

DB Projekt – Daniel Michrowski

Etap1

1. Opis projektu

W projekcie przedstawiam koncepcje bazy danych dla dowolnego przedsiębiorstwa usługowego – handlowego.

Do stworzenie modelu bazy danych wykorzystano opensourc'owy DBMS PostgreSQL ver. 12.4. System jest darmowy, posiada bogate dokumentacje oraz jest wspierany przez szeroką społeczność – z tych powodów wybrałem system PostgreSQL.

Baza danych składa się z następujących tabel:

- Contact_person
- Product
- Producer
- Client
- Place_object
- Supervisor
- Investment
- Sub_investment
- Product_order

Poniżej przedstawiam, krótki opis tabel:

Contact_person – tabela zawierająca kontakt to przedstawicieli handlowych i osób kontaktowych poszczególnych firm producenckich.

Product – tabela zawierająca nazwy produktów, nazwy producentów, parametry techniczne, typ produktu, cena detaliczna itp.

Producer – Informacje o producencie jak: nazwa producenta, rabaty na poszczególne typy produktów.

Client – tabela zawiera listę klientów, z danymi personalnymi oraz kontaktowymi.

Place_object – przechowuje adresy obiektów inwestycji.

Supervisor – w tabeli przechowywane są informacje o pracownikach firmy. Docelowo tabela posłuży do przypisania opiekuna inwestycji.

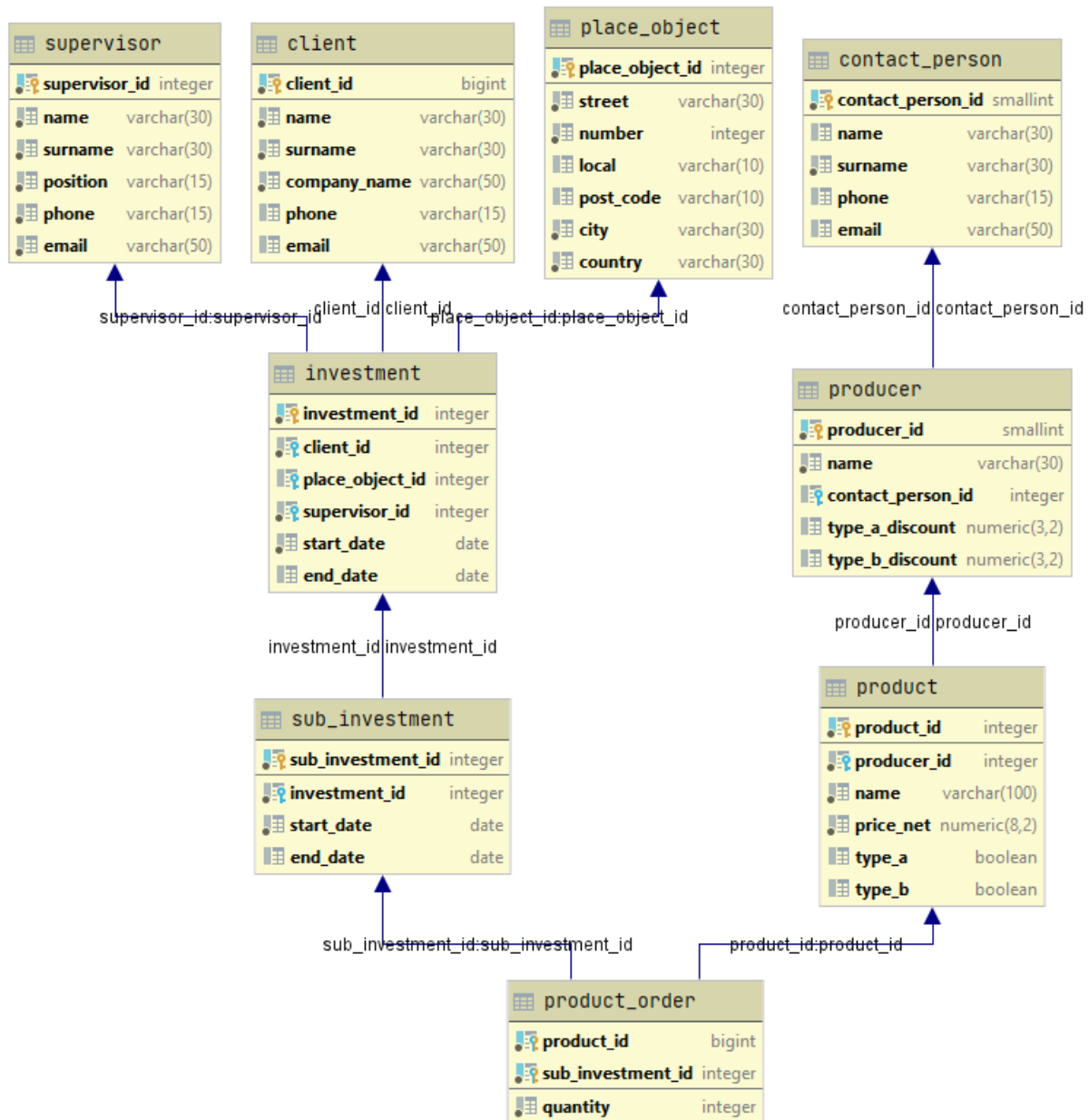
Investment – tabela zawierająca ID obiektu inwestycji, ID klienta oraz ID opiekuna inwestycji oraz czas rozpoczęcia i zakończenia inwestycji.

Sub_investment – tabela zawiera informacje o podinwestycjach. Założono, że jedna inwestycja może grupować pod sobą kilka podinwestycji.

Product_order – tabela przechowująca zamówienia produktów na daną podinwestycję.

2. Schemat ERD (Entity Relationship Diagram)

Schemat przedstawia relacje pomiędzy tabelami wraz z ich atrybutami.



Powered byFiles

W tabeli Product_order wykorzystano klucz kompozytowy, składający się z Product_id oraz sub_investment_id. W ten sposób możemy jednoznacznie określić zamówienie – dany produkt może pojawić się tylko raz dla danej podinwestycji.

Daty rozpoczęcia oraz zakończenia inwestycji/pod inwestycji zostały zabezpieczone warunkiem - end_date > start_date.

3. Tworzenie bazy danych

Dane są przechowywane w bazie danych „company” w schemacie „company”. Postanowiłem utworzyć nowy schemat grupujący wszystkie tabele oraz określiłem uprawnienia dla użytkowników dla schematu. Tym samym zrezygnowałem z wykorzystania domyślnego schematu „public”, co wydaje się dobrą praktyką, ze względów bezpieczeństwa – tworząc nowego role/user domyślnie jest mu przyznany dostęp do schematu „public”. Tworząc własny schemat, użytkownik tworzący schemat musi nadać nowemu użytkownikowi odpowiednie uprawnienia, aby mógł korzystać z jego zawartości.

```
company=> \dt company.*
          List of relations
Schema | Name | Type | Owner
-----+-----+-----+-----
company | client | table | postgres
company | contact_person | table | postgres
company | investment | table | postgres
company | place_object | table | postgres
company | producer | table | postgres
company | product | table | postgres
company | product_order | table | postgres
company | sub_investment | table | postgres
company | supervisor | table | postgres
(9 rows)
```

UTWORZONE TABELE W BAZIE DANYCH ZGODNIE ZE SCHEMATEM ERD.

Nie zdecydowałem się na stworzenie kilku schematów, dla każdej grupy użytkowników, ponieważ w założeniu, każda grupa użytkowników ma mieć dostęp do wszystkich tabel. Ograniczenia dostępu zrealizowałem ustawiając odpowiednie ograniczenia na operacje w schemacie.

```
company=> SELECT * FROM company.client;
```

client_id	name	surname	company_name	phone	email
1	Beverie	Nevet	Graham, Hudson and Hettinger	123 557 7970	bnevet0@github.io
2	Kelvin	Dolder	Hahn Group	129 842 9118	kdolder1@oracle.com
3	Kaylil	Lotte	Lueilwitz Inc	819 330 4006	klotte2@mapquest.com
4	Reeta	Astman	Casper-Harber	322 893 4668	rastman3@pen.io
5	Noby	Theze	Johnston-Spencer	976 549 1871	ntheze4@ning.com
6	Maximilian	Herculeson	Turcotte-Daniel	411 442 5859	mherculeson5@1und1.de
7	Lilly	Billiard	Goldner, Parker and Dibbert	991 112 7980	lbilliard6@earthlink.net
8	Olly	Bilovsky	Hartmann LLC	266 957 4203	obilovsky7@homestead.com
9	Nicolea	Luker	Stark-Brown	595 190 5940	nluker8@goo.ne.jp
10	Grier	Dory	Altenwerth, Kozey and Johnson	948 913 1268	gdory9@ovh.net

(10 rows)

TABELA „CLIENT” WYPEŁNIONA DANYMI.

Constraint

Poniższe zdjęcia przedstawiają warunek CONSTARINT „product_type” z tabeli „product” oraz jego reakcje przy próbie dodania nowych rekordów do tabeli. Warunek wymaga, aby atrybut type_B był różny od atrybutu type_A (założono, że jednego produkt nie może zakwalifikować do dwóch typów produktu).

Co warto zauważyć, wspomniane atrybuty może przyjąć wartość NULL, co oznacza, że typ produktu nie musi zostać określony.

```
CREATE TABLE product (  
    product_id SERIAL NOT NULL PRIMARY KEY,  
    producer_id INTEGER NOT NULL,  
    name VARCHAR(100) NOT NULL,  
    price_net NUMERIC(8,2) NOT NULL,  
    type_A BOOLEAN,  
    type_B BOOLEAN,  
    CONSTRAINT product_type CHECK(type_B != type_A),  
    FOREIGN KEY (producer_id) REFERENCES producer(producer_id)  
);
```

```
company=# \i C:/Users/yoszimitsu/Desktop/product.sql  
INSERT 0 1  
psql:C:/Users/yoszimitsu/Desktop/product.sql:2: BŁĄD: nowy rekord dla relacji "product" narusz  
a ograniczenie sprawdzające "product_type"  
DETAIL: Niepoprawne ograniczenia wiersza (4, 2, Sambuca Cream, 296.60, f, f).  
psql:C:/Users/yoszimitsu/Desktop/product.sql:3: BŁĄD: nowy rekord dla relacji "product" narusz  
a ograniczenie sprawdzające "product_type"  
DETAIL: Niepoprawne ograniczenia wiersza (5, 3, Water - Perrier, 583.46, f, f).  
psql:C:/Users/yoszimitsu/Desktop/product.sql:4: BŁĄD: nowy rekord dla relacji "product" narusz  
a ograniczenie sprawdzające "product_type"  
DETAIL: Niepoprawne ograniczenia wiersza (6, 4, Longos - Greek Salad, 201.76, f, f).  
INSERT 0 1  
INSERT 0 1  
psql:C:/Users/yoszimitsu/Desktop/product.sql:7: BŁĄD: nowy rekord dla relacji "product" narusz  
a ograniczenie sprawdzające "product_type"  
DETAIL: Niepoprawne ograniczenia wiersza (9, 7, Lamb - Loin Chops, 744.25, t, t).  
INSERT 0 1  
INSERT 0 1  
psql:C:/Users/yoszimitsu/Desktop/product.sql:10: BŁĄD: nowy rekord dla relacji "product" narus  
za ograniczenie sprawdzające "product_type"  
DETAIL: Niepoprawne ograniczenia wiersza (12, 10, Wine - Two Oceans Sauvignon, 860.28, f, f).  
company=#
```

REAKCJA CONSTRAINT „PRODUCT_TYPE” PRZY WSTAWIANIU DANYCH DO TABELI.

4. Użytkownicy

W bazie danych utworzono 3 grupy użytkowników:

- Management
- Employee
- Intern

Założono, że użytkownik Management ma uprawnienia do wszystkich operacji na tabakach oraz ma możliwość dodawania nowych użytkowników oraz tabel.

Employee może wykonywać operacje odczytu, zapisu, edycji oraz usuwania danych ze wszystkich tabel.

Intern ma możliwość jedynie odczytu oraz zapisu danych w tabelach.

Uprawnienia przypisano analogicznie do hierarchii w firmie, chroniąc bazę danych przed błędami użytkowników „Intern”.

Do każdej grupy przypisano poniżej wymienione uprawnienia:

```
CREATE ROLE management;
ALTER ROLE management LOGIN PASSWORD 'management';
ALTER ROLE management CREATEDB;
ALTER ROLE management CREATEROLE;

CREATE ROLE employee;
ALTER ROLE employee LOGIN PASSWORD 'employee';

CREATE ROLE intern;
ALTER ROLE intern LOGIN PASSWORD 'intern';

-- --ACCESS DB
REVOKE CONNECT ON DATABASE company FROM PUBLIC;
GRANT CONNECT ON DATABASE company TO management, employee, intern;

-- --ACCESS SCHEMA
REVOKE ALL ON SCHEMA company FROM PUBLIC;
GRANT USAGE ON SCHEMA company TO management, employee, intern;

-- --ACCESS TABLES
REVOKE ALL ON ALL TABLES IN SCHEMA company FROM PUBLIC ;
GRANT SELECT, INSERT ON ALL TABLES IN SCHEMA company TO intern;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA company TO employee;
GRANT ALL ON ALL TABLES IN SCHEMA company TO management;

-- --ACCESS SEQUENCES (FROM POSTGRES 8.2)
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA company TO management, employee, intern;
```

PRZYPISYWANE UPRAWNIENIA DLA UŻYTKOWNIKÓW BAZY DANYCH.

