

Sprawozdanie – Etap II

Relacyjne systemy zarządzania bazami danych

Funkcja użytkownika (UDF)

Stworzono funkcję użytkownika (UDF) `calc_pet_age` która pobiera datę urodzin i zwraca wiek zwierzęcia.

```
1  -- UDF
2  CREATE OR REPLACE FUNCTION calc_pet_age(date_of_birth DATE)
3  RETURNS INTEGER AS $$
4  DECLARE
5      pet_age INTEGER;
6  BEGIN
7      IF date_of_birth IS NULL THEN
8          RETURN NULL;
9      END IF;
10
11     SELECT DATE_PART('year', AGE(date_of_birth)) INTO pet_age;
12     RETURN pet_age;
13 END;
14 $$ LANGUAGE plpgsql;
15
16 SELECT pet_name, calc_pet_age(date_of_birth) AS age
17 FROM pet;
```

Data Output Messages Notifications

	pet_name character varying (100)	age integer
1	Napoleon	6
2	Kłatka	5
3	Laweta	7
4	Lea	8
5	Dakota	5
6	Lili	6
7	Boniek	7

Procedury do dodawania nowych elementów

Przygotowano procedury do dodawania nowych elementów do bazy danych.

Procedura dla weterynarzy do dodawania pomiarów zwierzęcia:

```

1  set role amazrczak;
2  call insert_pet_measurement_vet(
3    '2023-01-09',
4    33.50,
5    75.20,
6    5
7  );
8
9  select * from vet_amazurczak_measurements where pet_name = 'Dakota';
10

```

Data Output Messages Notifications

	measurement_id integer	pet_name character varying (100)	measurement_date date	weight numeric (5,2)	height numeric (5,2)
1	5	Dakota	2021-07-26	32.30	75.20
2	7	Dakota	2023-01-09	33.50	75.20

Procedura dla weterynarzy do dodawania testów laboratoryjnych:

```

1  set role amazrczak;
2  call insert_lab_test_vet(
3    '2020-12-12 17:00:00',
4    'Erytrocyty',
5    '56.2',
6    'Lekka utrata krwi',
7    5
8  );
9  select * from vet_amazurczak_lab_tests;
10

```

Data Output Messages Notifications

	lab_test_id integer	pet_name character varying (100)	test_date timestamp without time zone	test_type character varying (300)	test_results text	notes text
1	1	Laweta	2019-11-24 00:00:00	FT4	4.5 jednostek	Wyniki sugerują niedoczynność tarczycy
2	2	Dakota	2020-12-12 00:00:00	Leukocyty	13.0	Podejrzane lekkie zakażenie.
3	12	Dakota	2020-12-12 17:00:00	Erytrocyty	56.2	Lekka utrata krwi

Procedura dla weterynarzy do dodawania zabiegów medycznych:

```

1  set role amazrczak;
2  call insert_treatment_vet(
3    'Szczepienie',
4    'Podano szczepionkę na wściekliznę.',
5    'Przy okazji wizyty z powodu igieł w pysku',
6    5
7  );
8  select * from vet_amazurczak_treatments;
9

```

Data Output Messages Notifications

	treatment_id integer	visit_date timestamp without time zone	pet_name character varying (100)	treatment_type character varying (300)	procedure text	notes text
1	4	2020-12-12 17:00:00	Dakota	Zabieg	Usunięto igły jeża z pysku psa, podano antybiotyki.	[null]
2	6	2018-12-01 00:00:00	Napoleon	Szczepienie	Podano szczepionkę.	[null]
3	8	2019-11-11 00:00:00	Boniek	Szczepienie	Podano szczepionkę na wściekliznę.	[null]
4	13	2020-12-12 17:00:00	Dakota	Szczepienie	Podano szczepionkę na wściekliznę.	Przy okazji wizyty z powodu igieł w pysku

Procedura dla weterynarzy do dodawania diagnoz:

```
1 set role amazurczak;
2 call insert_diagnosis_vet(
3     'Opuchlizna urazowa',
4     'Opuchlizna wywołana urażeniami od igły',
5     5
6 );
7 select * from vet_amazurczak_diagnosis where pet_name='Dakota';
8
```

Data Output

Messages








































































































































Notifications

SQL

	<div>diagnosis_id</div> <div>integer</div>	<div>visit_date</div> <div>timestamp without time zone</div>	<div>pet_name</div> <div>character varying (100)</div>	<div>diagnosis_type</div> <div>text</div>	<div>notes</div> <div>text</div>
1	4	2020-12-12 17:00:00	Dakota	Zakażenie bakteryjne	Zakażenie bakteryjne spowodowane obrażeniami od igieł je...
2	11	2020-12-12 17:00:00	Dakota	Opuchlizna urazowa	Opuchlizna wywołana urażeniami od igły

Procedura dla weterynarzy do dodawania planu lekowego:

```
1 set role amazurczak;
2 call insert_medicine_plan_vet(
3     'Hydrokortyzol',
4     '4',
5     'ml',
6     '2 razy dziennie',
7     '1 tyg',
8     11
9 );
10 select * from vet_amazurczak_medication_plans where pet_name='Dakota';
11
```

Data Output		Messages	Notifications
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			
			

Procedura dla weterynarzy do dodawania recept:

```
1 set role amazurczak;
2 call insert_prescription_vet(
3     '2020-12-12',
4     'Hydrokortyzol',
5     1,
6     5
7 );
8 select * from vet_amazurczak_prescriptions where pet_name='Dakota';
```

Data Output						Messages	Notifications
	prescription_id integer	pet_name character varying (100)	prescription_date date	medicine_name character varying (100)	medicine_amount integer		
1	7	Dakota	2020-12-12	Amoksyicillin	1		
2	13	Dakota	2020-12-12	Hydrokortyzol	1		

