Projekt BoardBase

Sprawozdanie z Etapu 2 - Zaawansowane systemy baz danych

Krzysztof Dąbrowski (293101)
GR6 (Poniedziałek 8:15)

Spis treści

Spis rycin

Spis tabel

Spis wykazów

Kod projektu

Kod projektu i sprawozdań jest dostępny na GitHub BoardBase

1 Funkcje i procedury dodawania elementów

1.1 Funkcja

Przeczytałem, że liczenie arytmetycznej średniej ocen jest mało miarodajne dla małej liczby ocen, a użycie średniej bayesowskiej jest lepszym rozwiązaniem [1]. Napisałem User Defined Function (UDF) liczącą w ten sposób średnią ocenę gry.

Dla gry z 5 pozytywnymi ocenami średnia arytmetyczna wychodziła 8.8, a wynik mojej UDF `fair_game_rating` wyniósł 8.48(6).

1.2 Procedury dodawania elementów

Utworzyłem kilka procedur do dodawania nowych rekordów do bazy danych w pliku `create-records-procedures.sql`.

- `add_user` dodaje nowego użytkownika,
- `add_location` dodaje nowe miejsce do grania,
- `add_rating` dodaje nową ocenę gry,
- `add_review` dodaje nową recenzję,
- `add_game_wish` dodaje nową grę do listy życzeń.

Procedura `add_user` automatycznie wylicza hash hasła i przycina białe znaki z nazwy użytkownika i adresu email.
Pozostałe procedury ułatwiają ustawiają automatycznie id użytkownika na pasujące do aktualnie zalogowanego użytkownika do bazy za pomocą `WITH` z Common Table Expression (CTE).

1.3 Dokumentacja działania

Przy testowaniu procedur miałem kłopot z deklarowaniem zmiennych. Jestem przyzwyczajony do deklarowania zmiennych w ramach sesji Structured Query Language (SQL) za pomocą zwykłego `DECLARE` @nazwa zmiennej.
Przeczytałem, że w PostgreSQL trzeba deklarować zmienne w blokach [2]. Do tego dowiedziałem się, że w blokach nie mogę używać `SELECT` do wyświetlania wartości. Zamiast tego użyłem `RAISE NOTICE` [3].

Na początku nie działało mi tworzenie nowego użytkownika procedurą `add_user`.
Przypominałem sobie, że na poprzednim etapie ustawiłem uprawnienia do tworzenia użytkowników i tylko administratorzy mogą to robić. Przełączyłem się na konto administratora i procedura zadziałała (Wykaz 1).

Wykaz 1 Wynik procedury `add_user`

```
NOTICE: Created user: 1015
```

Zmieniłem użytkownika na *casual_gamer* i wykonałem kolejne procedury z mojego testowego pliku `try-simple-insert-procedures.sql`.
Zgodnie z oczekiwaniami otrzymałem wyniki Wykaz 2.

Wykaz 2 Wyniki pozostałych procedur dodawania rekordów

```
NOTICE: Created location: 11
NOTICE: Created rating: 61
NOTICE: Created review: 19
NOTICE: Created game wish: 16
```

Utworzenie rekordów sprawdziłem też zapytaniami `SELECT`.

2 Złożona procedura

Czasem zdarza się, że użytkownik posiada na kilka kont w systemie na przykład zalogowane mailem i założone za pośrednictwem konta dużego serwisu takiego jak na przykład Google.
W takiej sytuacji, której sam kiedyś doświadczyłem, użytkownik po jakimś czasie może chcieć połączyć swoje konta w jedno.

W tym celu przygotowałem procedurę `merge_user_accounts`, która pozwala zmigrować dane z jednego konta do drugiego.

Wszystkie aktualizacje danych są wykonywane w ramach transakcji. Dzięki temu baza przejdzie ze stanu spójnego do spójnego lub cofnie wszystkie zmiany.

Procedura sprawdza czy użytkownicy istnieją i czy są różni. Jeśli tak, to wykonuje następujące kroki:

1. Migruje oceny gry,
2. Migruje recenzje,
3. Migruje listy życzeń,
4. Migruje kolekcję gier,
5. Migruje lokalizacje,
6. Migruje dziennik rozegranych gier,
7. Usuwa stare konto.

Na każdym etapie sprawdzane jest czy użytkownik nie ma już danych wartości na docelowym koncie. Jeśli to możliwe, to dane są aktualizowane, a w przeciwnym razie dane starego konta są usuwane.

2.1 Dokumentacja działania

Do przetestowania transakcji w procedurze zasymulowałem błąd pod koniec dodając `RAISE EXCEPTION`, a następnie ją wywołałem i sprawdziłem gry użytkownika (Wykaz 3).

Wykaz 3 Wywołanie procedury `merge_user_accounts` z dodanym błedem

```
CALL merge_user_accounts(9, 10);
SELECT * FROM user_game_release WHERE user_id = 9;
```

Po wynikach (Wykaz 4) zaobserwowałem, że wystąpił błąd pod koniec działania procedury.

Po wynikach `SELECT` (Tabela 1) widać, że dane nie zostały zmigrowane, ponieważ użytkownik nadal ma gry, Mechanizm transakcji cofnął zmiany.

Tabela 1: Gry użytkownika źródłowego po nieudanym wykonaniu procedury

user_id	game_release_id	acquired_at
9	25	2016-02-05 14:00:00
9	28	2015-03-20 15:45:00
9	33	2019-11-01 17:15:00

Gdy usunąłem ręcznie rzucenie wyjątku, dane dane użytkownika zostały poprawnie przeniesione (Wykaz 5), a zapytanie `SELECT` (Tabela 2) nie zwróciło żadnych wyników.

Wykaz 4 Logi nieudanego łączenia kont

```
NOTICE: Starting merge: card_shark_99 (ID: 9) → family_fun_time (ID: 10)
NOTICE: Deleted conflicting ratings where target user already has ratings
NOTICE: Migrated ratings from source to target user
NOTICE: Migrated reviews from source to target user
NOTICE: Deleted conflicting game wishes where target user already has wishes
NOTICE: Migrated game wishes from source to target user
NOTICE: Updated acquired dates where source user acquired games earlier
NOTICE: Deleted conflicting game releases from source collection
NOTICE: Migrated unique game releases to target user collection
NOTICE: Transferred location ownership from source to target user
NOTICE: Updated play sessions where source user was winner
NOTICE: Deleted conflicting play participations where target user already participated
Test exception
CONTEXT: PL/pgSQL function merge_user_accounts(integer,integer) line 123 at RAISE
```

Tabela 2: Gry użytkownika źródłowego po poprawnym wykonaniu procedury

user_id	game_release_id	acquired_at

3 Wyzwalacze

Chciałem, żeby gdy użytkownik wejdzie w posiadanie gry została ona automatycznie usunięta z jego listy życzeń. W tym celu przygotowałem wyzwalacz, który to automatyzuje.