Poniższe zdjęcia przedstawiają dostęp do bazy danych użytkownika z grupy 'Intern':

```
company=> SELECT current_user;
current_user
-----
intern
(1 row)

company=> DELETE FROM company.client WHERE client_id=1;
BŁĄD: permission denied for table client
company=> _
```

```
company=> SELECT current_user;
current_user
-----
intern
(1 row)

company=> UPDATE company.client SET name='ALEX' WHERE client_id=1;
BŁĄD: permission denied for table client
company=>
```

```
company=> SELECT current_user;
current_user
-----
intern
(1 row)

company=> SELECT * FROM company.client;
client_id | name | surname | company_name | phone | email
-----
1 | Beverie | Nevet | Graham, Hudson and Hettinger | 123 557 7970 | bnevet0@github.io
2 | Kelvin | Dolder | Hahn Group | 129 842 9118 | kdolder1@oracle.com
3 | Kaylil | Lotte | Lueilwitz Inc | 819 330 4006 | klotte2@mapquest.com
4 | Reeta | Astman | Casper-Harber | 322 893 4668 | rastman3@pen.io
5 | Noby | Theze | Johnston-Spencer | 976 549 1871 | ntheze4@ning.com
6 | Maximilian | Herculeson | Turcotte-Daniel | 411 442 5859 | mherculeson5@1und1.de
7 | Lilly | Billiard | Goldner, Parker and Dibbert | 991 112 7980 | lbilliard6@earthlink.net
8 | Olly | Bilovsky | Hartmann LLC | 266 957 4203 | obilovsky7@homestead.com
9 | Nicolea | Luker | Stark-Brown | 595 190 5940 | nluker8@goo.ne.jp
10 | Grier | Dory | Altenwerth, Kozey and Johnson | 948 913 1268 | gdory9@ovh.net
(10 rows)
```

```
company=> SELECT current_user;
current_user
-----
intern
(1 row)

company=> insert into company.client (name, surname, company_name, phone, email)
company-> values ('Beverie', 'Nebet', 'Graham, Hudson and Hettinger', '123 557 7970', 'bnevet0@github.io');
INSERT 0 1
company=> _
```

KONTROLA UPRAWNIEŃ DLA UŻYTKOWNIKA INTERN

Jak widać użytkownik Intern nie ma uprawnień do usuwania oraz edycji danych w tabelach.

5. Zapytania SQL

SELECT

```
    c.product_id,  
    c.sub_investment_id,  
    a.price_net,  
    c.quantity,  
    ROUND(a.price_net * c.quantity, 2) value_net  
FROM company.product_order c  
INNER JOIN company.product a  
ON c.product_id = a.product_id  
INNER JOIN company.producer b  
ON a.producer_id = b.producer_id  
ORDER BY c.product_id;
```

```
company=> SELECT  
company->  
company-> c.product_id,  
company-> c.sub_investment_id,  
company-> a.price_net,  
company-> c.quantity,  
company-> ROUND(a.price_net * c.quantity, 2) value_net  
company-> FROM company.product_order c  
company-> INNER JOIN company.product a  
company-> ON c.product_id = a.product_id  
company-> INNER JOIN company.producer b  
company-> ON a.producer_id = b.producer_id  
company-> ORDER BY c.product_id;  
product_id | sub_investment_id | price_net | quantity | value_net  
-----  
1 | 1 | 318.27 | 5 | 1591.35  
2 | 2 | 296.60 | 79 | 23431.40  
3 | 3 | 583.46 | 43 | 25088.78  
4 | 4 | 201.76 | 7 | 1412.32  
5 | 5 | 580.59 | 5 | 2902.95  
6 | 6 | 595.75 | 71 | 42298.25  
7 | 7 | 744.25 | 395 | 293978.75  
8 | 8 | 3.81 | 504 | 1920.24  
9 | 9 | 816.59 | 5 | 4082.95  
10 | 10 | 860.28 | 57 | 49035.96  
(10 rows)
```

PRZYKŁADOWE ZAPYTANIE PRZEDSTAWIAJĄCE WARTOŚĆ ZAMÓWIENIA DANEGO PRODUKTU Z PODINWESTYCJI.

```

SELECT
    c.product_id,
    c.sub_investment_id,
    a.price_net,
    b.type_b_discount,
    ROUND(a.price_net * (1 - b.type_b_discount), 2) after_discount,
    c.quantity,
    ROUND(ROUND(a.price_net * (1 - b.type_b_discount), 2) * c.quantity, 2) value_net
FROM company.product_order c
INNER JOIN company.product a
ON c.product_id = a.product_id
INNER JOIN company.producer b
ON a.producer_id = b.producer_id
ORDER BY c.product_id;

```

```

company=>
company=> SELECT
company-> c.product_id,
company-> c.sub_investment_id,
company-> a.price_net,
company-> b.type_b_discount,
company-> ROUND(a.price_net * (1 - b.type_b_discount), 2) after_discount,
company-> c.quantity,
company-> ROUND(ROUND(a.price_net * (1 - b.type_b_discount), 2) * c.quantity, 2) value_net
company-> FROM company.product_order c
company-> INNER JOIN company.product a
company-> ON c.product_id = a.product_id
company-> INNER JOIN company.producer b
company-> ON a.producer_id = b.producer_id
company-> ORDER BY c.product_id;

```

product_id	sub_investment_id	price_net	type_b_discount	after_discount	quantity	value_net
1	1	318.27	0.06	299.17	5	1495.85
2	2	296.60	0.29	210.59	79	16636.61
3	3	583.46	0.54	268.39	43	11540.77
4	4	201.76	0.87	26.23	7	183.61
5	5	580.59	0.57	249.65	5	1248.25
6	6	595.75	0.98	11.92	71	846.32
7	7	744.25	0.67	245.60	395	97012.00
8	8	3.81	0.13	3.31	504	1668.24
9	9	816.59	0.08	751.26	5	3756.30
10	10	860.28	0.34	567.78	57	32363.46

```

(10 rows)

company=> _

```

PRZYKŁADOWE ZAPYTANIE PRZEDSTAWIAJĄCE WARTOŚĆ ZAMÓWIENIA DANEGO PRODUKTU Z PODINWESTYCJI
 UWZGLĘDNIAJĄC ZNIŻKI NA DANY PRODUKT.

6. Napotkane problemy

- Client encoding:

```
postgres=# DROP TABLE investment;
ERROR:  character with byte sequence 0xc5 0x81 in encoding "UTF8" has no equivalent in encoding "WIN1252"
postgres=# SET CLIENT_ENCODING TO 'UTF8';
SET
postgres=# DROP TABLE investment;
BŁĄD: nie można usunąć tabela investment ponieważ inne obiekty zależą od niego
DETAIL:  ograniczenie sub_investment_investment_id_fkey na tabela sub_investment zależy od tabela investment
HINT:  Użyj DROP ... CASCADE aby usunąć wraz z obiektami zależnymi.
postgres=# DROP TABLE investment CASCADE;
UWAGA: kasowanie kaskadowe do ograniczenie sub_investment_investment_id_fkey na tabela sub_investment
DROP TABLE
postgres=#
```

Solution: Konieczna była zmiana typu kodowania klienta z WIN1252 na UTF-8 do poprawnej egzekucji komend w Postgres CLI (psql).

- Numerowanie produktów za pomocą SERIAL. Numery ID nie są przypisywane po kolei. Luki powstają, gdy np. wstawiać nowy rekord, dane nie przejdą walidacji. Mimo, że nie dodano nowego produktu, został zajęty dany numer klucza głównego.

```
company=# SELECT * FROM product;
 product_id | producer_id | name | price_net | type_a | type_b
-----|-----|-----|-----|-----|-----
3 | 1 | Pasta - Shells, Medium, Dry | 318.27 | f | t
7 | 5 | Table Cloth 54x72 Colour | 580.59 | f | t
8 | 6 | Chinese Foods - Chicken | 595.75 | t | f
10 | 8 | Sage Derby | 3.81 | f | t
11 | 9 | Cape Capensis - Fillet | 816.59 | t | f
13 | 1 | Pasta - Shells, Medium, Dry | 318.27 | f | t
14 | 2 | Sambuca Cream | 296.60 | f | t
15 | 3 | Water - Perrier | 583.46 | f | t
16 | 4 | Longos - Greek Salad | 201.76 | t | f
17 | 5 | Table Cloth 54x72 Colour | 580.59 | f | t
18 | 6 | Chinese Foods - Chicken | 595.75 | t | f
19 | 7 | Lamb - Loin Chops | 744.25 | t | t
20 | 8 | Sage Derby | 3.81 | f | t
21 | 9 | Cape Capensis - Fillet | 816.59 | t | f
22 | 10 | Wine - Two Oceans Sauvignon | 860.28 | t | t
(15 rows)
```

Solution: Nie rozwiązano

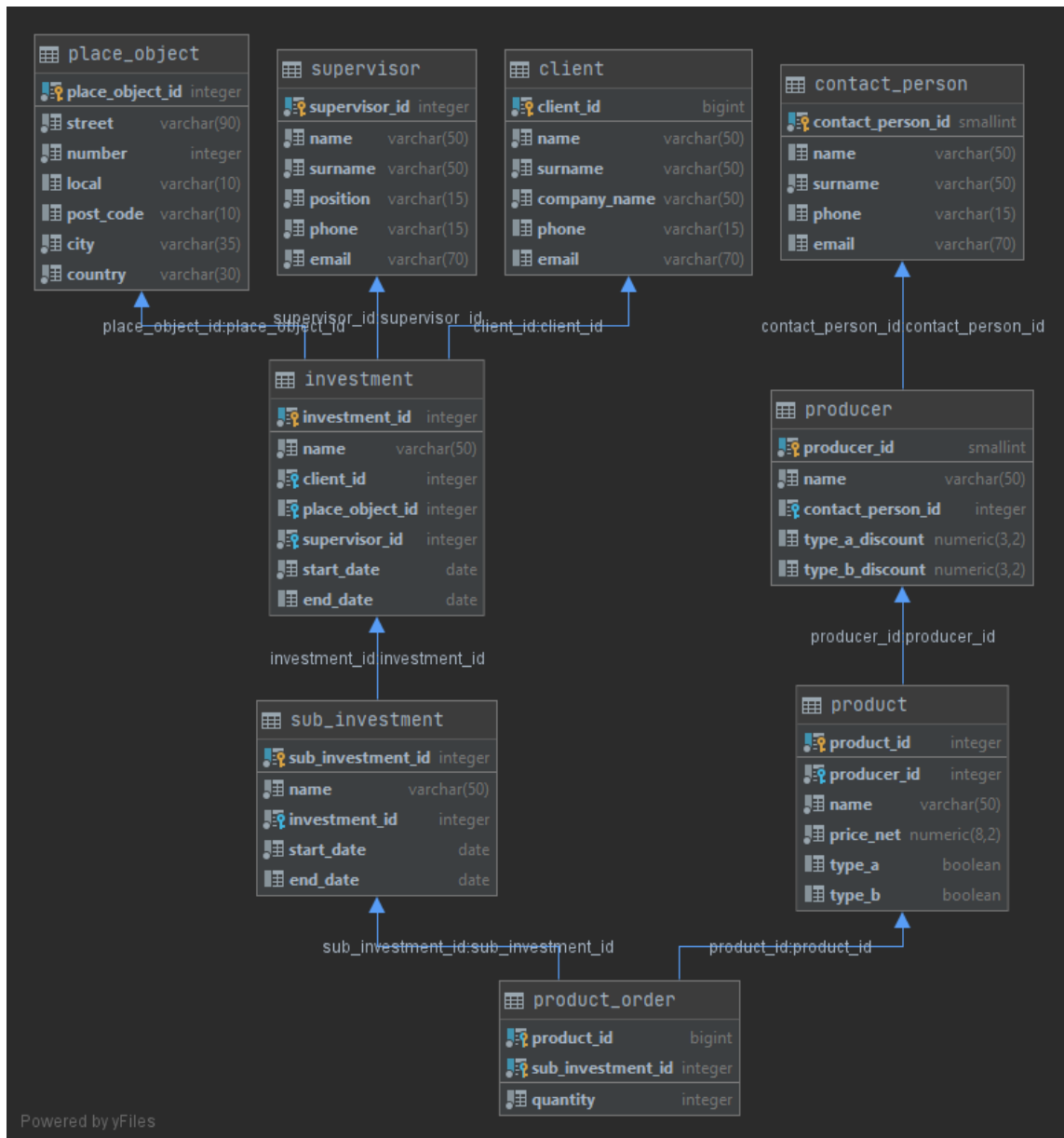
- ```
company=> \i C:/Users/yoszimitsu/Desktop/client.sql
psql:C:/Users/yoszimitsu/Desktop/client.sql:1: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:2: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:3: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:4: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:5: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:6: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:7: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:8: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:9: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
psql:C:/Users/yoszimitsu/Desktop/client.sql:10: BÄ?Ä,D: odmowa dostÄTpu do sekwencji client_client_id_seq
company=> \d
Did not find any relations
```

GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA company TO management, employee, intern;

[illegible]

## Poprawki Etap1

### Schemat ERD



Po etapie pierwszym dodano atrybut name do tabeli investment i sub\_investment. Dodatkowo zmieniono typy (wielkość VARCHAR) niektórych zmiennych jak: street, city. Zmiana poprawiła bazę pod względem walidacji danych, lecz nie wpłynęła na strukturę bazy danych.

Dodatkowo tabele uzupełniono danymi dopasowanymi do update'owanych tabel.

Wprowadzono dodatkowe ograniczenie na tabele producer

```
ALTER TABLE company.producer ADD CONSTRAINT unique_constraints UNIQUE (name);
```

## Etap2

### 1. Perspektywy

Utworzono perspektywę investment\_info, która przedstawia informacje o inwestycji w praktyczny sposób. Widok zawiera informacje z tabeli investment, opiekuna projektu (supervisor) oraz adres obiektu inwestycji, wywołany przez specjalnie utworzoną funkcję.

Przed stworzeniem widoku została dodana funkcja zwracająca adres obiektu z poszczególnych kolumn tabeli place\_object.

```
CREATE OR REPLACE FUNCTION company.address() RETURNS VARCHAR
LANGUAGE SQL
AS $$
 SELECT
 CONCAT('st. ', p.street, ' ', number, ' ', post_code, ' ', city, ' ', country)
 FROM company.place_object AS p;
$$;
```

PROCEDURA COMAPNY.ADDRESS() ZWRACAJĄCA PEŁEN ADRES OBIEKTU.

