

POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRYCZNY

KIERUNEK INFORMATYKA STOSOWANA

Migracja MySQL do PostgreSQL

Wykonał:
Aliaksandr KAROLIK

Sprawdzający:
dr inż. Krzysztof HRYNIÓW

20 listopada 2021



Spis treści

1	Wybrana baza danych do migracji	2
2	Porównanie MySQL z PostgreSQL	2
2.1	Typy baz	2
2.2	Przechowywane typy danych	2
2.3	Indeksy	2
2.4	Bezpieczeństwo	2
2.5	Różniczy dialektowe	3
3	Konfiguracja PostgreSQL	3
4	Migracja	3
4.1	Migracji funkcji	4
4.2	Migracja procedur	8
4.3	Migracja trigera	8
5	Wnioski	10

1 Wybrana baza danych do migracji

Obecnie baza została stworzona w oparciu o MySQL, zdecydowałem, że docelową bazą danych będzie PostgreSQL. Tak samo, jak wcześniej baza danych zostanie skonfigurowana w kontenerze dockerowym. Wybraną wersję PostgreSQL była 10.5 (Debian 10.5-2.pgdg90+1).

2 Porównanie MySQL z PostgreSQL

2.1 Typy baz

PostgreSQL to obiektowo-relacyjny system zarządzania bazą danych (ORDBMS), w którym nacisk położono na rozszerzalność i zgodność ze standardami. PostgreSQL jest zgodny z ACID, transakcyjny, posiada aktualizowane i zmateriałizowane widoki, wyzwalacze oraz klucze obce. Obsługuje również funkcje i procedury składowane.

MySQL jest systemem zarządzania relacyjną bazą danych (RDBMS) typu open-source. Podobnie jak PostgreSQL i wszystkie inne relacyjne bazy danych, MySQL używa tabel jako głównego komponentu i posiada mniej więcej taki sam zestaw funkcji jak PostgreSQL. Nowsze wersje MySQL (5.7+) obsługują nawet niektóre funkcje noSQL.

2.2 Przechowywane typy danych

Podczas analizy różnicy pomiędzy tymi bazami danych rzuciło mi się w oczy to że Postgres oferuje szerszą gamę typów danych niż MySQL. MySQL wspiera następujące typy danych: Numeric, date/time, character, spatial, JSON natomiast Postgre obsługuje również te typy danych oprócz spatial, ale dodatkowo wspiera takie typy danych jak XML, HSTORE, arrays, ranges, composite. Mając to na uwadze podczas tworzenia aplikacji musimy się zastanowić czy aplikacja operuje na unikalnych typach danych, które są dostępne MySQL, jeżeli nie i potrzebujemy obsługi danych niestrukturyzowanych to PostgreSQL może być lepszym wyborem.

2.3 Indeksy

Również ciekawą różnicę stanowią indeksy w tych bazach. Oby dwie bazy wspierają indeksy jednak algorytmy, które są używane podczas indeksowania są różne. MySQL wspiera B-tree; R-tree, hash jednak Postgre udostępnia szerszą listę algorytmów do indeksowania takich jak GiST, SP-GiST, GIN, and BRIN.

2.4 Bezpieczeństwo

Również ciekawostką dla mnie bezpieczeństwo w tych bazach. Obie bazy danych wspierają zarządzanie użytkownikami i grupami oraz nadawanie uprawnień według roli. PostgreSQL obsługuje filtrowanie klientów w oparciu o IP oraz uwierzytelnianie za pomocą PAM i Kerberos, natomiast MySQL obsługuje PAM, natywne usługi windowsowe

oraz LDAP do uwierzytelniania użytkowników. Pod względem bezpieczeństwa obie bazy danych posiadają porównywalne opcje.

2.5 Różnice dialektowe

Różnice dialektowe które zauważyłem pomiędzy PostgreSQL i MySQL są następujące:

- W PostgreSQL wielkość liter w zapytaniach ma znaczenie. Czyli WHERE name = 'John' oraz WHERE name = 'john' nie jest tym samym. W przypadku MySQL tego nie ma.
- W przypadku stosowania cudzysłowów MySQL radzi z name = 'John' oraz name = "John". PostgreSQL działa tylko z name = 'John'.
- W przypadku współpracy z danymi typu datetime. PostgreSQL mają w nazwach funkcji dolne podkreślenie czyli CURRENT_DATE() CURRENT_TIME(). MySQL ma funkcje standardowe czyli CURDATE() CURTIME().