Mój pomysł na logikę biznesową zakłada, że zapisują informację o rozgrywce (`play`) można podać większą liczbę uczestników (`player_count`) niż powiązanych graczy, ponieważ nie każdy uczestnik musi mieć konto w systemie. Może to jednak prowadzić do błędnej sytuacji podania mniejszej liczby `player_count` niż powiązanych graczy.
Przygotowałem wyzwalacz, który przy dodaniu relacji na większą liczbę graczy niż `player_count` aktualizuje `player_count` na większą wartość oraz drugi, który pilnuje, żeby `player_count` nie był ustawiony na mniejszą wartość niż liczba faktycznych graczy.

W PosgreSQL wyzwalacz musi wywoływać funkcję lub procedurę [4], więc do każdego wyzwalacza stworzyłem też odpowiadającą procedurę.

Wykaz 5 Udane połączenie kont

```
Started executing query at Line 4
NOTICE: Starting merge: card_shark_99 (ID: 9) → family_fun_time (ID: 10)
NOTICE: Deleted conflicting ratings where target user already has
 ↪  ratings
NOTICE: Migrated ratings from source to target user
NOTICE: Migrated reviews from source to target user
NOTICE: Deleted conflicting game wishes where target user already has
 ↪  wishes
NOTICE: Migrated game wishes from source to target user
NOTICE: Updated acquired dates where source user acquired games earlier
NOTICE: Deleted conflicting game releases from source collection
NOTICE: Migrated unique game releases to target user collection
NOTICE: Transferred location ownership from source to target user
NOTICE: Updated play sessions where source user was winner
NOTICE: Deleted conflicting play participations where target user
 ↪  already participated
NOTICE: Migrated play participations from source to target user
```

Kod wyzwalaczy umieściłem w pliku `triggers.sql`.

3.1 Dokumentacja działania

Dla użytkownika `casual_gamer` dodałem grę *Pandemic* (Wykaz 6) do jego listy życzeń Tabela 3.

Wykaz 6 Dodanie gry Pandemic do listy życzeń

```
CALL add_game_wish(4, NULL);
```

Tabela 3: Życzenia użytkownika `casual_gamer` po dodaniu gry *Pandemic* do listy życzeń

username	name	wished_at
casual_gamer	Pandemic	2025-11-07 10:39:01.453507
casual_gamer	Splendor	2024-09-20 14:00:00
casual_gamer	Gloomhaven	2024-08-20 16:45:00

Następnie dodałem egzemplarz gry *Pandemic* do kolekcji użytkownika. Po dodaniu gry do kolekcji automatycznie zniknęła ona z jego listy życzeń Tabela 4.

Tabela 4: Życzenia użytkownika `casual_gamer` po dodaniu gry *Pandemic* do kolekcji

username	name	wished_at
casual_gamer	Splendor	2024-09-20 14:00:00
casual_gamer	Gloomhaven	2024-08-20 16:45:00

Kod użyty do testów umieściłem w pliku `test-wishlist-trigger.sql`.

4 Inne elementy ograniczenia dostępu

Do tej pory przygotowałem już **18** elementów ograniczenia dostępu z **10 sugerowanych** w zakresie zadania.

- 5 procedur dodawania danych
- 1 złożona procedura
- 3 wyzwalacze
- 4 widoki użytkownika
- 1 widok administratora
- 4 widoki publiczne

Z tego powodu ograniczę się do zastanowienia, co jeszcze byłoby warto dodać w produkcyjnej aplikacji.

- widok statystyk użytkownika
- widok historii ceny danej edycji gry
- widok personalizowanych rekomendacji gier
- wyzwalacze pilnujące czy zwycięzca rozgrywki jest jej uczestnikiem

Sporą część zachowania spójności zapewniają już instrukcje kaskadowego usuwania powiązanych rekordów, które skonfigurowałem tworząc tabele.

5 Automatyzacja zadania

5.1 Wybór scheduler'a

Żeby uruchamiać automatycznie kod `SQL` w bazie zdecydowałem się użyć rozszerzenia pg_cron, o którym słyszałem w filmie *I replaced my entire tech stack with Postgres…* [5].

Rozważałem użycie scheduler'a pgAgent, ale po przeczytaniu artykułu porównującego te rozwiązania [6] i tabeli różnić [7], uznałem, że pg_cron bardziej mi pasuje, ponieważ

jest lżejszym rozwiązaniem i nie wymaga oddzielnego demona. Gdyby zadanie wymagało zaawansowanej logiki uruchamiania zadań w czasie i zależności między zadaniami pewnie wybrałbympgAgent'a.

5.2 Instalacja

Żeby zainstalować rozszerzenie nie mogłem już korzystać z domyślnego obrazu Docker [8] i musiałem skonfigurować własny obraz, który bazuje na tym bazowy, na którym zainstaluję rozszerzenie. Znalazłem obraz z od razu zainstalowanym rozszerzeniem pg_cron [9], ale używa on starej wersji Postgres, więc wolałem stworzyć swój.

Przygotowałem własny obraz z zainstalowanym rozszerzeniem pg_cron w pliku `Postgres.Dockerfile`, zmieniłem plik `docker-compose.yml`, żeby używał tego obrazu i skonfigurowałem System zarządzania bazami danych (SZBD) w pliku `postgresql.conf`.

Po skonfigurowaniu obrazu Docker okazało się, że nie uruchamia się kontener. Pojawiał się błąd, że w obrazach w wersji 18+ volumen musi być zamontowany poziom wyżej w kontenerze. Zdziwiło mnie to, ponieważ wcześniej używałem po prostu obrazu `18.0`, a ten błąd nie występował.
Przywróciłem konfigurację Docker Compose do używania bezpośrednio bazowego obrazu, ale błąd dalej występował. Poprawiłem więc ścieżkę volumenu, co rozwiązało problem, choć dalej zastanawia mnie, dlaczego wcześniej nie miałem tego problemu.
Przypuszczam, że został wgrany nowy obraz pod tag `18.0`, a przygotowanie mojego obrazu sprowokowało `docker pull`.