Kod perspektywy z użyciem funkcji address() przedstawiono poniżej:

```
CREATE OR REPLACE VIEW company.investment_info as
SELECT
 c.investment_id,
 c.name,
 k.company_name client,
 company.address_procedure() address,
 c.start_date,
 c.end_date,
 CONCAT(s.name, ' ', s.surname) supervisor
FROM company.investment c
INNER JOIN company.place_object p
ON c.place_object_id = p.place_object_id
INNER JOIN company.supervisor s
ON c.supervisor_id = s.supervisor_id
INNER JOIN company.client k
ON c.client_id = k.client_id
ORDER BY c.investment_id;
```

PERSPEKTYWA COMPANY.INVESTMENT\_INFO.

```
company=# SELECT * FROM company.investment_info;
```

| investment_id | name     | client                        | address                                              | start_date | end_date   | supervisor        |
|---------------|----------|-------------------------------|------------------------------------------------------|------------|------------|-------------------|
| 1             | invest1  | Graham, Hudson and Hettinger  | st. Thier 1 ,411 45 ,Mukařov ,Czech Republic         | 2018-11-28 | 2020-10-11 | Ogdon Kauschke    |
| 2             | invest2  | Hahn Group                    | st. Bowman 2 , ,Tarrafal ,Cape Verde                 | 2016-12-10 | 2020-10-12 | Chauncey Drewe    |
| 3             | invest3  | Luehlwitz Inc                 | st. Rieder 3 ,58884-000 ,Catolão do Rocha ,Brazil    | 2016-04-18 | 2020-10-10 | Jewel Dik         |
| 4             | invest4  | Casper-Harber                 | st. Gerald 4 ,93285 ,Saint-Denis ,France             | 2019-11-28 | 2020-10-12 | Cori Follacaro    |
| 5             | invest5  | Johnston-Spencer              | st. Morning 5 , ,Andros Town ,Bahamas                | 2016-07-26 | 2020-10-11 | Walliw Regler     |
| 6             | invest6  | Turcotte-Daniel               | st. Orin 6 ,57351 ,Kam-ŭlia ,Pakistan                | 2017-08-10 | 2020-10-12 | Stirling Williams |
| 7             | invest7  | Goldner, Parker and Dibbert   | st. Stone Corner 7 ,50770 ,-Grifw-ŭlia ,Pakistan     | 2017-03-05 | 2020-10-13 | Danella Valti     |
| 8             | invest8  | Hartmann LLC                  | st. Scott 8 ,97701 ,Ranua ,Finland                   | 2019-06-25 | 2020-10-12 | Katina Lubbock    |
| 9             | invest9  | Stark-Brown                   | st. Erie 9 , ,D-ŭrayy-ŭ ,Syria                       | 2018-01-08 | 2020-10-10 | Ladonna Matyasik  |
| 10            | invest10 | Altenwerth, Kozey and Johnson | st. Tomscot 10 , ,San Antonio del Monte ,El Salvador | 2017-03-14 | 2020-10-13 | Iormina Lodeke    |

(10 rows)

WYWOŁANIE PERSPEKTYWY COMPANY.INVESTMENT\_INFO.

Drugą perspektywą o nazwie product\_order\_value przedstawia zamówienia z danymi zawierającymi ceny produktów, ilość oraz wartość końcową.

```

CREATE VIEW company.product_orders_value as
SELECT
 c.sub_investment_id,
 s.name sub_investment_name,
 p.name product_name,
 a.price_net,
 (ROUND(a.price_net *
 (1 - CASE
 WHEN (COALESCE(a.type_a, false)) IS TRUE THEN
 b.type_a_discount
 WHEN (COALESCE(a.type_b, false)) IS TRUE THEN
 b.type_b_discount
 ELSE
 0
 END
),2)
) after_discount,
 c.quantity,
 ROUND(
 ROUND(a.price_net *
 (1 - CASE
 WHEN (COALESCE(a.type_a, false)) IS TRUE THEN
 b.type_a_discount
 WHEN (COALESCE(a.type_b, false)) IS TRUE THEN
 b.type_b_discount
 ELSE
 0
 END
),2) * c.quantity, 2) value_net
FROM company.product_order c
INNER JOIN company.product a
ON c.product_id = a.product_id
INNER JOIN company.sub_investment s
ON c.sub_investment_id = s.sub_investment_id
INNER JOIN company.product p
ON a.product_id = p.product_id
INNER JOIN company.producer b
ON a.producer_id = b.producer_id
ORDER BY c.product_id;

```

PERSPEKTYWA COMPANY.PRODUCT\_ORDER\_VALUE.

Podjąłem próbę wydzielenia powielanej części kodu do oddzielnej funkcji, lecz niestety nie udało mi się utworzyć tej funkcji w prawidłowy sposób. Więcej informacji w sekcji „Napotkane problemy”.

```
company=# SELECT * FROM company.product_orders_value;
```

| sub_investment_id | sub_investment_name | product_name                | price_net | after_discount | quantity | value_net |
|-------------------|---------------------|-----------------------------|-----------|----------------|----------|-----------|
| 2                 | sub_invest2         | Sambuca Cream               | 296.60    | 296.60         | 79       | 23431.40  |
| 3                 | sub_invest3         | Water - Perrier             | 583.46    | 268.39         | 43       | 11540.77  |
| 4                 | sub_invest4         | Longos - Greek Salad        | 201.76    | 135.18         | 7        | 946.26    |
| 5                 | sub_invest5         | Table Cloth 54x72 Colour    | 580.59    | 249.65         | 5        | 1248.25   |
| 6                 | sub_invest6         | Chinese Foods - Chicken     | 595.75    | 142.98         | 71       | 10151.58  |
| 7                 | sub_invest7         | Lamb - Loin Chops           | 744.25    | 334.91         | 395      | 132289.45 |
| 8                 | sub_invest8         | Sage Derby                  | 3.81      | 3.31           | 504      | 1668.24   |
| 9                 | sub_invest9         | Cape Capensis - Fillet      | 816.59    | 293.97         | 5        | 1469.85   |
| 10                | sub_invest10        | Wine - Two Oceans Sauvignon | 860.28    | 860.28         | 57       | 49035.96  |

(9 rows)

WYWOŁANIE PERSPEKTYWY COMPANY.PRODUCT\_ORDER\_VALUE.

## 2. Indeksy

Indeksy przyspieszają przetwarzanie zapytań SQL i powinny być nakładane na kolumny, w których wartości często się powtarzają lub są często wczytywane. Klucze główne tabel są automatycznie indeksowane.

Postanowiono stworzyć 3 indeksy:

- Dla wartości name i surname tabeli supervisor
- Dla wartości name tabeli investment
- Dla wartości name tabeli sub\_investment

Wartości name z tabel investment i sub\_investment są wykorzystywane w widokach i są ważnymi kolumnami z punktu widzenia logiki biznesowej.

Indeks supervisor\_idx został dodatkowo oznaczony jako UNIQUE co oznacza, że wartości „imię” i „nazwisko” nie mogą być duplikowane. Indeks UNIQUE znacząco przyspiesza przetwarzanie wartości.

```
company=# CREATE UNIQUE INDEX supervisor_idx ON supervisor (name, surname);
BŁĄD: relacja "supervisor" nie istnieje
company=# CREATE UNIQUE INDEX supervisor_idx ON company.supervisor (name, surname);
CREATE INDEX
company=# CREATE INDEX invest_name_idx ON company.investment (name);
CREATE INDEX
company=# CREATE INDEX sub_invest_name_idx ON company.sub_investment (name);
CREATE INDEX
```

|         |                |                     |                                                                                      |
|---------|----------------|---------------------|--------------------------------------------------------------------------------------|
| company | supervisor     | supervisor_idx      | CREATE UNIQUE INDEX supervisor_idx ON company.supervisor USING btree (name, surname) |
| company | investment     | invest_name_idx     | CREATE INDEX invest_name_idx ON company.investment USING btree (name)                |
| company | sub_investment | sub_invest_name_idx | CREATE INDEX sub_invest_name_idx ON company.sub_investment USING btree (name)        |

(129 rows)

INDEKSY UTWORZONE W BAZIE DANYCH COMPANY.

### 3. Procedury, funkcje, wyzwalacze

#### Wyjaśnienie różnic:

Funkcja musi zwracać wartość a w procedurze składowanej jest opcjonalna (procedura może zwrócić zero lub n wartości). Funkcje mogą mieć tylko parametry wejściowe a procedury mogą mieć parametry wejściowe/wyjściowe. Funkcje mogą być wywoływane z procedury, ale procedura nie może być wywołana z funkcji.

Jedną ze stworzonych funkcji jest funkcja address(), która zwraca adres w postaci łańcucha znaków z poszczególnych kolumn tabeli place\_object.

```
company=# CREATE OR REPLACE FUNCTION company.address(index INT) RETURNS VARCHAR
company=# LANGUAGE SQL
company=# AS $$
company$# SELECT
company$# CONCAT('st. ', p.street, ' ', number, ' ', post_code, ' ', city, ' ', country)
company$# FROM company.place_object AS p
company$# WHERE p.place_object_id = index;
company$# $$;
CREATE FUNCTION
company=# SELECT company.address(2);
address

st. Bowman 2 , ,Tarrafal ,Cape Verde
(1 row)

company=# SELECT company.address(4);
address

st. Gerald 4 ,93285 ,Saint-Denis ,France
(1 row)
```

FUNKCJA COMAPNY.ADDRESS(ID) ZWRACAJĄCA ADDRESS OBIEKTU Z REKORDU ID.

Utworzony procedurę new\_employee, która pozwala na wprowadzenie nowych pracowników do tabeli supervisor.

```
company=# CALL company.new_employee('daniel', 'michrowski', 'test', 'test', 'test@test.pl');
CALL
company=# SELECT * FROM company.supervisor;
 supervisor_id | name | surname | position | phone | email
-----+-----+-----+-----+-----+-----
1 | Ogdon | Kauschke | engineer | 228 911 2936 | okauschke0@wired.com
2 | Chauncey | Drewe | engineer | 474 860 1874 | cdrewe1@independent.co.uk
3 | Jewel | Dik | engineer | 974 941 9143 | jdik2@google.cn
4 | Cori | Follacaro | engineer | 752 997 3851 | cfollacaro3@a8.net
5 | Walliw | Regler | engineer | 789 582 4255 | wregler4@nydailynews.com
6 | Stirling | Williams | engineer | 925 222 8698 | swilliams5@mashable.com
7 | Danella | Valti | engineer | 317 883 0721 | dvalti6@ucsd.edu
8 | Katina | Lubbock | engineer | 857 635 9630 | klubbock7@lulu.com
9 | Ladonna | Matyasik | engineer | 146 450 5182 | lmatyasik8@chronoengine.com
10 | Iormina | Lodeke | engineer | 365 179 3664 | ilodeke9@google.co.jp
12 | daniel2 | michrowski2 | test2 | test2 | test2@test.pl
13 | daniel | michrowski | test | test | test@test.pl
(12 rows)
```

PROCEDURA COMPANY.NEW\_EMPLOYEE POZWALA NA ZAPIS NOWYCH PRACOWNIKÓW DO TABELI SUPERVISOR.



Kolejną procedurą jest new\_product\_and\_producer pozwalająca na wprowadzenie produktu od nowego producenta. W procedurze zastosowano obsługę błędów. Poniższe zdjęcia przedstawiają kod procedury:

```
CREATE OR REPLACE PROCEDURE company.new_product_and_producer(
product_name VARCHAR(50), price_net NUMERIC(8,2), type_a BOOLEAN, type_b BOOLEAN,
producer_name VARCHAR(50), type_a_discount NUMERIC(3,2), type_b_discount NUMERIC(3,2))
AS $$
declare
 v_state TEXT;
 v_msg TEXT;
 v_detail TEXT;
 v_hint TEXT;
 v_context TEXT;
begin
 INSERT INTO company.producer(
 name,
 type_a_discount,
 type_b_discount)
 VALUES(
 producer_name,
 type_a_discount,
 type_b_discount);

 INSERT INTO company.product(
 producer_id,
 name,
 price_net,
 type_a,
 type_b)
 VALUES(
 (SELECT producer_id FROM company.producer WHERE name = producer_name),
 product_name,
 price_net,
 type_a,
 type_b);

exception when others then
 get stacked diagnostics
 v_state = returned_sqlstate,
 v_msg = message_text,
 v_detail = pg_exception_detail,
 v_hint = pg_exception_hint,
 v_context = pg_exception_context;

 raise notice E'Got exception:
state : %
message: %
detail : %
hint : %
context: %', v_state, v_msg, v_detail, v_hint, v_context;

 raise notice E'Got exception:
SQLSTATE: %
SQLERRM: %', SQLSTATE, SQLERRM;

 raise notice '%', message_text; |

end;
$$ LANGUAGE plpgsql;
```

KOD PROCEDUR Z OBSŁUGĄ BŁĘDÓW.

Procedura zwraca szczegółowe informacje o błędzie. Komenda “exception when others then” obsługuje występujące błędy oraz zarządza transakcjami. Jeśli w czasie przetwarzania procedury wystąpi błąd, transakcje zostaną cofnięte (rollback). Jeśli procedura przebiegnie pomyślnie transakcje są zatwierdzone (commit).

