgresql.org/docs/current/sql-createschema.html. Accessed 19 paź 2025

2. PostgreSQL: Documentation: 18: 5.10. Schemas. https://www.postgresql.org/docs/current/ddl-schemas.html. Accessed 19 paź 2025

3. Malik TS How Do I Set/Change the Default Schema in PostgreSQL - CommandPrompt Inc. https://www.commandprompt.com/education/how-do-i-setchange-the-default-schema-in-postgresql/. Accessed 19 paź 2025

4. PostgreSQL: Documentation: 18: 8.7. Enumerated Types. https://www.postgresql.org/docs/current/datatype-enum.html. Accessed 19 paź 2025

5. sql - Composite key in associative tables - Stack Overflow. https://stackoverflow.com/questions/29585451/composite-key-in-associative-tables. Accessed 19 paź 2025

6. pgAdmin - PostgreSQL Tools. https://www.pgadmin.org/. Accessed 19 paź 2025

7. GitHub Copilot · Your AI pair programmer. https://github.com/features/copilot. Accessed 19 paź 2025

8. emn178 SHA256 - Online Tools. https://emn178.github.io/online-tools/sha256.html. Accessed 19 paź 2025

9. sql - Reset auto increment counter in postgres - Stack Overflow. https://stackoverflow.com/questions/5342440/reset-auto-increment-counter-in-postgres. Accessed 19 paź 2025

10. PostgreSQL: Documentation: 18: Chapter 21. Database Roles. https://www.postgresql.org/docs/current/user-manag.html. Accessed 19 paź 2025

11. PostgreSQL: Documentation: 18: GRANT. https://www.postgresql.org/docs/current/sql-grant.html. Accessed 19 paź 2025

12. Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel. https://www.mockaroo.com/. Accessed 19 paź 2025

13. PostgreSQL: Documentation: 18: 11.2. Index Types. https://www.postgresql.org/docs/current/indexes-types.html. Accessed 19 paź 2025

14. PostgreSQL: Documentation: 18: 5.5. Constraints. https://www.postgresql.org/docs/current/ddl-constraints.html#DDL-CONSTRAINTS-UNIQUE-CONSTRAINTS. Accessed 19 paź 2025

Zastosowane skróty

3NF Third Normal Form (Trzecia postać normalna)
AI Artificial Intelligence
ERD Entity Relationship Diagram
LLM Large Language Model
SZBD System zarządzania bazami danych