5.3 Automatyczne zadanie

Pomyślałem, że użytkownicy często chcieliby przeglądać statystyki gier. Niektóre ze statystyk wymagają agregacji. Ponieważ wyliczanie niektórych agregacji jest czasochłonne, a użytkownicy nie oczekują raczej w pełni aktualnych statystyk (pewnie wyniki z ostatniego dnia będą dobre), to utworzyłem zmaterializowany widok przechowujący te dane.
Za pomocą automatycznego zadania pg_cron, skonfigurowanego w pliku `automation.sql`, aktualizuję ten widok codziennie o 3 w nocy.
Skonfigurowane zadanie widać w wyniku poniższego zapytania (Wykaz 7) w Tabela 5.

Wykaz 7 Zadania pg_cron

```
1 SELECT jobid, schedule, command
2 FROM cron.job
```

Tabela 5: Automatyczne zadania

jobid	schedule	command
1	0 3 * * *	REFRESH MATERIALIZED VIEW main.game_popularity_stats

SQL widoku zapisałem w pliku `game-stats-materialized-view.sql`. Widok wylicza między innymi bayesowską średnią ocenę gry, którą zaimplementowałem wykorzystując moją UDF `fair_game_rating`, liczbę gier rozegranych w ostatnich 90 dniach, czy też customowy ranking popularności, który wyliczam heurystycznie na podstawie kilku innych parametrów i inne statystyki.

6 Kopia zapasowa

6.1 Wybór podejścia

Zastanawiałem się nad wyborem między trzema głownymi podejściami do backupu [10]:

1. SQL dump,
2. backup plików bazy danych,
3. backup plików i Write Ahead Log (WAL) z odtwarzaniem bazy point in time z WAL.

Uznałem, że podejście bazujące na WAL jest zbyt złożone na moje potrzeby, gdy zapoznałem się z mechanizmem jego działania [11].
Podejście drugie wymaga zatrzymania SZBD, co sprawia, że w produkcyjnych środowiskach jest mało praktyczne. Dokumentacja sugeruje, żeby raczej używać podejścia bazującego na SQL dump [12], więc to na nie się zdecydowałem.

6.2 Wykonanie kopii zapasowej

Zgodnie z zaleceniami z dokumentacji [13], użyłem narzędzia `pg_dump` do zrobienia backupu.
Za pierwszym razem narzędzie `pg_dump` nie zadziałało, ponieważ okazało się, że domyślny pakiet homebrew instaluje wersję 14, a mój SZBD jest w wersji 18. Na szczęście przyczyna błędu była dobrze opisana w komunikacie i wystarczyło zainstalować odpowiednią wersję. Zrobiłem to poleceniem `brew install postgresql@18`.

Ponieważ przeczytałem, że domyślnie `pg_dump` loguje się do bazy taką samą nazwą użytkownika [13], jak zalogowane konto, to nadpisałem nazwę użytkownika i kilka innych podstawowych parametrów w wywołaniu Wykaz 8.

Wykaz 8 Wykonanie kopii zapasowej pg_dump

```
pg_dump --dbname=boardbase --username=postgres --host=localhost
  ↪  --file=2025-11-10_boardbase_backup.sql
```

Zgodnie z moimi oczekiwaniami powstały plik zawiera kod SQL do odtworzenia schematu bazy, danych, a nawet utworzonego rozszerzenia. Zawiera nawet konfigurację mojego cron joba.

6.3 Weryfikacja kopii zapasowej

Żeby sprawdzić poprawność kopii zapasowej postanowiłem odtworzyć z niej bazę danych.

Do kontenera z SZBD zamontowałem nowy volumen.
Przed wgraniem kopii zapasowej utworzyłem użytkowników, ponieważ skrypt kopii nie tworzy użytkowników [13]. W moim skrypcie `1_create-groups` tworzę grupy i nadaję im role. Ponieważ nie powstał jeszcze schemat `main`, to nadanie do niego ról nie zadziałało. Nie jest to problem, ponieważ przypisanie uprawnień jest zawarte w kopii zapasowej.
Po utworzeniu grup, mogłem stworzyć użytkowników i przypisać ich do grup skryptem `2_craete-users`.

Wczytałem kopię zapasową i spróbowałem uruchomić kilka zapytań.
Okazało się, że **zapytania, w których nie podawałem jawnie nazwy schematu, nie działały**.
Sprawdziłem, i okazało się, że w pliku backupu **nie ma mojej instrukcji zmiany domyślnego `search_path`**, żeby zawierał schemat `main`. Wykonałem ją ręcznie. Wtedy zapytania działały już poprawnie.

7 Inny SZBD

Wybrałem SZBD MS SQL Server. Używałem go pośrednio w pracy w aplikacjach backendowych przez Object Relational Mapper (ORM) i chciałbym lepiej poznać tą technologię.

Tak jak w przypadku Postgres, zdecydowałem się na uruchomienie MS SQL Serwer w kontenerze Docker skonfigurowanym w pliku `docker-compose.yml`.
Przy konfiguracji kontenera wzorowałem się na oficjalnej dokumentacji [14]. Nie chciałem jednak używać obrazu *latest*, jak rekomenduje dokumentacja [14], ponieważ sądzę, że nie jest to dobra praktyka, ponieważ jest on automatycznie zmieniany.
Na szczęście dokumentacja sugeruje alternatywnie użycie konkretnej wersji obrazu

`mcr.microsoft.com/mssql/server:2025-GA-ubuntu`. Zdziwiłem się jednak, gdy spróbowałem pobrać ten obraz, ponieważ okazało się, że **taki obraz nie istnieje**.
Sprawdziłem to na oficjalnej liście tagów tego obrazu [15]. Spojrzałem też na informację o aktualnych wersjach SQL Server [16], gdzie dowiedziałem się, że wersja 2025 nie została jeszcze oficjalnie wydana, co sugerował podany w dokumentacji [14] tag `2025-GA-ubuntu`, gdzie w nomenklaturze Microsoft GA oznacza publicznie wydaną wersję.
Najwyraźniej **dokumentacja Microsoft [14] podaje zmyślony tag do wersji MS SQL Server, która jeszcze nie istnieje**.