Poniżej przedstawiono wywołania procedury new\_product\_and\_producer:

```
company=# CALL company.new_product_and_producer('soap', 3.45, true, false, 'Dove', 0.25, 0.0);
CALL
company=# SELECT * FROM company.product;
```

| product_id | producer_id | name                        | price_net | type_a | type_b |
|------------|-------------|-----------------------------|-----------|--------|--------|
| 1          | 1           | Pasta - Shells, Medium, Dry | 318.27    | f      | t      |
| 2          | 2           | Sambuca Cream               | 296.60    | f      |        |
| 3          | 3           | Water - Perrier             | 583.46    | f      | t      |
| 4          | 4           | Longos - Greek Salad        | 201.76    | t      | f      |
| 5          | 5           | Table Cloth 54x72 Colour    | 580.59    | f      | t      |
| 6          | 6           | Chinese Foods - Chicken     | 595.75    | t      | f      |
| 7          | 7           | Lamb - Loin Chops           | 744.25    | t      |        |
| 8          | 8           | Sage Derby                  | 3.81      | f      | t      |
| 9          | 9           | Cape Capensis - Fillet      | 816.59    | t      | f      |
| 10         | 10          | Wine - Two Oceans Sauvignon | 860.28    |        |        |
| 13         | 15          | soap                        | 3.45      | t      | f      |

(11 rows)

```
company=# SELECT * FROM company.producer;
```

| producer_id | name         | contact_person_id | type_a_discount | type_b_discount |
|-------------|--------------|-------------------|-----------------|-----------------|
| 1           | Agimba       | 1                 | 0.45            | 0.06            |
| 2           | Brightbean   | 2                 | 0.18            | 0.29            |
| 3           | Kazu         | 3                 | 0.07            | 0.54            |
| 4           | LiveZ        | 4                 | 0.33            | 0.87            |
| 5           | Roomm        | 5                 | 0.35            | 0.57            |
| 6           | Skipstorm    | 6                 | 0.76            | 0.98            |
| 7           | Thoughtworks | 7                 | 0.55            | 0.67            |
| 8           | Dabfeed      | 8                 | 0.42            | 0.13            |
| 9           | Tazz         | 9                 | 0.64            | 0.08            |
| 10          | Divanoodle   | 10                | 0.93            | 0.34            |
| 15          | Dove         |                   | 0.25            | 0.00            |

(11 rows)

WYWOŁANIE PROCEDURY Z POZYTYWNYM REZULTATEM.

Poniżej przedstawiono obsługę błędów. W tym przypadku błąd dotyczył próby wstawienia nazwy producenta, która istniała już w tabeli.

```

company=# CALL company.new_product_and_producer('soap', 3.45, true, false, 'Dove', 0.25, 0.0);
UNWAGA: Got exception:
 state : 23505
 message: podwójna wartość klucza narusza ograniczenie unikalności "unique_constraints"
 detail : Klucz (name)=(Dove) już istnieje.
 hint :
 context: wyrażenie SQL "INSERT INTO company.producer(
 name,
 type_a_discount,
 type_b_discount)
 VALUES(
 producer_name,
 type_a_discount,
 type_b_discount)"
funkcja PL/pgSQL company.new_product_and_producer(character varying,numeric,boolean,boolean,character varying,
UNWAGA: Got exception:
 SQLSTATE: 23505
 SQLERRM: podwójna wartość klucza narusza ograniczenie unikalności "unique_constraints"
BŁĄD: kolumna "message_text" nie istnieje
LINE 1: SELECT message_text
 ^
QUERY: SELECT message_text
CONTEXT: funkcja PL/pgSQL company.new_product_and_producer(character varying,numeric,boolean,boolean,character
company=#

```

ZWRÓCENIE BŁĘDU PRZY WYWOŁANIU PROCEDURY NEW\_PRODUCT\_AND\_PRODUCER.

Poniżej przedstawiono obsługę błędów związaną z wprowadzeniem nowego produktu, który przypisano jednocześnie do grupy a oraz b (ograniczenie product\_type sprawdza czy produkt nie jest przypisany do obu grup). W przykładzie można zauważyć, że pomimo że, błąd wystąpił przy wstawianiu danych do drugiej tabeli (w procedurze dane są wstawiane do dwóch tabel), dane które zostały pomyślnie wstawione o pierwszej tabeli zostały wycofane. Wywołanie SELECT \* FROM company.producer potwierdza, że producent „Nivea” nie został dodany.

```

company=# CALL company.new_product_and_producer('soap', 3.45, true, true, 'Nivea', 0.25, 0.0);
UNWAGA: Got exception:
 state : 23514
 message: nowy rekord dla relacji "product" narusza ograniczenie sprawdzające "product_type"
 detail : Niepoprawne ograniczenia wiersza (14, 18, soap, 3.45, t, t).
 hint :
 context: wyrażenie SQL "INSERT INTO company.product(
 producer_id,
 name,
 price_net,
 type_a,
 type_b)
 VALUES(
 (SELECT producer_id FROM company.producer WHERE name = producer_name),
 product_name,
 price_net,
 type_a,
 type_b)"
funkcja PL/pgSQL company.new_product_and_producer(character varying,numeric,boolean,boolean,character varying,numeric,numeric), wiersz 19 w wyrażeniu SQL
UNWAGA: Got exception:
 SQLSTATE: 23514
 SQLERRM: nowy rekord dla relacji "product" narusza ograniczenie sprawdzające "product_type"
BŁĄD: kolumna "message_text" nie istnieje
LINE 1: SELECT message_text
 ^
QUERY: SELECT message_text
CONTEXT: funkcja PL/pgSQL company.new_product_and_producer(character varying,numeric,boolean,boolean,character varying,numeric,numeric), wiersz 52 w RAISE
company=# SELECT * FROM company.producer;
 producer_id | name | contact_person_id | type_a_discount | type_b_discount

1 | Agimba | 1 | 0.45 | 0.06
2 | Brightbean | 2 | 0.18 | 0.29
3 | Kazu | 3 | 0.07 | 0.54
4 | LiveZ | 4 | 0.33 | 0.87
5 | Roomm | 5 | 0.35 | 0.57
6 | Skipstorm | 6 | 0.76 | 0.98
7 | Thoughtworks | 7 | 0.55 | 0.67
8 | Dabfeed | 8 | 0.42 | 0.13
9 | Tazz | 9 | 0.64 | 0.08
10 | Divanoodle | 10 | 0.93 | 0.34
15 | Dove | | 0.25 | 0.00
(11 rows)

company=#

```

TRANSAKCA Z TABELI COMPANY.PRODUCER ZOSTAŁA COFNIĘTA PO POJAWIENIU SIĘ BŁĘDU.

#### 4. Wyzwalacz

Wyzwalacze pozwalają na zautomatyzowanie pewnych poleceń w przypadku wystąpienia określonego działania w bazie danych.

Poniżej przedstawiono trigger, który usuwa zależności sub\_investment przed próbą usunięcia rekordu z investment. Zapobiega to przed pojawieniem się błędu, o powiązanych elementach w innych tabelach. Funkcja, która jest wyzwalana przez trigger, wykonuje polecenie równoznaczne z DELETE ... CASCADE.

```
CREATE OR REPLACE FUNCTION company.clear_sub_investment() RETURNS TRIGGER AS $$
BEGIN
 DELETE FROM company.product_order WHERE sub_investment_id = (
 SELECT sub_investment_id FROM company.sub_investment WHERE investment_id = OLD.investment_id);
 DELETE FROM company.sub_investment WHERE investment_id = OLD.investment_id;
 RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER clear_sub_investment BEFORE DELETE ON company.investment
FOR EACH ROW EXECUTE PROCEDURE company.clear_sub_investment();
```

WYZWALACZ USUWAJĄCY ODNIESIENIA PRZY USUWANIU REKORDU Z TABELI COMPANY.INVESTMENT.

| company=# SELECT * FROM company.investment;     |                   |               |                 |               |            |            |
|-------------------------------------------------|-------------------|---------------|-----------------|---------------|------------|------------|
| investment_id                                   | name              | client_id     | place_object_id | supervisor_id | start_date | end_date   |
| 1                                               | invest1           | 1             | 1               | 1             | 2018-11-28 | 2020-10-11 |
| 2                                               | invest2           | 2             | 2               | 2             | 2016-12-10 | 2020-10-12 |
| 3                                               | invest3           | 3             | 3               | 3             | 2016-04-18 | 2020-10-10 |
| 4                                               | invest4           | 4             | 4               | 4             | 2019-11-28 | 2020-10-12 |
| 5                                               | invest5           | 5             | 5               | 5             | 2016-07-26 | 2020-10-11 |
| 6                                               | invest6           | 6             | 6               | 6             | 2017-08-10 | 2020-10-12 |
| 7                                               | invest7           | 7             | 7               | 7             | 2017-03-05 | 2020-10-13 |
| 8                                               | invest8           | 8             | 8               | 8             | 2019-06-25 | 2020-10-12 |
| 9                                               | invest9           | 9             | 9               | 9             | 2018-01-08 | 2020-10-10 |
| 10                                              | invest10          | 10            | 10              | 10            | 2017-03-14 | 2020-10-13 |
| (10 rows)                                       |                   |               |                 |               |            |            |
| company=# SELECT * FROM company.sub_investment; |                   |               |                 |               |            |            |
| sub_investment_id                               | name              | investment_id | start_date      | end_date      |            |            |
| 1                                               | sub_invest1       | 1             | 2017-02-16      | 2019-09-05    |            |            |
| 2                                               | sub_invest2       | 2             | 2017-12-20      | 2019-09-27    |            |            |
| 3                                               | sub_invest3       | 3             | 2017-10-18      | 2019-11-09    |            |            |
| 4                                               | sub_invest4       | 4             | 2017-04-05      | 2020-03-17    |            |            |
| 5                                               | sub_invest5       | 5             | 2017-11-14      | 2020-10-13    |            |            |
| 6                                               | sub_invest6       | 6             | 2017-10-14      | 2019-01-07    |            |            |
| 7                                               | sub_invest7       | 7             | 2017-12-26      | 2019-11-17    |            |            |
| 8                                               | sub_invest8       | 8             | 2017-12-25      | 2019-12-18    |            |            |
| 9                                               | sub_invest9       | 9             | 2017-02-27      | 2019-12-10    |            |            |
| 10                                              | sub_invest10      | 10            | 2017-02-05      | 2019-11-10    |            |            |
| (10 rows)                                       |                   |               |                 |               |            |            |
| company=# SELECT * FROM company.product_order;  |                   |               |                 |               |            |            |
| product_id                                      | sub_investment_id | quantity      |                 |               |            |            |
| 1                                               | 1                 | 5             |                 |               |            |            |
| 2                                               | 2                 | 79            |                 |               |            |            |
| 3                                               | 3                 | 43            |                 |               |            |            |
| 4                                               | 4                 | 7             |                 |               |            |            |
| 5                                               | 5                 | 5             |                 |               |            |            |
| 6                                               | 6                 | 71            |                 |               |            |            |
| 7                                               | 7                 | 395           |                 |               |            |            |
| 8                                               | 8                 | 504           |                 |               |            |            |
| 9                                               | 9                 | 5             |                 |               |            |            |
| 10                                              | 10                | 57            |                 |               |            |            |
| (10 rows)                                       |                   |               |                 |               |            |            |

STAN TABEL PRZED USUNIĘCIEM REKORDU Z INVESTMENT.

```
company=# DELETE FROM company.investment WHERE investment_id = 1;
DELETE 1
company=# SELECT * FROM company.investment;
 investment_id | name | client_id | place_object_id | supervisor_id | start_date | end_date
-----+-----+-----+-----+-----+-----+-----
 2 | invest2 | 2 | 2 | 2 | 2016-12-10 | 2020-10-12
 3 | invest3 | 3 | 3 | 3 | 2016-04-18 | 2020-10-10
 4 | invest4 | 4 | 4 | 4 | 2019-11-28 | 2020-10-12
 5 | invest5 | 5 | 5 | 5 | 2016-07-26 | 2020-10-11
 6 | invest6 | 6 | 6 | 6 | 2017-08-10 | 2020-10-12
 7 | invest7 | 7 | 7 | 7 | 2017-03-05 | 2020-10-13
 8 | invest8 | 8 | 8 | 8 | 2019-06-25 | 2020-10-12
 9 | invest9 | 9 | 9 | 9 | 2018-01-08 | 2020-10-10
 10 | invest10 | 10 | 10 | 10 | 2017-03-14 | 2020-10-13
(9 rows)

company=# SELECT * FROM company.sub_investment;
 sub_investment_id | name | investment_id | start_date | end_date
-----+-----+-----+-----+-----
 2 | sub_invest2 | 2 | 2017-12-20 | 2019-09-27
 3 | sub_invest3 | 3 | 2017-10-18 | 2019-11-09
 4 | sub_invest4 | 4 | 2017-04-05 | 2020-03-17
 5 | sub_invest5 | 5 | 2017-11-14 | 2020-10-13
 6 | sub_invest6 | 6 | 2017-10-14 | 2019-01-07
 7 | sub_invest7 | 7 | 2017-12-26 | 2019-11-17
 8 | sub_invest8 | 8 | 2017-12-25 | 2019-12-18
 9 | sub_invest9 | 9 | 2017-02-27 | 2019-12-10
 10 | sub_invest10 | 10 | 2017-02-05 | 2019-11-10
(9 rows)

company=# SELECT * FROM company.product_order;
 product_id | sub_investment_id | quantity
-----+-----+-----
 2 | 2 | 79
 3 | 3 | 43
 4 | 4 | 7
 5 | 5 | 5
 6 | 6 | 71
 7 | 7 | 395
 8 | 8 | 504
 9 | 9 | 5
 10 | 10 | 57
(9 rows)
```

ZADZIAŁANIE TRIGGERA. SUB\_INVESTMENT, PODPIĘTE POD USUNIĘTY REKORD Z INVESTMENT, ZOSTAŁY USUNIĘTE Z TABELI SUB\_INVESTMENT ORAZ PRODUCT\_ORDER.