3 Konfiguracja PostgreSQL

```
postgres:
  image: postgres:10.5
  restart: always
  environment:
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=postgres
  logging:
    options:
      max-size: 10m
      max-file: "3"
  ports:
    - '5438:5432'
  volumes:
    - postgres-data:/var/lib/postgresql/data
```

4 Migracja

Jako główne narzędzie do migracji wybrałem **airbyte.io** również była podjęta próba migracji przy pomocy **PgLoader** ale nie udało się do końca poprawnie skonfigurować to narzędzie. Ze względu na brak czasu przeszedłem w stronę **airbyte.io**. Szczerze mówiąc, to narzędzie było najlepszym do tej pory softem, który używałem. Cały proces instalacji polegał na pobraniu git repozytorium oraz odpalenia docker-compose. Po odpaleniu komendy bezpośrednio mogłem zajmować się konfiguracją procesu migracji.

Proces polegał na ustawieniu odpowiedniego źródła oraz punktu końcowego, czyli źródło MySQL, oraz PostgreSQL jak punkt końcowy. Na rysunku 1 została przedstawiona część konfiguracji narzędzia. Definicja punktu końcowego wygląda identycznie, jak i źródła.

Zalety narzędzia to:

- łatwe postawienie nie mogę tego jeszcze raz nie powiedzieć, bo to było najlepsze doświadczenia z nowym narzędziem w ciągu całych studiów, duży szacunek dla developerów,
- przeniesienie wszystkich danych z wykreowaniem ich hashy, aby mieć pewność, że cały proces przeszedł poprawnie,
- możliwość ustawienia cyklicznej migracji danych. Podczas testowania zostało to ustawione na manualny tryb. Na rysunku 2 jest przedstawiona możliwość konfiguracji,
- możliwość wprowadzenia normalizacji podczas migracji danych.

Każde narzędzie ma jakieś wady i to narzędzie nie jest wyjątkiem, podczas migracji nie są przenoszone triggerzy, procedury, funkcji, indeksy. Ale trudno mi powiedzieć, że to jest wada, bo z informacji z internetu wynika, że proces migracji nie polega na przeniesieniu funkcjonalność, polega on na przeniesieniu danych i z tym zadaniem narzędzie świetnie poradziło.

Dodatkowym minusem narzędzia jest generowanie dużego zbioru dodatkowych danych typu tabeli które zawierają informacje o źródłowych tabeli zazwyczaj takie tabeli mają prefiks `airbyte_raw`. Przykład takiej tabeli jest przedstawiony na rysunku 3 taka tabela zawiera hashe które są walidowane przez narzędzie podczas synchronizacji.

Również narzędzie dodaje hasze w tabeli docelowe które przy pomocy których można zwalidować działanie narzędzia. Na rysunku 4 jest przedstawiony proces synchronizacji jednego rekordu jak widać hasze innych rekordów nie zostały zmienione co świadczy o tym że nie były one nadpisywane. Dalej zostanie przedstawiony proces przepisywania funkcjonalności bazy MySQL do PostgreSQL.

4.1 Migracji funkcji

Migrację funkcjonalności rozpocząłem od przeniesienia funkcji i miałem bardzo dobre nastawienia, bo wydawało mi się, że to będzie dosyć proste zadanie, ale okazało się wręcz przeciwnie. Pierwsze napotkany problem był taki, że wybrana wersja PostgreSQL nie miała funkcji wbudowanej do obliczenia różnicy pomiędzy datami przypominam, że zaimplementowana funkcja obliczała długość lotu.

Podczas rozwiązywania tego problemu niszczyły mnie bezsensowne komunikaty o błędach które dostawałem od PostgreSQL a czasami zdarzała się sytuacja, że napisana funkcja była akceptowana przez bazę, ale po uruchomieniu wyrzucała błąd. Nie rozumiem czemu błąd nie był wyświetlany przy kroku dodawania funkcji. Na rysunku 5 został zaprezentowany wynik działania funkcji jak widać uzyskany wynik jest poprawny.

0

2

3

+

Create your first source

Sources are tools where the data will be replicated from.

Name * - Pick a name to help you identify this source in Airbyte

aircompany_id

Source type

MySQL

[Setup Guide](#)

host * - Hostname of the database.