Gdy już to zrozumiałem, zdecydowałem się użyć najnowszej stabilnej wersji SQL Server 2022.
Skonfigurowałem edycję SQL Server na developerską i zmieniłem domyślny Collation [17], tak żeby wspierał kodowanie UTF-8, ponieważ wiem, że większość danych w bazie zmieści się w jednym bajcie UTF-8.
W oddzielnym artykule dokumentacji [18] znalazłem informację, gdzie kontener przechowuje dane i zamontowałem tam volumen.

Działanie SZBD sprawdziłem za łącząc się z terminala narzędziem `sqlcmd` i pobierając wersję zapytaniem (Wykaz 9).

Wykaz 9 Pobranie wersji MS SQL Server

```
1 SELECT @@VERSION;
2 GO;
```

Otrzymałem spodziewany wynik (Wykaz 10).

Wykaz 10 Wynik zapytania pobierającego wersję MS SQL Server

```
1 Microsoft SQL Server 2022 (RTM-CU21-GDR) (KB5068406) - 16.0.4222.2 (X64)
2     Oct  3 2025 16:55:17
3     Copyright (C) 2022 Microsoft Corporation
4     Developer Edition (64-bit) on Linux (Ubuntu 22.04.5 LTS) <X64>
```

8 Analiza zmian między SZBD

8.1 Artykuły porównawcze

Analizę zmian zacząłem od zapoznania się z artykułem od *Google Cloud PostgreSQL vs SQL Server: What are the key differences?* [19]. Okazał się on jednak bardzo powierzchowny i prawdopodobnie mijał się miejscami z prawdą. W tabeli porównującej

prezentowane jest w tym artykule, że Microsoft SQL Server "Uses Transact-SQL or T-SQL (standard SQL + extra functionality)" a PostgreSQL "Uses Standard SQL". Z mojego doświadczenia i dokumentacji [20] wynika, że Postgres używa własnej wersji składni `PL/pgSQL`, a nie standardowego SQL.

Następnie przeczytałem bardziej obszerny artykuł *A Complete Comparison of PostgreSQL vs Microsoft SQL Server* [21]. Wydaje się on rzetelny, ale zauważyłem, że o ile są w artykule opisane rodzaje indeksów PostgreSQL, to dla Microsoft SQL Server (MSSQL) wspomniane są tylko indeksy *clustered* i *nonclustered* oraz automatyczne indeksy tworzone na kluczach i *constraint*. Sugeruje to, że MSSQL nie ma konkretnych typów indeksów, co nie jest prawdą [22]. Podobne nieścisłości zauważyłem w sekcji opisującej wyzwalacze.

8.2 Wpływ różnic

Typ binarny `VARBINARY` w Transact-SQL (T-SQL) wymaga podania maksymalnej długości (można podać `max`) [23], a używany do tej pory przeze mnie `BYTEA` w PostgreSQL tego nie wymaga [24].
Różnią się też trochę typy do przechowywania dat [21].
Oznacza to, że będzie konieczna zmiana typów niektórych kolumn w bazie danych.

Odpowiednikiem automatycznego generowania sekwencyjnych wartości `GENERATED ALWAYS AS IDENTITY` z Postgres w MSSQL jest `IDENTITY(1,1)` [25].

MSSQL ma wbudowany scheduler [21], [26], więc nie będę musiał instalować rozszerzeń lub korzystać z zewnętrznych narzędzi.

Składnia elementów programowalnych jest nieco inna [21]. Będzie konieczna konwersja kodu do składni T-SQL.

Przeczytałem też, że MSSQL nie ma wsparcia dla `JSON`, które jest dostępne w Postgres [27]. Nie korzystałem jednak z tej funkcji.

9 Migracja na inny SZBD

9.1 SQL Server Migration Assistant

Początkowo miałem nadzieję użyć oficjalnego narzędzia od Microsoft SQL Server Migration Assistant, ponieważ przeczytałem, że jest to możliwe w artykule [27]. Niestety narzędzie to wspiera tylko migrację z MySQL i Oracle [28], z SZBD omawianych w zadaniu.

Oceniam to narzędzie jako nieprzydatne do mojej migracji.

9.2 pg_dump

Post na *Stack Overflow* [29] opisuje, że jednym ze sposobów na migrację jest wygenerowanie skryptów SQL w możliwie uniwersalnej postaci do utworzenia struktury tabel i danych, a potem wczytanie ich w innym SZBD.

Z podobnym podejściem spotkałem się w wątku na *Server Fault* [30]. Autor opisywał w nim też transformację formatu wyeksportowanych danych skryptem i wczytanie ich narzędziem `bcp`. Do przeniesienia schematu bazy sugerował narzędzie Full Convert.

Sądzę, że to podejście nie przeniesie żadnych elementów programowalnych, a jedynie dane i schemat, co i tak może wymagać ręcznych poprawek.
Chciałbym spróbować bardziej automatycznego narzędzia, ale możliwe, że wrócę do tego podejścia, ponieważ mimo swoich ograniczeń wydaje się mało zawodne.

9.3 DBConvert

Narzędzie DBConvert automatycznie migruje tabele, indeksy i widoki oraz dane z bazy PostgreSQL do MSSQL.
Dokonywane jest mapowanie typów, a w przypadku braku jednoznacznego dopasowania narzędzie pozwala na ręczne dobranie docelowych typów.
Wygląda na to, że nie wykonuje migracji elementów programowalnych [31].
Narzędzie to jest dostępne tylko w wersji na system Windows.

Migracja danych i schematu z narzędziem DBConvert wydaje się łatwiejsza niż za pomocą `pg_dump`, jednak chciałbym użyć narzędzia, które zmigruje też automatyzację bazy. Niedostępność na systemie MacOS sprawia, że nie mogę bezpośrednio uruchomić tego narzędzia.

9.4 Sqlines

Narzędzie Sqlines składa się z dwóch modułów i wspiera migrację z PostgreSQL do MSSQL [32].
Moduł *SQLines Data* realizuje migrację danych i schematu. Moduł *SQLines SQL Converter* pozwala przekonwertować skrypt SQL do formatu docelowej bazy (T-SQL). Możliwa jest też konwersja pojedynczych plików w wersji online.

Wersja próbna narzędzia wprowadza nietypowe ograniczenia.
Moduł *SQLines Data* może nie zaimportować jednego wiersza z oryginalnej tabeli, a moduł *SQLines SQL Converter* może dodawać lub usuwać komentarze [33].