Trigger wykorzystano również do zapisu logów odnośnie do zmiany cen produktów. Utworzono nową tabelę do zapisu logów.

```
CREATE TABLE company.log_price_update(
 log_id BIGSERIAL NOT NULL,
 log_date TIMESTAMP NOT NULL,
 product_id INTEGER NOT NULL,
 product_name VARCHAR(50) NOT NULL,
 old_price NUMERIC(8,2) NOT NULL,
 new_price NUMERIC(8,2) NOT NULL,
 product_producer VARCHAR(50) NOT NULL);
```

NOWA TABELA LOG\_PRICE\_UPDATE DO ZAPISU ZMIAN CEN PRODUKTÓW.

Tabele będzie zawierać date i czas zmiany ceny produktu (log\_date). W takim przypadku warto zmienić strefę czasową na GMT. Jest to dobra praktyka dotycząca zapisu dat i godzin patrząc pod względem aplikacji działających na poziomie globalnym.

```
company=# SET TIMEZONE='GMT';
SET
company=# SHOW TIMEZONE;
 TimeZone

 GMT
(1 row)

company=#
```

ZAMIANA STREFY CZASOWEJ NA GMT.

```
CREATE OR REPLACE FUNCTION company.log_price_update() RETURNS TRIGGER AS $$
BEGIN
 INSERT INTO company.log_price_update (log_date, product_id, product_name, old_price, new_price, product_producer)
 VALUES (NOW(), OLD.product_id, OLD.name, OLD.price_net, NEW.price_net, OLD.producer_id);
 RETURN OLD;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_price_update_trigger AFTER UPDATE ON company.product
FOR EACH ROW EXECUTE PROCEDURE company.log_price_update();
```

FUNKCJA COMPANY.LOG\_PRICE\_UPDATE WYWOLOWANA PRZY ZMIANIE CEN PRODUKTÓW.

Poniżej przedstawiono rezultat utworzonego triggera. W tabeli log\_price\_update zapisaliśmy informacje o zmianie ceny produktu.

```
company=# UPDATE company.product SET price_net = 33.45 WHERE product_id = 1;
UPDATE 1
company=# SELECT * FROM company.log_price_update;
 log_id | log_date | product_id | product_name | old_price | new_price | product_producer
-----+-----+-----+-----+-----+-----+-----
 5 | 2020-11-22 11:41:26.191782 | 1 | Pasta - Shells, Medium, Dry | 3.45 | 33.45 | 1
(1 row)

company=#
```

ZAPIS LOGÓW PRZY ZMIANIE CEN PRODUKTÓW.

## 5. Automatyzacja działania

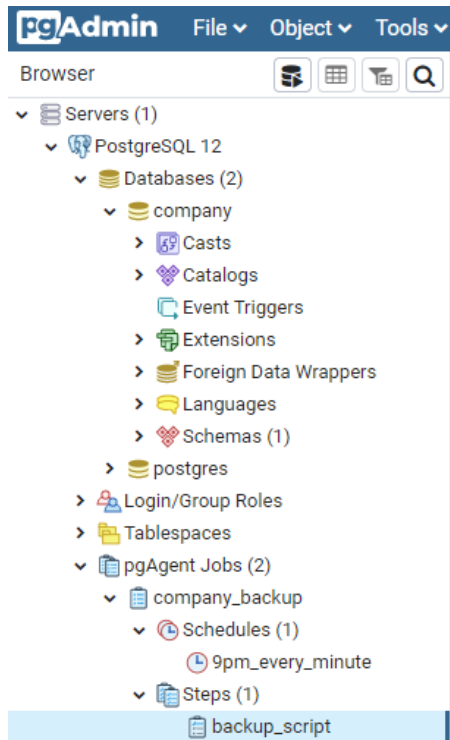
W ramach automatyzacji działania w bazie danych, wykonano automatyczny backup. Do wykonaniu backup wykorzystano mechanizm z bazy danych PostgreSQL pg\_dump.

Pierwszym krokiem jest napisanie komendy obejmującej maszynę, na której ma zostać wykonana procedura (localhost), użytkownika (management) oraz ścieżkę do zapisu plików backup.

```
cd C:\\Users
pg_dump -h "localhost" -U "management" -f "C:\\Users\\yoszimitsu\\Documents\\it_projects\\Company_DB\\DB\\backup\\company_backup
```

SKRYPT TWORZĄCY BACKUP BAZY DANYCH COMPANYY.

Po przygotowaniu odpowiedniego skryptu i zapisaniu go w formacie .bat można przejść do automatyzacji procesu. Do tego celu można wykorzystać pgAgent ze środowiska PostgreSQL lub Task Scheduler ze środowiska Windows.



PANEL PGADMIN Z ZAKŁADKĄ PGAGENT JOBS

Przy konfiguracji Job’a można wpisać komendę SQL lub odnieść się do pliku z systemu. W tym przypadku wskazano na wcześniej przygotowany skrypt.

backup\_script

General

Code

SQL

1

C:\Users\yoszimitsu\Documents\it\_projects\Company\_DB\DB\company\_backup\_script.bat

KONFIGURACJA JOB’A.

Jon będzie się wykonywał w godzinach określonych w Schedules.

9pm\_every\_minute

General

Repeat

Exceptions

SQL

Schedules are specified using a **cron-style** format.

For each selected time or date element, the schedule will execute.

e.g. To execute at 5 minutes past every hour, simply select '05' in the Minutes list box.

Values from more than one field may be specified in order to further control the schedule.

e.g. To execute at 12:05 and 14:05 every Monday and Thursday, you would click minute 05, hours 12 and 14, and weekdays Monday and Thursday.

For additional flexibility, the Month Days check list includes an extra Last Day option. This matches the last day of the month, whether it happens to be the 28th, 29th, 30th or 31st.

Days

Week Days

x Sunday

x Monday

x Tuesday

x Wednesday

x Thursday

x Friday

x Saturday

x

Month Days

Select the month days...

Months

Select the months...

Times

Hours

x 21

x

Minutes

x 00

x 01

x 02

x 03

x 04

x 05

x 06

x 07

x 08

x 09

x 10

x 11

x 12

x 13

x 14

x 15

x 16

x 17

x 18

x 19

x 20

x 21

x 22

x 23

x 24

x 25

x 26

x 27

x 28

x 29

x 30

x 31

x 32

x 33

x 34

x 35

x 36

x 37

x 38

x 39

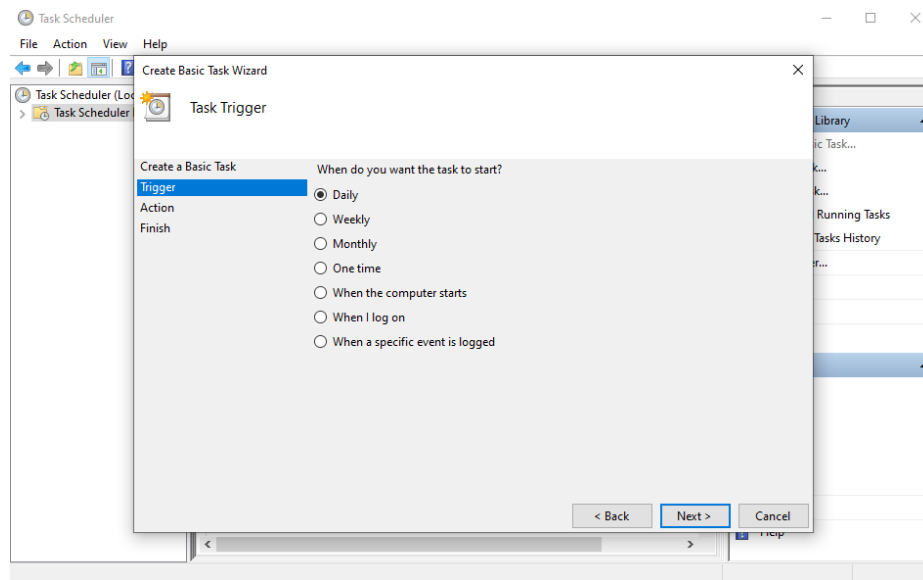
x 40

x 41

KONFIGURACJA SCHEDULES.



Również do tego celu można wykorzystać Task Scheduler z Windows.



KONFIGURACJA TASK SCHEDULER.

Niezależnie od wyboru narzędzia do automatyzacji, częścią wykonawczą jest skrypt zawierający narzędzie `pg_dump` z środowiska PostgreSQL.

## 6. Napotkane problemy

Podjąłem próbę stworzenia funkcji zwracającej zniżki na produkt w zależności od typu produktu (a lub b). Niestety `SELECT type_a FROM company.product` zwraca całą kolumnę wartości i nie może być przyrównana do `true`. Wartości musiałby być sprawdzane po kolei, następnie musiałyby być z nich stworzona cała kolumna i dopiero przekazana do widoku `product_order_value`, gdzie miała zastąpić zwiłokrotniony kod.

```
CREATE OR REPLACE FUNCTION company.product_type_disc() RETURNS TABLE (discount NUMERIC(3,2))
LANGUAGE plpgsql
AS $$
BEGIN
 CASE
 WHEN (COALESCE((SELECT type_a FROM company.product), false)) IS TRUE THEN
 SELECT type_a_discount FROM company.producer;
 WHEN (COALESCE((SELECT type_b FROM company.product), false)) IS TRUE THEN
 SELECT type_b_discount FROM company.producer;
 ELSE
 -- Empty result set for other types
 END CASE;
END
$$;
```

## Poprawki Etap2

Po drugim etapie dodano indeksy na start\_date i end\_date dla investment i sub\_investment. Dаты mogą być zmiennymi często używanymi do filtracji inwestycji patrząc ze strony biznesowej. Dodanie indeksów może znacząco przyspieszyć takie operacje filtrowania.

```
CREATE INDEX sub_investment_start_date ON company.sub_investment (start_date);
CREATE INDEX sub_investment_end_date ON company.sub_investment (end_date);

CREATE INDEX investment_start_date ON company.investment (start_date);
CREATE INDEX investment_end_date ON company.investment (end_date);
```

Poprawiono również błąd przy funkcji z obsługą błędów. Błąd był spowodowany złym przypisaniem zmiennej.

|                                                                                                                    |                                                                              |                                                                                                                                            |
|--------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>raise notice E'Got exception: state : % message: % detail : % hint : % context: %, v_state, v_msg, v_de</pre> | <pre>69 69 70 70 71 71 72 72 73 73 74 74 75 75 76 76 77 77 78 78 79 79</pre> | <pre>raise notice E'Got exception: state : % message: % detail : % hint : % context: %, v_state, v_msg, v_detail, v_hint, v_context;</pre> |
| <pre>raise notice E'Got exception: SQLSTATE: % SQLERRM: %, SQLSTATE, SQLERRM;</pre>                                | <pre>76 76 77 77 78 78 79 79</pre>                                           | <pre>raise notice E'Got exception: SQLSTATE: % SQLERRM: %, SQLSTATE, SQLERRM;</pre>                                                        |
| <pre>raise notice '%', message_text;</pre>                                                                         | <pre>80 80</pre>                                                             | <pre>raise notice '%', v_msg;</pre>                                                                                                        |
| <pre>end;</pre>                                                                                                    | <pre>81 81 82 82</pre>                                                       | <pre>end;</pre>                                                                                                                            |
| <pre>\$\$ LANGUAGE plpgsql;</pre>                                                                                  | <pre>83 83</pre>                                                             | <pre>\$\$ LANGUAGE plpgsql;</pre>                                                                                                          |

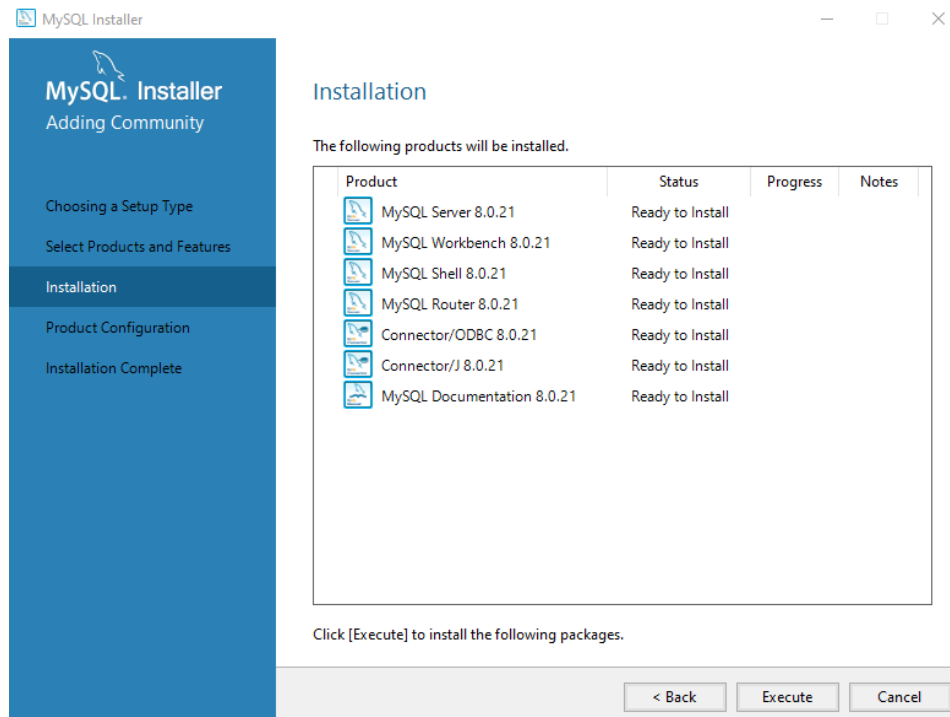
Po poprawce obsługa błędów przebiega prawidłowo.

```
company=> CALL company.new_product_and_producer('soap', 3.45, true, true, 'Dove', 0.25, 0.0);
UWAGA: Got exception:
state : 23505
message: podwójna wartość klucza narusza ograniczenie unikalności "unique_constraints"
detail : Klucz (name)=(Dove) już istnieje.
hint :
context: wyrażenie SQL "INSERT INTO company.producer(
name,
type_a_discount,
type_b_discount)
VALUES(
producer_name,
type_a_discount,
type_b_discount)"
funkcja PL/pgSQL company.new_product_and_producer(character varying,numeric,boolean,boolean,character varying,numeric,numeric), wiersz 10 w wyrażenie SQL
UWAGA: Got exception:
SQLSTATE: 23505
SQLERRM: podwójna wartość klucza narusza ograniczenie unikalności "unique_constraints"
UWAGA: podwójna wartość klucza narusza ograniczenie unikalności "unique_constraints"
CALL
company=>
```