127.0.0.1

port * - Port of the database. (e.g. 3306)

3317

database * - Name of the database.

airlines

username * - Username to use to access the database.

root

password - Password associated with the username.

jdbc_url_params - Additional properties to pass to the jdbc url string when connecting to the database formatted as 'key=value' pairs separated by the symbol '&'. (example: key1=value1&key2=value2&key3=value3)

(a) Ustawienie źródła

Source connector

aircompany_id

Destination connector

postgresql

Sync frequency

manual

Namespace Configuration - Define which namespace should be used in the destination. See more on [Connection docs](#)

Mirror source structure

Table Prefix - Prefix to prefix all destination tables created as part of this connection. Useful for identifying tables on a per-connection basis and namespacing streams from different sources so they do not overwrite each other.

prefix-

Select the data you want to sync

To save the new schema, click on Save changes. **Note that it will delete all the data in your destination and start syncing from scratch.**

Search name

All

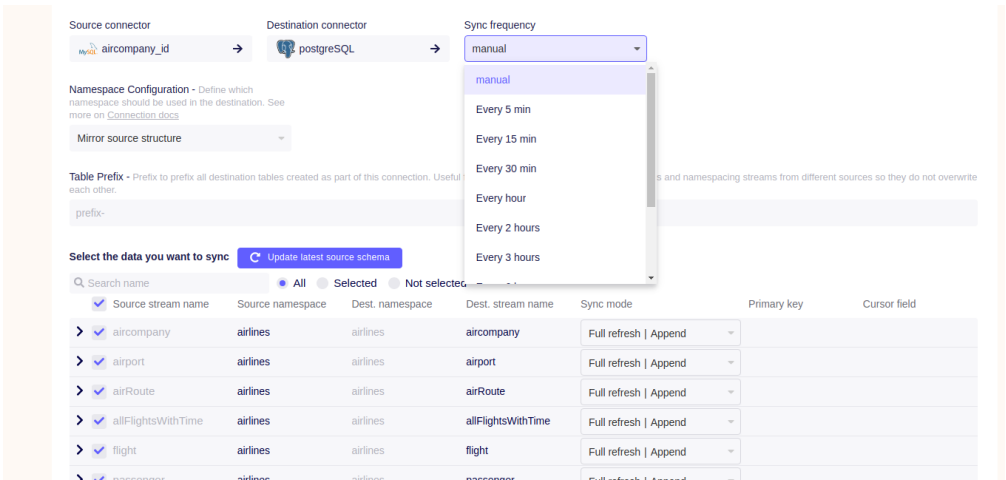
Selected

Not selected

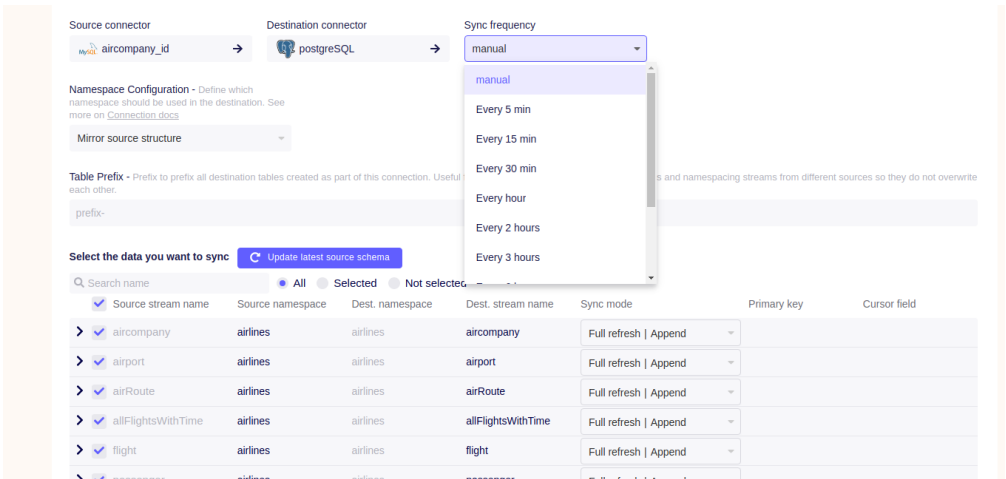
Source stream name	Source namespace	Dest. namespace	Dest. stream name	Sync mode	Primary key	Cursor field
> <input checked="" type="checkbox"/> aircompany	airlines	airlines	aircompany	Incremental Append		aircompany_id ↕
> <input checked="" type="checkbox"/> airport	airlines	airlines	airport	Incremental Append		airport_id ↕
> <input checked="" type="checkbox"/> airRoute	airlines	airlines	airRoute	Incremental Append		airRoute_id ↕
> <input checked="" type="checkbox"/> allFlightsWithTime	airlines	airlines	allFlightsWithTime	Full refresh Append		
> <input checked="" type="checkbox"/> flight	airlines	airlines	flight	Incremental Append		flight_id ↕
> <input checked="" type="checkbox"/> passenger	airlines	airlines	passenger	Incremental Append		passenger_id ↕
> <input checked="" type="checkbox"/> passengersWithDest...	airlines	airlines	passengersWithDest...	Full refresh Append		
> <input checked="" type="checkbox"/> pilot	airlines	airlines	pilot	Incremental Append		pilot_id ↕
> <input checked="" type="checkbox"/> plane	airlines	airlines	plane	Incremental Append		plane_id ↕
> <input checked="" type="checkbox"/> tariff	airlines	airlines	tariff	Incremental Append		tariff_id ↕
> <input checked="" type="checkbox"/> ticket	airlines	airlines	ticket	Incremental Append		ticket_id ↕