Niestety najnowsza wspierana wersja bazy PostgreSQL to 16.x [32], a niestety moja baza jest w wersji 18.0.
Narzędzie to jest dostępne tylko w wersji na system Windows.
Mimo tych potencjalnych ograniczeń chciałbym je wypróbować z uwagi na wsparcie

konwersji skryptów SQL. Nie spotkałem się jeszcze z innym narzędziem, które by to robiło.

9.5 DBeaver

DBeaver to uniwersalny program to pracy z bazami danych, który wspiera też migrację danych [34].
Dostępna jest darmowa wersja nawet na MacOS.

Ponieważ brak importu pojedynczych wierszy Sqlines (Sekcja 9.4), będzie kłopotliwy przy kluczach obcych, chciałbym użyć DBeaver do importu danych, a Sqlines do dostosowania elementów programowalnych.

9.6 Ispirer

Firma Ispirer oferuje narzędzia do migracji baz danych. Niestety nie wspierają PostgreSQL jako źródła, a jedynie jako docelową bazę [35].

10 Migracja bazy

Zdecydowałem się na hybrydowe podejście, w którym użyję DBeaver do importu danych i Sqlines do dostosowania elementów programowalnych.
Wybrałem DBeaver ponieważ jest dostępny na MacOS i posiada darmową wersję.
Sqlines nie jest dostępny na MacOS, ale ponieważ operuje na plikach kodu SQL to stosunkowo łatwo będzie mi skorzystać z tego narzędzia na innym komputerze. Niestety narzędzie to wspiera tylko wersje PostgreSQL do 16.x, ale liczę na to, że mój kod działający na PostgreSQL 18.0 będzie kompatybilny.

Myślałem nad zastosowaniem wyłącznie obu narzędzi Sqlines, jednak obawiałem się ograniczenia wersji próbnej, która może nie zaimportować po jednym wierszu z każdej tabeli. Spodziewam się, że spowodowałoby to problemy z kluczami obcymi, które musiałbym ręcznie naprawiać.
Musiałbym też przenieść dane bazy na Windows. Spodziewam się, że byłoby to możliwe za pomocą eksportu i importu volumenów Docker [36], ale wolę tego uniknąć.

10.1 Dane

10.1.1 DBeaver

DBeaver wymaga podania istniejącej bazy i schemat, do której zostaną zmigrowane wybrane tabele. W tym celu utworzyłem nową bazę *boardbase* na MSSQL i schemat `main`.

Do utworzenia schematu spróbowałem przekonwertować skrypt 1_create-schema.sql Sqlines online. Niestety narzędzie zmieniło tylko komentarz. Kod SQL pozostał bez zmian (poza dodaniem `GO`) i nie działał na MSSQL. Poprawiłem go ręcznie.

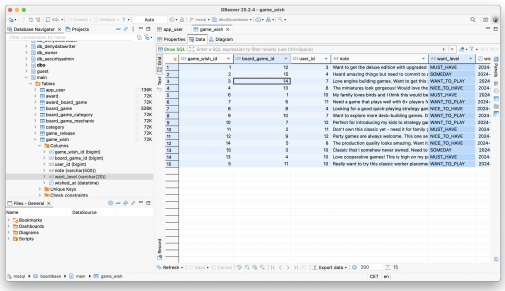
Dowiedziałem się, że w MSSQL nie ma konceptu `search_path` [37], ale można skonfigurować domyślny schemat dla danego użytkownika [38]. Niestety nie udało mi się tego zrobić dla użytkownika `sa`.

Po uruchomieniu migracji napotkałem błąd przy tworzeniu tabeli `game_wish`.

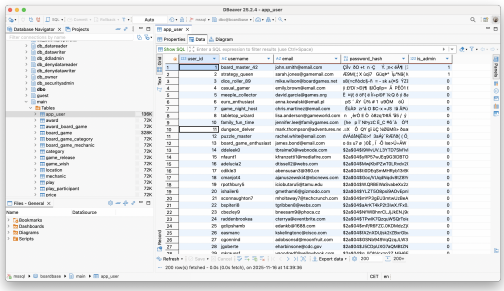
> SQL Error [4188] [S0001]: Column or parameter 'want_level' has type 'text' and collation 'Latin1_General_100_CI_AS_SC_UTF8'. The legacy LOB types do not support UTF-8 or UTF-16 encodings. Use types varchar(max), nvarchar(max) or a collation which does not have the _SC or _UTF8 flags. Column or parameter 'want_level' has type 'text' and collation 'Latin1_General_100_CI_AS_SC_UTF8'. The legacy LOB types do not support UTF-8 or UTF-16 encodings. Use types varchar(max), nvarchar(max) or a collation which does not have the _SC or _UTF8 flags.

Zauważyłem, że DBeaver zmienił typ kolumny `want_level` na `text`. Oryginalnie używałem własnego typu `ENUM`. `ENUM` nie jest wspierany w MSSQL [39].
Zmieniłem domyślne mapowanie typu kolumny `want_level` na `nvarchar(20)`.

Po tej poprawie dane zostały poprawnie zaimportowane do MSSQL (Rysunek 1). Sprawdziłem, że tabela `game_wish` zawiera poprawne informacje w kolumnie `want_level`, teraz zapisane jako zwykły tekst (Rysunek 1a).
Sprawdziłem też, czy zostały przeniesione moje `CONSTRAINT` dbające na przykład o poprawność formatu maili czy niedopuszczanie pustych wartości w niektórych kolumnach. Niestety żadne z nich nie zostały przeniesione, nawet tak podstawowe jak `UNIQUE`.
Nie przeniosły się też klucze obce.



(a) Zmigrowana tabela game_wish



(b) Zmigrowana tabela app_users

Rysunek 1: Migracja danych za pomocą DBeaver

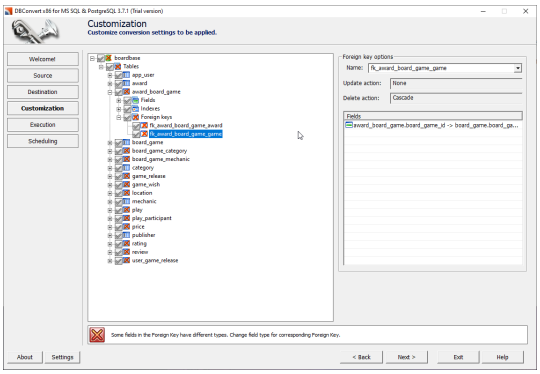
Nie jest to zadowalający mnie wynik. Chciałbym wypróbować narzędzi działających na systemie Windows. W tym celu wyeksportowałem Docer'owy volumen z danymi i

przeniosłem go komputer z systemem Windows.