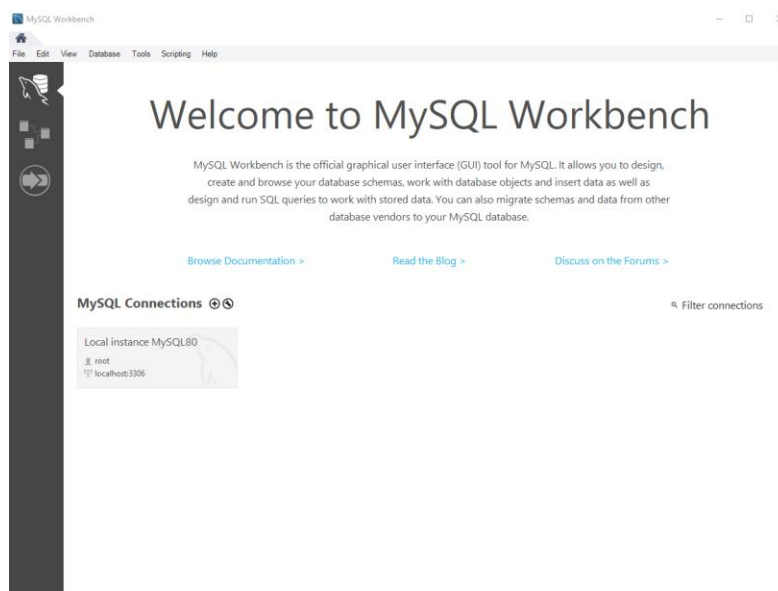
## Etap3

### 1. Instalacja środowiska MySQL

Po pierwszej części projektu przygotowanego w PostgreSQL zdecydowano się na migrację do MySQL.



### INSTALACJA MYSQL



### PANEL MYSQL WORKBENCH PO INSTALACJI.

## 2. Różnice pomiędzy PostgreSQL a MySQL

PostgreSQL i MySQL należą do baz danych o otwartym kodzie źródłowym i darmowej licencji, choć MySQL oferuje również wersję komercyjną.

PostgreSQL lepiej implementuje standard SQL, przez co jest postrzegany jako pewniejszy DBMS. Jest wykorzystywany w aplikacjach każdego rodzaju, od małych aplikacji internetowych po duże aplikacje wymagające analizy danych. Za wadę PostgreSQL względem MySQL można uznać wolniejszą operację odczytu danych. Za to posiada szereg zalet jak możliwość tworzenia własnych typów danych, szybsze operacje zapisu danych, zapewnia lepszą ochronę przy transakcjach oraz w spełnia zasady ACID.

MySQL jest szybką i lekką bazą danych, lecz poprzez mniejszą zgodność ze standardem SQL należy uważać przy większych zastosowaniach. Za to nadaje się idealnie do małych i średnich aplikacji internetowych, na których chcemy dokonywać podstawowych operacji na danych. MySQL zapewnia doskonałą szybkość odczytu danych.

Wybór pomiędzy tych dwóch DBMS zależy od potrzebnych zastosowań. Jeśli nie ma potrzeby korzystania z szerszych możliwości PostgreSQL oraz nie w bazie danych nie będą wykonywane skomplikowane operacje można skierować się w stronę MySQL. Lecz jeśli baza danych może przyjąć duże rozmiary oraz istnieje faktyczna potrzeba zaawansowanych operacji na danych lepiej postawić na PostgreSQL.

Istnieją różnice w dialekcie pomiędzy dwoma systemami baz danych. Problemy wydają się być większe przy przenoszeniu zapytań z MySQL po PostgreSQL niż w drugą stronę chociażby z racji na kwestie z apostrofami/cudzysłowami.

## 3. Migracja

Dostępnych jest kilka narzędzi pozwalających na migrację bazy danych z PostgreSQL do MySQL. W pierwszej kolejności wypróbowano MySQL Workbench, ponieważ jest to natywny client do DBMS MySQL. Wypróbowano również zewnętrzne narzędzie ESF Database Migration Toolkit.

### 3.1 MySQL Workbench Migration

MySQL Workbench udostępnia narzędzie Migration, które pozwala na przeprowadzenie migracji spośród kilku systemów DB.

Pierwszym krokiem jest wybranie źródła oraz docelowej bazy danych. Aby była możliwa migracja z PostgreSQL trzeba zainstalować odpowiedni Driver (ODBC).

The screenshot shows the 'Source Selection' dialog box in MySQL Workbench. The 'Database System' is set to 'PostgreSQL'. The 'Connection Method' is 'ODBC (manually entered parameters)'. The 'Advanced' tab is selected, showing fields for Driver, Hostname, Port, Username, Password, and Database. The 'Test Connection' button is highlighted.

Scripting Help

Source Selection

**Source RDBMS Connection Parameters**

Database System: PostgreSQL Select a RDBMS from the list of supported systems

Stored Connection: Select from saved connection settings

Connection Method: ODBC (manually entered parameters) Method to use to connect to the RDBMS

Parameters Advanced

Driver: PostgreSQL ODBC Driver (ANSI) The Driver name of your ODBC driver (E.g. psqodbc, PostgreSQL ANSI).

Hostname: 127.0.0.1 Port: 5432 Name or IP address of the server host. - TCP/IP port.

Username: management Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Database: company The database to connect to. Cannot be blank.

☐ Store connection for future usage as

Test Connection Open ODBC Administrator < Back Next > Cancel

KONFIGURACJA BAZY ŹRÓDŁOWEJ W WORKBENCH MIGRATION.

### Target RDBMS Connection Parameters

|                                           |                                                                                        |                                                                      |
|-------------------------------------------|----------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| Stored Connection:                        | <input type="text"/>                                                                   | Select from saved connection settings                                |
| Connection Method:                        | <input type="text" value="Standard (TCP/IP)"/>                                         | Method to use to connect to the RDBMS                                |
| <div>Parameters <b>SSL</b> Advanced</div> |                                                                                        |                                                                      |
| Hostname:                                 | <input type="text" value="127.0.0.1"/>                                                 | Port: <input type="text" value="3306"/>                              |
|                                           |                                                                                        | Name or IP address of the server host - and TCP/IP port.             |
| Username:                                 | <input type="text" value="root"/>                                                      |                                                                      |
|                                           |                                                                                        | Name of the user to connect with.                                    |
| Password:                                 | <input type="button" value="Store in Vault ..."/> <input type="button" value="Clear"/> | The user's password. Will be requested later if it's not set.        |
| Default Schema:                           | <input type="text"/>                                                                   |                                                                      |
|                                           |                                                                                        | The schema to use as default schema. Leave blank to select it later. |

### KONFIGURACJA BAZY DOCELOWEJ

Pierwszym napotkanym problemem był odpowiedni wybór Drivera. Okazało się, że pojawił się problem z kodowaniem znaków.

Reverse Engineer Source

Selected schema metadata will now be fetched from the source RDBMS and reverse engineered so that its structure can be determined.

☒ Connect to source DBMS  
☒ Reverse engineer selected schemas  
☐ Post-processing of reverse engineered schemas

Reverse engineering catalog information

A task has failed executing.

Message Log

```
- Reverse engineering catalog information
Traceback (most recent call last):
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\db_postgresql_re_grt.py", line 356, in reverseEngineer
 return PostgreSQLReverseEngineering.reverseEngineer(connection, catalog_name, schemata_list, context)
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\db_generic_re_grt.py", line 241, in reverseEngineer
 catalog = cls.reverseEngineerCatalog(connection, catalog_name)
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\db_generic_re_grt.py", line 401, in reverseEngineerCatalog
 ds.reverseEngineerSequences(connection, schema)
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\db_postgresql_re_grt.py", line 80, in reverseEngineerSequences
 min_value, max_value, start_value, increment_by, last_value, is_cycled, ncache = ds.execute_query(connection, seq_details_query %
(schema.name, seq_name)).fetchone()
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\db_generic_re_grt.py", line 80, in execute_query
 return ds.get_connection(connection_object).cursor().execute(query, *args, **kwargs)
pyodbc.DataError: ('22P05', '[22P05] ERROR: character with byte sequence 0xc5 0x81 in encoding "UTF8" has no equivalent in encoding
"WIN1252";\nError while executing the query (1) (SQLExecDirectW)')

Traceback (most recent call last):
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\workbench\wizard_progress_page_widget.py", line 197, in thread_work
 self.func()
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\migration_schema_selection.py", line 183, in task_reveng
 self.main.plan.migrationSource.reverseEngineer()
 File "C:\Program Files\MySQL\MySQL Workbench 8.0 CE\modules\migration.py", line 364, in reverseEngineer
 self.state.sourceCatalog = self._rev_eng_module.reverseEngineer(self.connection, self.selectedCatalogName, self.selectedSchemataNames,
self.state.applicationData)
SystemError: DataError("('22P05', '[22P05] ERROR: character with byte sequence 0xc5 0x81 in encoding "UTF8" has no equivalent in encoding
"WIN1252";\nError while executing the query (1) (SQLExecDirectW)'): error calling Python module function DbPostgresqlRE.reverseEngineer
ERROR: Reverse engineer selected schemas: DataError("('22P05', '[22P05] ERROR: character with byte sequence 0xc5 0x81 in encoding "UTF8"
has no equivalent in encoding "WIN1252";\nError while executing the query (1) (SQLExecDirectW)'): error calling Python module function
DbPostgresqlRE.reverseEngineer
Failed
```

### PROBLEM KODOWANIA PODCZAS MIGRACJI.

Rozwiązaniem było zmiana Drivera na typ UNICODE. Wpłynęło to na rodzaj kodowania znaków. Tym samym problem został rozwiązany.

**Source RDBMS Connection Parameters**

Database System: PostgreSQL ▼ Select a RDBMS from the list of supported systems

Stored Connection: ▼ Select from saved connection settings

Connection Method: ODBC (manually entered parameters) ▼ Method to use to connect to the RDBMS

Parameters **Advanced**

|           |                                                    |                                                                     |
|-----------|----------------------------------------------------|---------------------------------------------------------------------|
| Driver:   | PostgreSQL ODBC Driver(UNICODE)                    | The Driver name of your ODBC driver (E.g. psqldb, PostgreSQL ANSI). |
| Hostname: | 127.0.0.1                                          | Name or IP address of the server host. - TCP/IP port.               |
| Port:     | 5432                                               |                                                                     |
| Username: | management                                         | Name of the user to connect with.                                   |
| Password: | <span>Store in Vault ...</span> <span>Clear</span> | The user's password. Will be requested later if it's not set.       |
| Database: | company                                            | The database to connect to. Cannot be blank.                        |

ZMIANA DRIVERA NA UNICODE.

Następnym problemem był odczytanie „min\_value” z tabeli systemowych PostgreSQL.

Fix File "C:\Program Files\MySQL\MySQL Workbench 6.3 CE\modules\db\_postgresql\_re\_grt.py"  
lines ~70 change to

```
seq_details_query = """SELECT min_value, max_value, start_value,
increment_by, last_value, cycle as is_cycled, cache_size as cache_value
FROM pg_catalog.pg_sequences
WHERE schemaname = '%s' AND sequencename = '%s' """
```

EDYCJA PLIKÓW ŹRÓDŁOWYCH MYSQL DOSTOSOWUJĄC JE DO POSTGRESQL

Rozwiązaniem była edycja plików źródłowych MySQL i odpowiednie przypisanie nazw zmiennych. Różnice mogły pojawić się wraz z nowymi wersjami PostgreSQL.

Edycja pliku db\_postgresql\_re\_grt.py rozwiązała problem.

Następnie sprawdzana jest konfiguracja. Po pomyślnej konfiguracji należy wybrać schemat bazy danych, do migrowania.

**select the schemata you want to migrate:**

include

▼

☐
☐

Catalog/Schema

company

company

schemata selected Unselect All

**Schema Name Mapping Method**

Choose how the reverse engineered schemas and objects should be mapped.

☐ Keep schemas as they are: Catalog.Schema.Table -> Schema.Table  
☒ Only one schema: Catalog.Schema.Table -> Catalog.Table  
☐ Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema\_Table

## WYBRÓ SCHEMATU DO MIGRACJI.

Podczas próby odczytu schematu z PostgreSQL występował „crash” w MySQL Workbench i aplikacja zamykała się bez podania żadnego błędu.

Rozwiązaniem był downgrade do wersji 8.12.