(b) Wybór tabeli które mają być przeniesione

Rysunek 1: Konfiguracja airbyte.io



Rysunek 2: Konfiguracja migracji



Rysunek 3: Konfiguracja migracji

Query Editor

```
1 SELECT * FROM airlines.aircompany
```

Data Output

nip	aircompany_id	aircompany_name	_airflyte_ab_id
4854559	1	Ryanair	2346762ecf7-4c73-999b-a560712d65a
1975148	2	Flydubai	64b4083-998e-4c08-86ca-51ee54dees72
5335416	3	Wizzair	15d3f748-48fa-48af-5316-0102e8615886
3316880	4	Vueling	9176254-9621-4b72-9aaf-d5543d316d24
4428129	5	Fair	4592888-402c-4a0f-84ba-e6c116a4537f
48159	6	Test	c582716-8084-49ca-8884-f502046a11b
48159	7	Test2	58b286c-3e53-46a0-88fa-574269331740

Schemas

aircompany

aircompany_id int AI PK
nip int
aircompany_name varchar(30)

Result Grid

#	aircompany_id	nip	aircompany_name
1	1	4854559	Ryanair
2	2	1975148	Flydubai
3	3	5335416	Wizzair
4	4	3316880	Vueling
5	5	4428129	Fair
6	6	48159	Test
7	7	48159	Test2

(a) Tabela przed dodanie rekordu testowego

Query Editor

```
1 SELECT * FROM airlines.aircompany
```

Data Output

nip	aircompany_id	aircompany_name	_airflyte_ab_id
4854559	1	Ryanair	2346762ecf7-4c73-999b-a560712d65a
1975148	2	Flydubai	64b4083-998e-4c08-86ca-51ee54dees72
5335416	3	Wizzair	15d3f748-48fa-48af-5316-0102e8615886
3316880	4	Vueling	9176254-9621-4b72-9aaf-d5543d316d24
4428129	5	Fair	4592888-402c-4a0f-84ba-e6c116a4537f
48159	6	Test	c582716-8084-49ca-8884-f502046a11b
48159	7	Test2	58b286c-3e53-46a0-88fa-574269331740
48159	8	Test3	23454877-d477-4429-8278-d11125528a3

Schemas

aircompany

aircompany_id int AI PK
nip int
aircompany_name varchar(30)