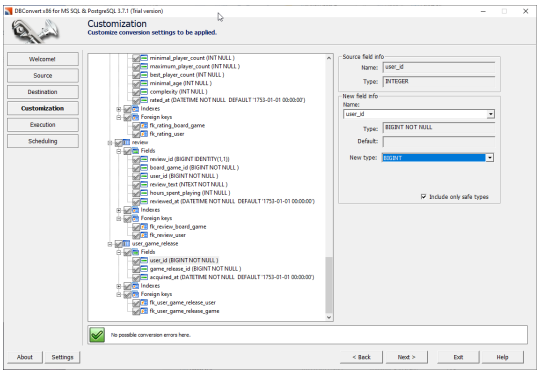
10.1.2 DBConvert

Po wczytaniu volumenu z danymi i odpaleniu baz na Widowsie uruchomiłem narzędzie DBConvert.

Nowy typ dla kolumny `want_level` narzędzie ustawiło domyślnie na VARCHAR(800), co zadziałałoby lepiej niż `TEXT`, ale i tak poprawiłem typ ręcznie na `VARCHAR(20)`. Domyślnie narzędzie DBConvert pokazywało problemy z kluczami obcymi (Rysunek 2a). Klucze główne miały ustawiony typ `BIGINT`, a klucze obce `INT`. Rozwiązałem to zmieniając typy kluczy obcych na `BIGINT`.



(a) Domyślne ustawienie typów



(b) Poprawione klucze obce

Rysunek 2: Konfiguracja importu danych w DBConvert

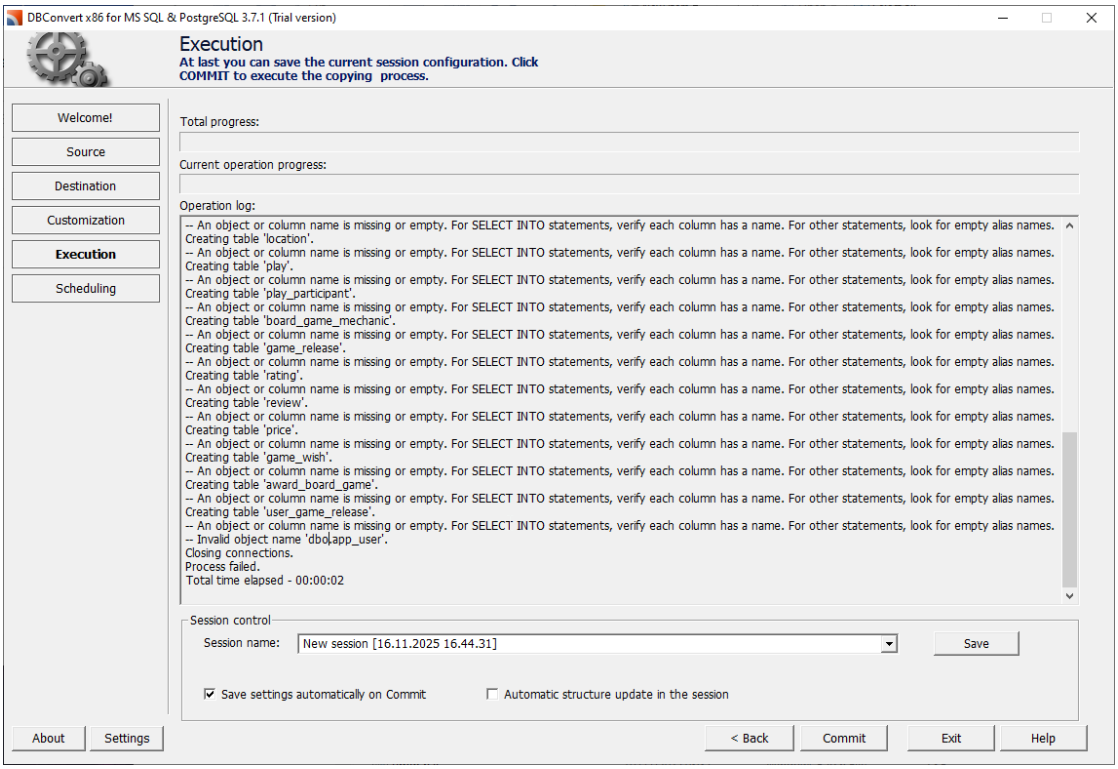
Niestety po uruchomieniu migracji kończyła się ona błędami (Rysunek 3).

10.1.3 Full Convert

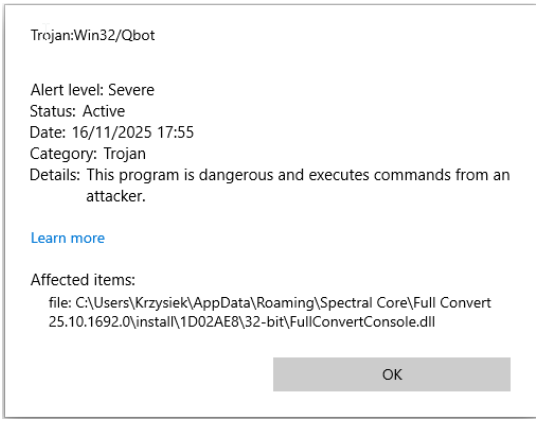
Chciałem wypróbować jeszcze narzędzie Full Convert. Jest ono dostępne tylko na Windows, ale na to byłem już przygotowany.

Podczas instalacji Windows Defender oznaczył jeden z plików jako trojan, co spowodowało niepowodzenie instalacji (Rysunek 4). Wolałem nie ryzykować i zrezygnowałem z użycia tego narzędzia.