Migration Task List

Manual Editing

OVERVIEW

Overview

SOURCE & TARGET

Source Selection

Target Selection

Fetch Schemas List

Schemas Selection

Reverse Engineer Source

OBJECT MIGRATION

Source Objects

Migration

Manual Editing

Target Creation Options

Create Schemas

Create Target Results

DATA MIGRATION

Data Transfer Setup

Bulk Data Transfer

REPORT

Migration Report

Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

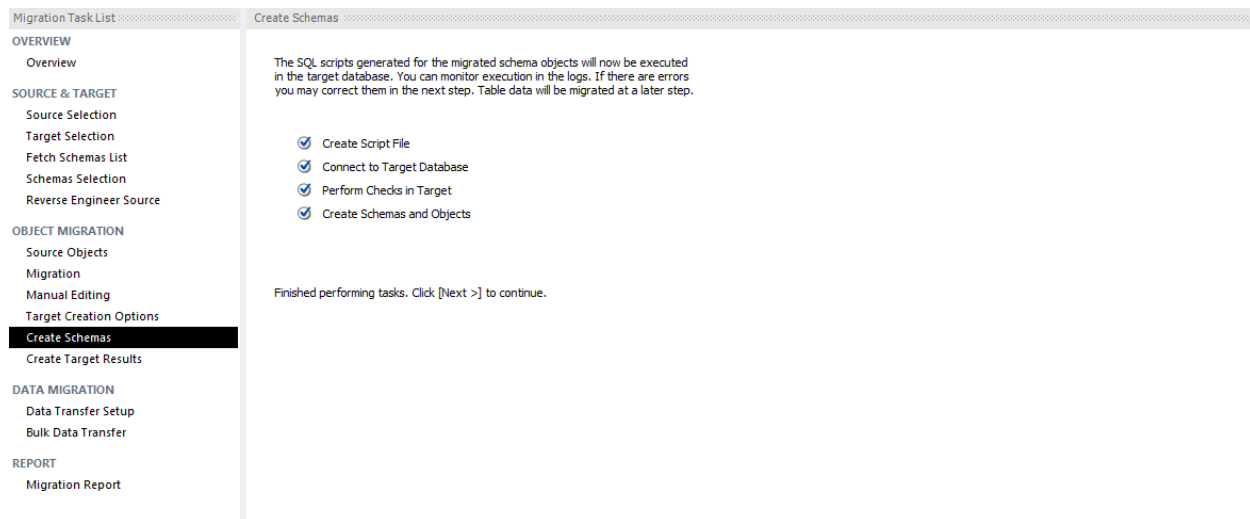
View: Migration Problem

No migration problems found. 14 warning(s).  
Use the View pulldown menu to review all objects.

## IFORMACJA O UKONCZENIU PROCESU MIGRACJI (POBRANIA ZAWARTOSCI Z POSTGRESQL).

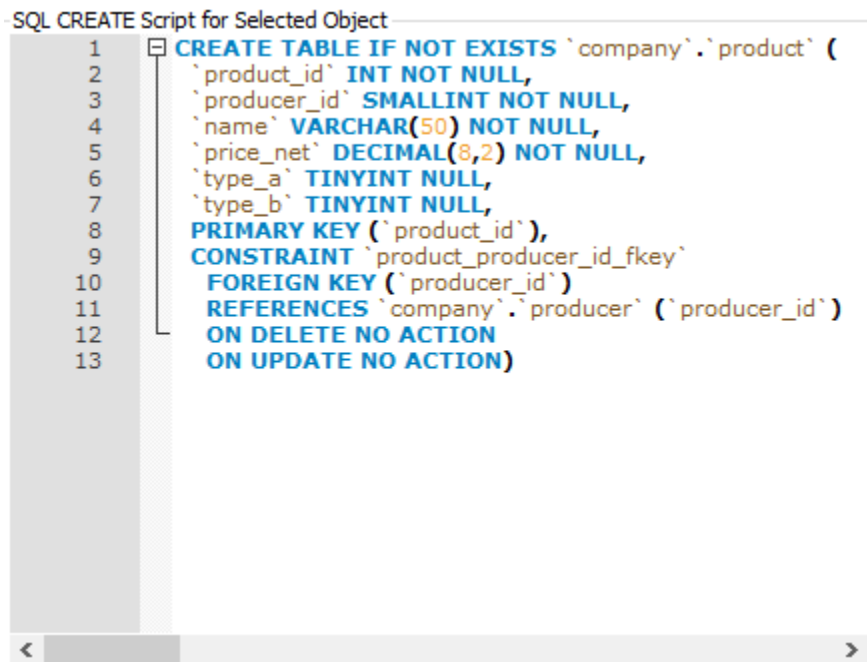


Po przeinstalowaniu wersji MySQL na starszą, udało się odczytać schemat z PostgreSQL.



### TWORZENIE ZMIGROWANEGO SCHEMATU W MYSQL.

Przed migracją jest możliwość podejrzenia zapytań SQL, które przygotowało narzędzie do migracji. Jeśli zauważymy błędy lub braki, na tym etapie można je uzupełnić.



SQL CREATE Script for Selected Object

```
1 CREATE TABLE IF NOT EXISTS `company`.`producer` (
2 `producer_id` SMALLINT NOT NULL,
3 `name` VARCHAR(50) NOT NULL,
4 `contact_person_id` SMALLINT NULL,
5 `type_a_discount` DECIMAL(3,2) NULL,
6 `type_b_discount` DECIMAL(3,2) NULL,
7 PRIMARY KEY (`producer_id`),
8 UNIQUE INDEX `unique_constraints` (`name` ASC) VISIBLE,
9 CONSTRAINT `producer_contact_person_id_fkey`
10 FOREIGN KEY (`contact_person_id`)
11 REFERENCES `company`.`contact_person` (`contact_person_id`)
12 ON DELETE NO ACTION
13 ON UPDATE NO ACTION)
```

SQL CREATE Script for Selected Object

```
1 CREATE TABLE IF NOT EXISTS `company`.`investment` (
2 `investment_id` INT NOT NULL,
3 `name` VARCHAR(50) NOT NULL,
4 `client_id` BIGINT NOT NULL,
5 `place_object_id` INT NULL,
6 `supervisor_id` INT NOT NULL,
7 `start_date` DATE NOT NULL,
8 `end_date` DATE NULL,
9 PRIMARY KEY (`investment_id`),
10 INDEX `invest_name_idx` (`name` ASC) VISIBLE,
11 CONSTRAINT `investment_client_id_fkey`
12 FOREIGN KEY (`client_id`)
13 REFERENCES `company`.`client` (`client_id`)
14 ON DELETE NO ACTION
15 ON UPDATE NO ACTION,
16 CONSTRAINT `investment_place_object_id_fkey`
17 FOREIGN KEY (`place_object_id`)
18 REFERENCES `company`.`place_object` (`place_object_id`)
19 ON DELETE NO ACTION
20 ON UPDATE NO ACTION,
21 CONSTRAINT `investment_supervisor_id_fkey`
22 FOREIGN KEY (`supervisor_id`)
23 REFERENCES `company`.`supervisor` (`supervisor_id`)
24 ON DELETE NO ACTION
25 ON UPDATE NO ACTION)
```

SQL CREATE Script for Selected Object

```
1 CREATE TABLE IF NOT EXISTS `company`.`sub_investment` (
2 `sub_investment_id` INT NOT NULL,
3 `name` VARCHAR(50) NOT NULL,
4 `investment_id` INT NOT NULL,
5 `start_date` DATE NOT NULL,
6 `end_date` DATE NULL,
7 PRIMARY KEY (`sub_investment_id`),
8 INDEX `sub_invest_name_idx` (`name` ASC) VISIBLE,
9 CONSTRAINT `sub_investment_investment_id_fkey`
10 FOREIGN KEY (`investment_id`)
11 REFERENCES `company`.`investment` (`investment_id`)
12 ON DELETE NO ACTION
13 ON UPDATE NO ACTION)
```

ZAPYTANIA TWORZACE TABELY W MYSQL. BRAK OGARNICZEŃ, KTÓRE BYŁY W POSTGRESQL.

W powyżej przedstawionych tabelach zabrakło ograniczeń, które były zdefiniowane w PostgreSQL. Poniżej przedstawiono jedno z ograniczeń, którego zabrakło.

```
CREATE TABLE company.sub_investment (
 sub_investment_id SERIAL NOT NULL PRIMARY KEY,
 name VARCHAR(50) NOT NULL,
 investment_id INTEGER NOT NULL,
 start_date DATE NOT NULL,
 end_date DATE,
 CONSTRAINT check_dates CHECK(end_date > start_date),
 FOREIGN KEY(investment_id) REFERENCES company.investment(investment_id)
);
```

OGRANICZENIE W TABELU SUB\_INVESTMENT SPRAWDZAŁO CZY DATA KONCOWA NIE JEST WCZEŚNIEJSZA NIŻ POCZĄTKOWA. JEDNO Z UTRACONYCH OGRANICZEŃ W CZASIE MIGRACJI.

Po zaakceptowaniu schematu, tabele zostały utworzone w bazie MySQL.

Następnym krokiem było przeniesienie zawartości tabel.

Migration Task List

Overview

SOURCE & TARGET

Source Selection

Target Selection

Fetch Schemas List

Schemas Selection

Reverse Engineer Source

OBJECT MIGRATION

Source Objects

Migration

Manual Editing

Target Creation Options

Create Schemas

Create Target Results

DATA MIGRATION

Data Transfer Setup

Bulk Data Transfer

REPORT

Migration Report

Data Transfer Setup

Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

☒ Online copy of table data to target RDBMS

☐ Create a batch file to copy the data at another time

Batch File:  Browse...

You should edit this file to add the source and target server passwords before running it.

☐ Create a shell script to use native server dump and load abilities for fast migration

Bulk Data Copy Script:  Browse...

Edit the generated file and change passwords at the top of the generated script.  
Run it on the source server to create a zip package containing a data dump as well as a load script.  
Copy this to the target server, extract it, and run the import script. See the script output for further details.

Options

☒ Truncate target tables (i.e. delete contents) before copying data

Worker tasks  ⓘ

☒ Enable debug output for table copy

☒ Driver sends data already encoded as UTF-8.

KONFIGURACJA PRZENIESIEMIA ZAWARTOŚCI TABEL DO MYSQL.

# Message Log

```

00:41:15 [DB2][copytable]: 1 - client_id: MYSQL_TYPE_LONGLONG
00:41:15 [DB2][copytable]: 2 - name: MYSQL_TYPE_STRING
00:41:15 [DB2][copytable]: 3 - surname: MYSQL_TYPE_STRING
00:41:15 [DB2][copytable]: 4 - company_name: MYSQL_TYPE_STRING
00:41:15 [DB2][copytable]: 5 - phone: MYSQL_TYPE_STRING
00:41:15 [DB2][copytable]: 6 - email: MYSQL_TYPE_STRING
00:41:15 [INF][copytable]: Truncating table `company`.`client`

`company`.`client`:Finished copying 10 rows in 0m00s
41:15 [INF][copytable]: Statement execution failed: Incorrect decimal value: '0.4' for column 'type_a_discount' at row 1:

ERROR: `company`.`producer`:Inserting Data: Incorrect decimal value: '0.4' for column 'type_a_discount' at row 1
ERROR: `company`.`producer`:Failed copying 11 rows
41:15 [INF][copytable]: Re-enabling triggers for schema 'company'
41:15 [DB1][copytable]: Retrieving trigger definitions
41:15 [INF][copytable]: No triggers found for 'company'

Copy helper has finished

Data copy results:
- `company`.`contact_person` has succeeded (10 of 10 rows copied)
- `company`.`investment` has succeeded (9 of 9 rows copied)
- `company`.`product` has succeeded (11 of 11 rows copied)
- `company`.`log_price_update` has succeeded (1 of 1 rows copied)
- `company`.`place_object` has FAILED (0 of 10 rows copied)
- `company`.`supervisor` has succeeded (12 of 12 rows copied)
- `company`.`sub_investment` has succeeded (9 of 9 rows copied)
- `company`.`product_order` has succeeded (9 of 9 rows copied)
- `company`.`producer` has FAILED (0 of 11 rows copied)
- `company`.`client` has succeeded (10 of 10 rows copied)
8 tables of 10 were fully copied
Click [Retry] to retry copying remaining data from tables
Copy data to target RDBMS finished
Tasks finished with warnings and/or errors; view the logs for details
Finished performing tasks.

```

## REZULAT PRZENOSZENIA DANYCH.

Poniższe zdjęcie przedstawia rezultat przenoszenia danych. Dane z 8 z 10 tabel zostały przeniesione bez problemu. Problem pojawił się przy tabeli producer oraz place\_object. Niestety nie udało się rozwiązać tego problemu.

The following tasks will now be performed. Please monitor the execution.

- ☒ Prepare information for data copy
- ☒ Determine number of rows to copy
- ☐ Copy data to target RDBMS

Click [Next >] to execute.

A task has failed executing.

```
Message Log
00:42:02 [INF][copytable]: --table "company"."company"."log_price_update" "company" "log_price_update" -
"log_id", "log_date", "product_id", "product_name", "old_price", "new_price", "product_producer"
00:42:02 [INF][copytable]: --table "company"."company"."place_object" "company" "place_object" "place_object_id"
"place_object_id", "street", "number", "local", "post_code", "city", "country"
00:42:02 [INF][copytable]: --table "company"."company"."supervisor" "company" "supervisor" "supervisor_id"
"supervisor_id", "name", "surname", "position", "phone", "email"
00:42:02 [INF][copytable]: --table "company"."company"."sub_investment" "company" "sub_investment" "sub_investment_id"
"sub_investment_id", "name", "investment_id", "start_date", "end_date"
00:42:02 [INF][copytable]: --table "company"."company"."product_order" "company" "product_order"
"product_id", "sub_investment_id", "product_id", "sub_investment_id" "product_id::INT as 'product_id', 'sub_investment_id', 'quantity'
00:42:02 [INF][copytable]: --table "company"."company"."producer" "company" "producer" "producer_id"
"producer_id", "name", "contact_person_id::SMALLINT as 'contact_person_id', 'type_a_discount', 'type_b_discount'
00:42:02 [INF][copytable]: --table "company"."company"."client" "company" "client" "client_id" "client_id"
"name", "surname", "company_name", "phone", "email"
00:42:02 [INF][copytable]: Opening ODBC connection to [Postgresql] 'DRIVER=PostgreSQL ODBC Driver
(UNICODE);SERVER=127.0.0.1;PORT=5432;DATABASE=company;UID=management;UseDeclareFetch=1;PWD=XXX'
00:42:03 [INF][copytable]: ODBC connection to 'DRIVER=PostgreSQL ODBC Driver
(UNICODE);SERVER=127.0.0.1;PORT=5432;DATABASE=company;UID=management;UseDeclareFetch=1;PWD=' opened
00:42:03 [INF][copytable]: Connecting to MySQL server at 127.0.0.1:3306 with user root
00:42:03 [INF][copytable]: Connection to MySQL opened
00:42:03 [INF][copytable]: Resuming copy of table 'company'. 'contact_person'. Starting on record with keys: 'contact_person_id': 10
00:42:03 [INF][copytable]: Resuming copy of table 'company'. 'investment'. Starting on record with keys: 'investment_id': 10
00:42:03 [INF][copytable]: Resuming copy of table 'company'. 'product'. Starting on record with keys: 'product_id': 13
00:42:03 [ERR][copytable]: Exception: Get last copied row: Cannot get last copied record from table with no PK.

Loading table information from file c:\users\yoszim~1\appdata\local\temp\tmpzhgbig
ROW_COUNT:"company"."company"."contact_person": 0
ROW_COUNT:"company"."company"."investment": 0
ROW_COUNT:"company"."company"."product": 0
ERROR: Determine number of rows to copy: Error getting row count from source tables, wbcopytables exited with code 1
Failed

Resuming...
Failed

Resuming...
Failed

Resuming...
Failed
```

BŁĄD PRZY ZAKOŃCZENIU MIGRACJI DANYCH.