Result Grid

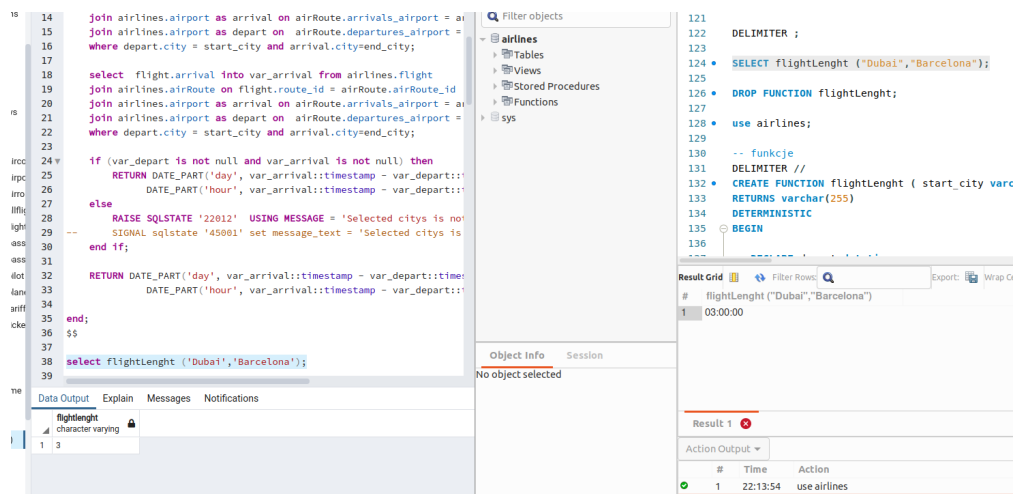
#	aircompany_id	nip	aircompany_name
3	3	5335416	Wizzair
4	4	3316880	Vueling
5	5	4428129	Fair
6	6	48159	Test
7	7	48159	Test2
8	8	48159	Test3

Action Output

#	Time	Action
8	18:50:02	SELECT * FROM airlines.aircompany LIMIT 0, 11
9	18:52:35	Insert into aircompany(nip, aircompany_name
10	18:52:39	SELECT * FROM airlines.aircompany LIMIT 0, 11
11	18:54:45	Insert into aircompany(nip, aircompany_name

(b) Tabela po dodaniu testowego rekordu i synchronizacji

Rysunek 4: Przykład synchronizacji



Rysunek 5: Konfiguracja migracji

4.2 Migracja procedur

Podczas migracji procedur okazało się, że w wersji 10.5 PostgreSQL, która była używana nie ma procedur i trzeba tworzyć funkcje zamiast procedur, które zwracają void. Podczas migracji znów dużo czasu było stracone na walce z dialektem oraz funkcjami wbudowanymi. Ale procedury zostały poprawnie przeniesione. Na rysunkach 6 oraz 7 zostały przedstawione wyniki działania procedur, jak widać, wszystko działa poprawnie.

```

104
105
106 select * from airlines.getraiseByNameAndSurname('Lewie', 'Hearnes');
107
108 SELECT version();
109

```

	name character varying	surname character varying	depart character varying	arrivals character varying	departure_time character varying	arrival_time character varying
1	Lewie	Hearnes	Dubai International Airport	Barcelona-El Prat Airport	2021-01-23T12:45:56Z	2021-01-23T15:45:56Z
2	Lewie	Hearnes	Warsaw Chopin Airport	Charles de Gaulle Airport	2021-02-23T13:45:56Z	2021-02-23T17:45:56Z

Rysunek 6: Wynik procedury po odnalezieniu lotów dla pasażera

4.3 Migracja triggera

Migracja triggerów nie obyła się bez przygód, czyli stabilnie PostgreSQL niszczył mnie error handlingiem i komunikatami o błędach. Drugą ciekawostką dla mnie było

airlines/postgres@postgres

Query Editor Query History

```
1 SELECT * FROM airlines.airroute
2 ORDER BY airroute_id ASC
```

Data Output Explain Messages Notifications

plane_id	aircompany	arrivals_airport	departures_airport	_airbyte_ab_id	_airbyte_emitted_at	_airbyte_normalized_at	_airbyte_airroute_hashid	airroute_id
double precision	double precision	double precision	double precision	character varying	timestamp with time zone	timestamp with time zone	text	integer
1	1	2	2	1f9f2e7e2122-49e4880c39f9a479501b	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.042921+00	air9f2e6c0412b45d018c30edf16af02	1
2	4	1	5	334017a5c171b-439c83acab054ed0e0d8	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.042921+00	9ba4dc0816137a2d3823a6c116f1c251	2
3	2	3	2	470ff2a3c07a5-4c08-a999-7385c59b0676	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.042921+00	b0af532a59a5dc29483ca46a0fda4c4	3
4	5	2	5	18c98038-df28-4636-88c1-fe8541b4dc99	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.042921+00	b9426f09289c1fcd35c25734607c4a58	4
5	1	1	4	2f34122b6-e124-42b2-aa9-627810c82967	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.042921+00	b8520713fa3617a485e8e97932042375	5
6	[null]	[null]	7	6	[null]	[null]	[null]	6