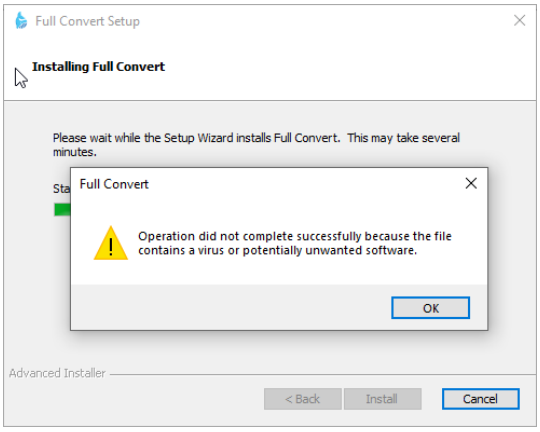
Uznałem, że **wykorzystam import danych**, który zrobiłem za pomocą **DBeaver**. Klucze obce i `CONSTRAINT` dodam dodam na kolejnym etapie.



Rysunek 3: Błędy migracji DBConvert



(a) Problem wykryty przez Windows Defender



(b) Instalacja zakończona niepowodzeniem

Rysunek 4: Problemy z instalacją Full Convert

10.2 Skrypty SQL

Elementy programowalne nie zostały przeniesione narzędziami Extract Transform Load (ETL). Nie przeniosły się też klucze i `CONSTRAINT`.
Przekonwertuję skrypty, żeby odtworzyć elementy programowalne i dodać brakujące relacje.

Za pomocą polecenia Command Line Interface (CLI) *SQLines SQL Converter* przekonwertowałem moje skrypty do składni T-SQL.
Konwersja zadziałała bez błędów, lecz skrypty wymagały ręcznych poprawek. Zauważyłem, że dla niektórych instrukcji narzędzie przepisało zupełnie bez zmian oryginalny kod, mimo, że korzysta on wyraźnie ze składni Postgres np. w 2_create-custom-types.sql.

Dostosowałem kod tworzący tabele ręcznie, za pomocą find and replace z wyrażeniami regularnymi. Poprawiłem też kod wczytujący dane, żeby od razu przetestować kolejne elementy.
Rozważałem zdjęcie relacji i wczytanie danych DBeaver i ich przywrócenie, lecz zdecydowałem się na wczytanie danych na podstawie moich skryptów.

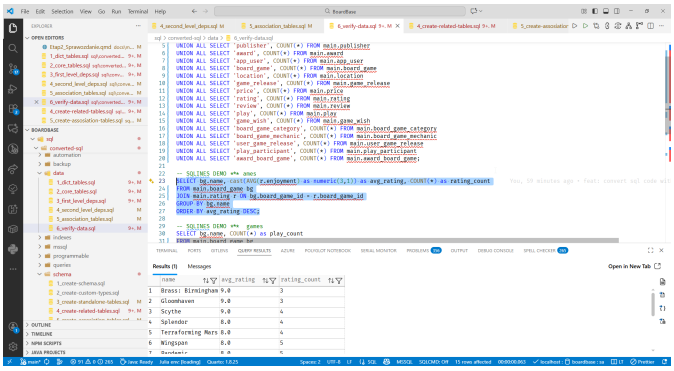
- dodałem schemat `main` do nazw tabel,
- poprawiłem deklarację `IDENTITY` na `IDENTITY(1,1)`,
- usunąłem niedziałające dodawanie komentarzy `EXECUTE sp_addextendedproperty`,
- usunąłem porównania używające wyrażeń regularnych `~` zastępując je `LIKE`,
- zmieniłem `ON DELETE RESTRICT` na `SET` ustawiający odpowiednią wartość,
- znów zamieniłem użycie enum na `VARCHAR(20)` w kolumnie `want_level`,
- dodałem nazwę schematu `main` do konfiguracji kluczy obcych `REFERENCES`,
- skrypt do wylistowania tabeli przepisałem ręcznie,
- dodałem nazwę schematu `main` do nazw tabel w `INSERT INTO`,
- poprawiłem wczytanie binarnych danych,
- zamieniłem `TRUE` i `FALSE` na `1` i `0`,
- wykomentowałem independentnet `TRUNCATE TABLE` i resetowanie sekwencji, ponieważ nie działało.

W ten sposób udało mi się odtworzyć strukturę bazy razem z relacjami i wczytać dane, co demonstruję wykonując przykładowe proste zapytanie Rysunek 5.

Bardziej złożone zapytania zostały poprawnie przekonwertowane i działają bez mojej ingerencji.

Instrukcje tworzące indeksy zamieniłem ręcznie na `CREATE INDEX` i `CREATE UNIQUE INDEX` bez podawania typu ineksów.
Kod testowania indeksów wykomentówałem, ponieważ pozostał w składni Postgres.

Do poprawy widoku `game_catalog` spróbowałem wykorzystać Large Language Model (LLM) w chacie z Gemini. W drugiej iteracji widok zadziałał poprawnie.



Rysunek 5: Przykładowe zapytanie na MSSQL

Ręcznie przepisałem tworzenie grup 1_create-groups.sql i użytkowników 2_create-users.sql.
Tworząc użytkowników musiałem lekko rozbudować ich hasła, ponieważ MSSQL narzuca wymagania siły haseł.

Skoro LLM dobrze poradził sobie z przepisaniem kodu widoku postanowiłem użyć go też do aktualizacji kodu wyzwalaczy `triggers.sql` (chat z Gemini).

W kolejnym chacie przekonwertowałem proste procedury create-records-procedures.sql. Przy testowaniu zauważyłem, że użytkownik *casual_gamer* nie ma uprawnień do wykonania procedury. Dodałem ręcznie uprawnienie do uruchomienia procedury.

Źródła

1. Bayesian averages in custom ranking - Algolia. https://www.algolia.com/doc/guides/managing-results/must-do/custom-ranking/how-to/bayesian-average. Accessed 1 lis 2025

2. PostgreSQL: Documentation: 18: 41.3. Declarations. https://www.postgresql.org/docs/current/plpgsql-declarations.html#PLPGSQL-DECLARATION-COLLATION. Accessed 2 lis 2025

3. Steiger S sql - How to perform a select query in a DO block? - Stack Overflow. https://stackoverflow.com/questions/14652477/how-to-perform-a-select-query-in-a-do-block. Accessed 2 lis 2025

4. PostgreSQL: Documentation: 18: CREATE TRIGGER. https://www.postgresql.org/docs/current/sql-createtrigger.html. Accessed 7 lis 2025

5. Fireship I replaced my entire tech stack with Postgres... - YouTube. https://www.youtube.com/watch?v=3JW732GrMdg. Accessed 13 paź 2025