Procesu migracji (formalnie) również nie został zakończony. Przy próbie przejścia do końcowego etapu migracji jakim jest generowanie raportu pojawiał się komunikat „Resumin... Failed”. Być może migracja danych musi się odbyć bez problemu by przejść do ostatniego etapu.

### 3.2 ESF Database Migration Toolkit

Spróbowano również migracji za pomocą ESF Database Migration Toolkit. Pierwszym krokiem było wybór źródła oraz docelowej bazy danych.

ESF Database Migration Toolkit - Pro - TRIAL

**Choose a Data Source**

From where do you want to copy data? You can copy data from one of the following sources.

Source: PostgreSQL

Server: localhost Port: 5432

Username: management

Password: ●●●●●●●●

Database: company

Schema: company

KONFIGURACJA ŹRÓDŁOWEJ BAZY DANYCH.

ESF Database Migration Toolkit - Pro - TRIAL

**Choose a Destination**

To where do you want to copy data? You can copy data to one of the following destinations.

Destination: MySQL

Server: localhost Port: 3306

Username: root Engine:

Password: ●●●● CharSet:

Database: company\_ESF\_migration

KONFIGURACJA DOCELOWEJ BAZY DANYCH.

Konfiguracja była o wiele szybsza i prostsza.

**Execution**

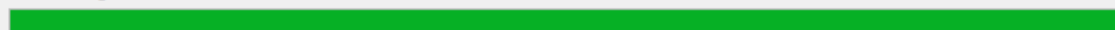
Had specified necessary information. Click Submit to execute.



Table Progress:



Record Progress:



Log:

```
=====
Table 'product' exists. Skip it ...
=====
Table 'product_order' exists. Skip it ...
=====
Table 'sub_investment' exists. Skip it ...
=====
Table 'supervisor' exists. Skip it ...
=====
Table 'investment_info' exists. Skip it ...
=====
Getting 'investment_info1' table structure ...
Creating table 'investment_info1' ...
Open SOURCE('investment_info1') recordset failed!
Błąd: permission denied for view investment_info1

0 records inserted.
Time Spent: 00:00:00.187
=====
Getting 'product_orders_value' table structure ...
Get 'product_orders_value' structure failed!
Błąd: bieżąca transakcja została przerwana, polecenia ignorowane do końca bloku transakcji

=====
Total error(s) count: (2).
Total Time: 00:00:00.406, migration completed!
```

Browse Log

Clear

Save as job ...

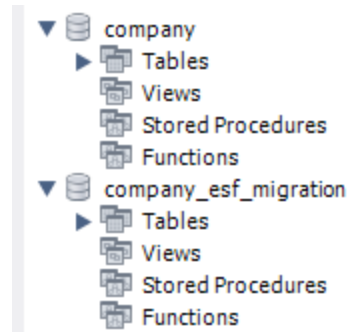
**REZULTAT MIGRACJI.**

Jedyny błędy jakie wystąpiły były związane z perspektywą product\_orders\_value oraz investment\_info (problem prawdopodobnie wynikał z niewystarczających uprawnień użytkownika „management”).

Poza tym wszystkie dane zostały zmigrowane do MySQL.

### 3.3 Podsumowanie migracji

Po weryfikacji schematów w MySQL okazało się, że, zmigrowane zostały tylko tabele oraz dane (pomijając place\_object oraz producent przy migracji przez Workbench). Perspektywy zostały przeniesione tylko przez narzędzie ESF i zapisano jako tabele. Procedury oraz funkcje nie zostały przeniesione.



ZMIGROWANE SCHEMATY W ŚRODOWISKU MYSQL WORKBENCH.

| supervisor_id | name     | surname    | position | phone        | email                       |
|---------------|----------|------------|----------|--------------|-----------------------------|
| 1             | Ogdon    | Kauschke   | engineer | 228 911 2936 | okauschke0@wired.com        |
| 2             | Chauncey | Drewe      | engineer | 474 860 1874 | cdrewe1@independent.co.uk   |
| 3             | Jewel    | Dik        | engineer | 974 941 9143 | jdik2@google.cn             |
| 4             | Cori     | Follacaro  | engineer | 752 997 3851 | cfollacaro3@a8.net          |
| 5             | Walliw   | Regler     | engineer | 789 582 4255 | wregler4@nydailynews.com    |
| 6             | Stirling | Williams   | engineer | 925 222 8698 | swilliams5@mashable.com     |
| 7             | Danella  | Valti      | engineer | 317 883 0721 | dvalti6@ucsd.edu            |
| 8             | Katina   | Lubbock    | engineer | 857 635 9630 | klubbock7@lulu.com          |
| 9             | Ladonna  | Matyasik   | engineer | 146 450 5182 | lmatyasik8@chronoengine.com |
| 10            | Iormina  | Lodeke     | engineer | 365 179 3664 | ilodeke9@google.co.jp       |
| 12            | daniel2  | michrow... | test2    | test2        | test2@test.pl               |
| 13            | daniel   | michrowski | test     | test         | test@test.pl                |
| NULL          | NULL     | NULL       | NULL     | NULL         | NULL                        |

ZAWARTOŚĆ TABELI SUPERVISOR ZE SCHEMATU COMPANY\_ESF\_MIGRATION.

```
company=# Select * from company.supervisor;
 supervisor_id | name | surname | position | phone | email
-----+-----+-----+-----+-----+-----
 1 | Ogdon | Kauschke | engineer | 228 911 2936 | okauschke0@wired.com
 2 | Chauncey | Drewe | engineer | 474 860 1874 | cdrewe1@independent.co.uk
 3 | Jewel | Dik | engineer | 974 941 9143 | jdik2@google.cn
 4 | Cori | Follacaro | engineer | 752 997 3851 | cfollacaro3@a8.net
 5 | Walliw | Regler | engineer | 789 582 4255 | wregler4@nydailynews.com
 6 | Stirling | Williams | engineer | 925 222 8698 | swilliams5@mashable.com
 7 | Danella | Valti | engineer | 317 883 0721 | dvalti6@ucsd.edu
 8 | Katina | Lubbock | engineer | 857 635 9630 | klubbock7@lulu.com
 9 | Ladonna | Matyasik | engineer | 146 450 5182 | lmatyasik8@chronoengine.com
 10 | Iormina | Lodeke | engineer | 365 179 3664 | ilodeke9@google.co.jp
 12 | daniel2 | michrowski2 | test2 | test2 | test2@test.pl
 13 | daniel | michrowski | test | test | test@test.pl
(12 rows)
```

ZAWARTOŚĆ TABLI SUPERVISOR W ŚRODOWISKU POSTGRESQL.

Należałoby dostosować wszystkie procedury oraz funkcje do składni MySQL.



Podjęto próbę przeniesienia funkcji. Niestety natrafiłem na problem ze składnią, których nie udało mi się poprawić.

```
DELIMITER $$

CREATE FUNCTION company_esf_migration.address() RETURNS VARCHAR
BEGIN
 DECLARE address VARCHAR;

 SELECT CONCAT("st. ", street, " ", number, " ", post_code, " ", city, " ", country) INTO address
 FROM company_esf_migration.place_object;

 RETURN address;number
END$$

DELIMITER ;
```

#### PRÓBA PRZENIESIENIA FUNKCJI ADDRESS() DO MYSQL.

```
12:41:50 CREATE FUNCTION company_esf_migration.address() RETURNS VARCHAR BEGIN
DECLARE address VARCHAR; SELECT CONCAT("st. ", street, " ", number, " ", post_code, " ", city, "
,", country) INTO address FROM company_esf_migration.place_object; RETURN address;number
END Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near 'BEGIN DECLARE address VARCHAR;
SELECT CONCAT("st. ", street, " ", numb' at line 2 0.000 sec
```

#### BŁĄD ZWRACANY PRZEZ MYSQL.

Funkcja z obsługą błędów należałoby napisać zupełnie od nowa, ponieważ wykorzystywała narzędzie wbudowane w PostgreSQL.

### 3.4 Wnioski

Migracja zakończyła się częściowym powodzeniem. Udało się przenieść schemat bazy danych, wszystkie tabel z odpowiednimi typami oraz kluczami. Dane zawarte w tabelach również zostały poprawnie przeniesione.

Perspektywy zostały przeniesione jako tabele do bazy MySQL. Funkcje, procedury i wyzwalacze nie zostały przeniesione. Należałoby zaimplementować je zgodnie ze składnią MySQL.

Narzędzie MySQL Workbench Migration przysporzyło sporo problemów. Dane z 2 tabel nie zostały przeniesione, wskazując niezrozumiałe błędy. Problemy wynikały również z działania samego narzędzia pokazując, że nie jest to najlepsze narzędzie do migracji.

O wiele sprawniej przebiegła migracja używając ESF Database Migration Toolkit. Wszystkie dane zostały przeniesione poprawnie, bez żadnych problemów konfiguracyjnych.

## 4. Połączenie bazodanowe

Do połączenia bazodanowego pomiędzy PostgreSQL i MySQL można użyć narzędzie Foreign Data Wrapper.

[https://github.com/EnterpriseDB/mysql\\_fdw/blob/master/README.md](https://github.com/EnterpriseDB/mysql_fdw/blob/master/README.md)

Niestety przy próbie konfiguracji narzędzia wystąpił błąd.

```
PS C:\Users\yosizimitsu\Desktop\mysql_fdw-master> make USE_PGXS=1
process_begin: CreateProcess(NULL, mysql_config --include, ...) failed.
Makefile:16: pipe: No error
process_begin: CreateProcess(NULL, mysql_config --libs, ...) failed.
Makefile:17: pipe: No error
process_begin: CreateProcess(NULL, uname, ...) failed.
Makefile:27: pipe: No error
Makefile:44: *** PostgreSQL 9.5, 9.6, 10, 11, 12, or 13 is required to compile this extension. Stop.
PS C:\Users\yosizimitsu\Desktop\mysql_fdw-master> postgres --version
postgres (PostgreSQL) 12.4
PS C:\Users\yosizimitsu\Desktop\mysql_fdw-master>
```

BŁĄD PRZY PRÓBIE KONFIGURACJI NARZĘDZIA FOREIGN DATA WRAPPER W ŚRODOWISKU WINDOWS.

Prawdopodobnie narzędzie jest kompatybilne z systemami Linux. W systemie Windows nie udało mi się rozwiązać problemu.

Istnieją narzędzie 'dblink' w PostgreSQL, które pozwalają na połączenie między bazami danych w systemie PostgreSQL.

```
company=# CREATE EXTENSION dblink WITH SCHEMA company;
CREATE EXTENSION
company=#
```

```
company=# CREATE SERVER server_company_remote FOREIGN DATA WRAPPER dblink_fdw OPTIONS (host 'remote', dbname 'remote', port '5432');
CREATE SERVER
company=#
```

```
company=# CREATE USER MAPPING FOR postgres SERVER server_company_remote OPTIONS (user 'root',password 'root');
CREATE USER MAPPING
company=#
```

```
company=# GRANT USAGE ON FOREIGN SERVER server_company_remote TO postgres;
GRANT
company=#
```

```
company=# SELECT * from company.dblink('host=localhost user=postgres password=123 dbname=company','select * from company.client') AS x(client_id integer,name VARCHAR(50), surname VARCHAR(50), company_name VARCHAR(50), phone VARCHAR(15), email VARCHAR(70));
```

| client_id | name       | surname    | company_name                  | phone        | email                    |
|-----------|------------|------------|-------------------------------|--------------|--------------------------|
| 1         | Beverie    | Nevet      | Graham, Hudson and Hettinger  | 123 557 7970 | bnevet0@github.io        |
| 2         | Kelvin     | Dolder     | Hahn Group                    | 129 842 9118 | kdolder1@oracle.com      |
| 3         | Kaylil     | Lotte      | Lueilwitz Inc                 | 819 330 4006 | klotte2@mapquest.com     |
| 4         | Reeta      | Astman     | Casper-Harber                 | 322 893 4668 | rastman3@pen.io          |
| 5         | Noby       | Theze      | Johnston-Spencer              | 976 549 1871 | ntheze4@ning.com         |
| 6         | Maximilian | Herculeson | Turcotte-Daniel               | 411 442 5859 | mherculeson5@1und1.de    |
| 7         | Lilly      | Billiard   | Goldner, Parker and Dibbert   | 991 112 7980 | lbilliard6@earthlink.net |
| 8         | Olly       | Bilovsky   | Hartmann LLC                  | 266 957 4203 | obilovsky7@homestead.com |
| 9         | Nicolea    | Luker      | Stark-Brown                   | 595 190 5940 | nluker8@goo.ne.jp        |
| 10        | Grier      | Dory       | Altenwerth, Kozey and Johnson | 948 913 1268 | gdory9@ovh.net           |
| 11        | test       | test       | test                          | 123          | test@test.pl             |

(11 rows)

POLACZENIE BAZ DANYCH W SRODOWISKU POSTGRESQL Z UZYCIEM DBLINK.