(a) Ustawienie źródła

airlines/postgres@postgres

Query Editor Query History

```
1 SELECT * FROM airlines.airport
2 ORDER BY airport_id ASC
```

Data Output Explain Messages Notifications

city	airport_name	_airbyte_ab_id	_airbyte_emitted_at	_airbyte_normalized_at	_airbyte_airport_hashid	airport_id
character varying	character varying	character varying	timestamp with time zone	timestamp with time zone	text	integer
1 Dubai	Dubai International Airport	111eb972-57fb-41ae-8079-f8e42bbdc115	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.040984+00	a5e7dd3790cef954c8aef9f31e4087	1
2 Barcelona	Barcelona-El Prat Airport	160eb7e7-01fa-4330-95e8-d08ed1fde0aa	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.040984+00	444ee7fa5a56544baf201cf13b28c00	2
3 Warsaw	Warsaw Chopin Airport	d796fa71-94d2-4389-b253-578ac5f216cb	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.040984+00	7295e6aaaf64c95635f3a876ab6555c1	3
4 London	London Heathrow Airport	59e2ca0b-9c9d-428f-927f-994986a22c1b	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.040984+00	84e24d24242ba969d8d712d8e7d1632	4
5 Paris	Charles de Gaulle Airport	3fa525cf-79c2-4dff-b6c5-b07d3fbdcf84	2021-11-16 21:31:30.673+00	2021-11-16 21:31:36.040984+00	7d798dc5de54e38bd3c5941b792edf2	5
6 [null]	Test1	[null]	[null]	[null]	[null]	6
7 [null]	Test2	[null]	[null]	[null]	[null]	7

(b) Wybór tabeli które mają być przeniesione

Rysunek 7: Wynik działania procedury dodającej airRoute

to, że w PostgreSQL nie definiuje się bezpośrednio triggera a proces wygląda tak, że najpierw jest definiowana funkcja, która będzie odpalana w triggerze a później dopiero sam trigger.

Na rysunku 8 został przedstawiony kod oraz wynik działania triggera.

```

111 CREATE OR REPLACE FUNCTION trigger_function_for_passenger()
112 RETURNS TRIGGER
113 LANGUAGE PLPGSQL
114 AS
115 $$
116 declare
117 rowcount int;
118 BEGIN
119
120     SELECT COUNT(*)
121     INTO rowcount
122     FROM airlines.passenger
123     where passenger.passport = new.passport;
124 IF rowcount > 0 THEN
125     RAISE SQLSTATE '45001' USING MESSAGE = 'Passport must be unique. This passenger already in database';
126 END IF;
127 RETURN NEW;
128 END;
129 $$
130
131 CREATE TRIGGER check_passport
132 BEFORE insert
133 ON airlines.passenger
134 FOR EACH ROW
135 EXECUTE PROCEDURE trigger_function_for_passenger();
136 insert into airlines.passenger(name,surname, passport) values('Lewie','Hearnes','2G3WD58236');

```

Data Output Explain Messages Notifications

```

ERROR:  Passport must be unique. This passenger already in database
CONTEXT:  PL/pgSQL function trigger_function_for_passenger() line 11 at RAISE
SQL state: 45001

```

Rysunek 8: Wynik oraz implementacja triggera

5 Wnioski

Cały proces okazał się dużo bardziej skomplikowany niż na początku myślałem po analizie informacji w internecie. Różnice w składni zapytań były bardzo małe, jednak różnice przy deklaracji funkcji, procedur oraz triggerów były bardzo duże względem MySQL. Wiązało się to z dużo, dużo większym debugowaniem oraz szukaniem rozwiązania podczas translacji, która w ciągu całej migracji niszczyła komórki nerwowe mojego mózgu. Nowością dla mnie było rozbitcie wyzwalacza na deklarację funkcji oraz deklarację wyzwalacza wywołującego tę funkcję.

Szczerze mówiąc, ta migracja bardzo mocno zniechęciła mnie do korzystania z PostgreSQL i jest mi bardzo smutno, bo akurat w pracy zaczynają używać tej bazy.

Również nie udało mi się skonfigurować połączenia dblink pomiędzy bazami danych związane to głównie było z brakiem czasu oraz doświadczenia w konfigurowaniu takiego połączenia. Dodatkowe problemy podczas konfiguracji dblinku stanowił docker to również było związane z brakiem doświadczenia konfiguracji kontenerów dla baz danych.