6. Kozlov P Comparing PostgreSQL Job Schedulers: pg_cron vs. pgAgent. https://medium.com/@peter.kozlov/comparing-postgresql-job-schedulers-pg-cron-vs-pgagent-95a2dc96488e. Accessed 9 lis 2025

7. k-caps pg agent vs pg cron. https://gist.github.com/k-caps/27f9e4f3504a974ee14bb8f8762f68ef. Accessed 9 lis 2025

8. Docker Hub: postgres - Official Image. https://hub.docker.com/_/postgres. Accessed 9 lis 2025

9. Docker Hub: cleisonfmelo/postgres-pg-cron - Docker Image. https://hub.docker.com/r/cleisonfmelo/postgres-pg-cron. Accessed 9 lis 2025

10. PostgreSQL: Documentation: 8.1: Backup and Restore. https://www.postgresql.org/docs/8.1/backup.html. Accessed 10 lis 2025

11. PostgreSQL: Documentation: 8.1: On-line backup and point-in-time recovery (PITR). https://www.postgresql.org/docs/8.1/backup-online.html#BACKUP-ARCHIVING-WAL. Accessed 10 lis 2025

12. PostgreSQL: Documentation: 8.1: File system level backup. https://www.postgresql.org/docs/8.1/backup-file.html. Accessed 10 lis 2025

13. PostgreSQL: Documentation: 8.1: Backup and Restore. https://www.postgresql.org/docs/8.1/backup.html#BACKUP-DUMP. Accessed 10 lis 2025

14. amitkh-msft Docker: Run Containers for SQL Server on Linux - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-ver17&tabs=cli&pivots=cs1-bash#prerequisites. Accessed 13 lis 2025

15. Microsoft Artifact Registry: mssql/server/tags. https://mcr.microsoft.com/en-us/artifact/mar/mssql/server/tags. Accessed 13 lis 2025

16. aartig13 Latest updates and version history for SQL Server - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/troubleshoot/sql/releases/download-and-install-latest-updates. Accessed 13 lis 2025

17. WilliamDAssafMSFT Collation and Unicode Support - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/relational-databases/collations/collation-and-unicode-support?view=sql-server-ver17#utf-8-support. Accessed 13 lis 2025

18. amitkh-msft Configure and Customize SQL Server Docker Containers - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/linux/sql-server-linux-docker-container-configure?view=sql-server-ver17&pivots=cs1-bash#persist-your-data. Accessed 13 lis 2025

19. PostgreSQL vs. SQL Server: What's the difference? | Google Cloud. https://cloud.google.com/learn/postgresql-vs-sql. Accessed 14 lis 2025

20. PostgreSQL: Documentation: 18: 41.1. Overview. https://www.postgresql.org/docs/current/plpgsql-overview.html. Accessed 14 lis 2025

21. A Complete Comparison of PostgreSQL vs Microsoft SQL Server | EDB. https://www.enterprisedb.com/blog/microsoft-sql-server-mssql-vs-postgresql-comparison-details-what-differences. Accessed 14 lis 2025

22. MikeRayMSFT Indexes - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/relational-databases/indexes/indexes?view=sql-server-ver17. Accessed 14 lis 2025

23. MikeRayMSFT binary and varbinary (Transact-SQL) - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/t-sql/data-types/binary-and-varbinary-transact-sql?view=sql-server-ver17. Accessed 14 lis 2025

24. PostgreSQL: Documentation: 18: 8.4. Binary Data Types. https://www.postgresql.org/docs/current/datatype-binary.html. Accessed 14 lis 2025

25. VanMSFT IDENTITY (Property) (Transact-SQL) - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql-identity-property?view=sql-server-ver17#examples. Accessed 14 lis 2025

26. rwestMSFT Create a Schedule | Microsoft Learn. https://learn.microsoft.com/en-us/ssms/agent/create-a-schedule. Accessed 14 lis 2025

27. (2023) How to Migrate PostgreSQL to SQL Server Step by Step | Estuary. https://estuary.dev/blog/postgres-to-sql-server/. Accessed 15 lis 2025

28. SQL Server Migration Assistant - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/ssma/sql-server-migration-assistant?view=sql-server-ver17. Accessed 15 lis 2025

29. (2011) Stack Overflow: sql server - How to migrate a PostgreSQL database into a SQLServer one? https://stackoverflow.com/questions/6563846/how-to-migrate-a-postgresql-database-into-a-sqlserver-one. Accessed 15 lis 2025

30. (2009) Server Fault: sql server - Best tool to migrate a PostgreSQL database to MS SQL 2005? https://serverfault.com/questions/65407/best-tool-to-migrate-a-postgresql-database-to-ms-sql-2005. Accessed 15 lis 2025

31. PostgreSQL to SQL Server conversion and synchronization. | DBConvert. https://dbconvert.com/postgresql/mssql/. Accessed 15 lis 2025

32. PostgreSQL to Microsoft SQL Server Migration - SQLines Tools. https://www.sqlines.com/postgresql-to-sql-server. Accessed 15 lis 2025

33. Download Tools - SQLines Tools. https://www.sqlines.com/download. Accessed 15 lis 2025

34. Data migration | DBeaver Documentation. https://dbeaver.com/docs/dbeaver/Data-migration/. Accessed 15 lis 2025

35. Requesting a License. https://www.ispirer.com/requesting-license?ttype=database&plan=free&cost=0. Accessed 16 lis 2025

36. Volumes | Docker Docs. https://docs.docker.com/desktop/use-desktop/volumes/#import-a-volume. Accessed 16 lis 2025

37. (2019) sql server - Does T-SQL have a Schema search path? - Database Administrators Stack Exchange. https://dba.stackexchange.com/questions/240259/does-t-sql-have-a-schema-search-path. Accessed 16 lis 2025

38. ALTER USER (Transact-SQL) - SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-user-transact-sql?view=sql-server-ver17. Accessed 16 lis 2025

39. SHAIK S (2023) ENUM Equivalent in SQL Server - Big Data & SQL. https://bigdatansql.com/2023/03/25/enum-equivalent-in-sql-server/. Accessed 16 lis 2025

Zastosowane skróty

CLI Command Line Interface
CTE Common Table Expression
ETL Extract Transform Load
LLM Large Language Model
MSSQL Microsoft SQL Server
ORM Object Relational Mapper
SQL Structured Query Language
SZBD System zarządzania bazami danych
T-SQL Transact-SQL
UDF User Defined Function
WAL Write Ahead Log