Procedury dla recepcjonistów do dodawania zwierząt i ich właścicieli:

```
1 set role jopolska;
2 call insert_pet_owner('Anita','Wielkopolska','+48 444-444-444','anita.wielk.mail@gmail.com');
3
4 call insert_pet('Śnieżka',null,'F',2,36);
5 select * from recept_pet_owner where owner_id=36;
```

Data Output Messages Notifications

	owner_id integer	name text	phone_number text	email_address character varying (100)	pet_name character varying (100)
1	36	Anita Wielkopolska	+48 444-444-444	anita.wielk.mail@gmail.com	Śnieżka

Procedura dla recepcjonistów do dodawania wizyt:

```
1 set role jopolska;
2
3 call insert_visit_recept('2024-04-20 16:00:00','Problemy z stawami',6,38);
4 select * from recept_visits where pet_name='Śnieżka';
5
```

Data Output Messages Notifications

	visit_id integer	visit_date timestamp without time zone	reason text	pet_name character varying (100)	pet_owner text
1	31	2024-04-20 16:00:00	Problemy z stawami	Śnieżka	Anita Wielkopolska

Procedury dla recepcjonistów do dodawania pracowników i ich stanowisk:

```
1 set role jopolska;
2
3 call insert_staff_recept('Agata','Nowak-Kowalska','444-343-454',null,3,'2024-01-02');
4 select * from recept_staff where phone_number = '444-343-454';
5
```

Data Output Messages Notifications

	staff_id integer	name text	phone_number text	position text	specialization text
1	16	Agata Nowak-Kowalska	444-343-454	Sprzątac	[null]

```
1 set role jopolska;
2
3 call insert_staff_position('2023-07-02','2023-09-02',16,3);
4 select * from recept_staff_positions where position='Sprzątac';
5
```

Data Output Messages Notifications

	staff_position_id integer	staff_name text	position text	specialization text	from_date date	to_date date
1	2	Sebastian Danielczyk	Sprzątac	[null]	2019-02-01	[null]
2	12	Agata Nowak-Kowal...	Sprzątac	[null]	2024-01-02	[null]
3	15	Agata Nowak-Kowal...	Sprzątac	[null]	2023-07-02	2023-09-02

Nie wszyscy użytkownicy mogą dodawać dane do bazy danych. Weterynarze mogą dodawać pomiary zwierząt-pacjentów, wykonane testy, procedury/zabiegi medyczne, diagnozy, plany lekowe i recepty. Są oni ograniczeni do dodawania danych tylko dla ‘swoich’

pacjentów, tzn. dla pacjentów z którymi choć raz mieli wizytę. Recepcjoniści mogą dodawać wizyty, pracowników i ich stanowiska, zwierzęta oraz właścicieli zwierząt. Pozostali użytkownicy nie mogą dodawać danych do bazy danych.

Procedury przeznaczone dla weterynarzy weryfikują, czy dany weterynarz może wprowadzać dane dla danego zwierzęcia (czy zwierzę jest jego pacjentem) oraz walidują dane wejściowe. Sprawdzają, między innymi, czy dane nie są null'em, czy mają odpowiedni zakres i typ wartości, czy nie są wartościami pustymi. Procedury dla recepcjonistów również walidują dane w podobny sposób przed wstawieniem ich do tabeli.

Kod procedur oraz funkcji w nich wykorzystanych (do weryfikacji czy np. dany weterynarz może wprowadzić dane dla danego zwierzęcia) znajduje się w insert procedures.sql oraz util.sql, załączone wraz z sprawozdaniem.

Złożona procedura

Stworzono procedurę `add_staff_with_position`, która przyjmuje dane na temat pracownika i jego pozycji pracy. Procedura waliduje dane wejściowe i sprawdza, czy pozycja o danej nazwie (np. „Ochroniarz”) już istnieje. Jeśli nie, to ją tworzy. Następnie, dodaje ona dane do tabel `Staff` i `Staff_position`.

```
8  call add_staff_with_position('Adam', 'Witowski', '444-777-666', null,
9    'Ochroniarz', '2024-04-02', null);|
10 select * from recept_staff_positions where position='Ochroniarz';
11
```

Data Output Messages Notifications

	staff_position_id integer	staff_name text	position text	specialization text	from_date date	to_date date
1	16	Adam Witowski	Ochroniarz	[null]	2024-04-02	[null]

```

1 CREATE OR REPLACE PROCEDURE add_staff_with_position(
2     first_name VARCHAR(100),
3     last_name VARCHAR(100),
4     phone_number TEXT,
5     specialization TEXT,
6     nposition_name TEXT,
7     from_date DATE,
8     to_date DATE
9 )
10 LANGUAGE plpgsql
11 AS $$
12 DECLARE
13     nstaff_id INTEGER;
14     nposition_id INTEGER;
15 BEGIN
16     BEGIN
17         IF nposition_name IS NULL OR nposition_name = '' THEN
18             RAISE EXCEPTION 'Position name cannot be null or empty.';
19         END IF;
20
21         SELECT pos.position_id INTO nposition_id
22             FROM position pos
23             WHERE pos.position_name = nposition_name;
24
25         IF nposition_id IS NULL THEN
26             INSERT INTO position (position_name)
27                 VALUES (nposition_name)
28                 RETURNING position_id INTO nposition_id;
29         END IF;
30
31         IF first_name IS NULL OR first_name = '' OR last_name IS NULL OR last_name = '' THEN
32             RAISE EXCEPTION 'Staff name cannot be null or empty.';
33         END IF;
34
35         IF phone_number IS NULL OR phone_number = '' THEN
36             RAISE EXCEPTION 'Phone number cannot be null or empty.';
37         END IF;
38
39         INSERT INTO staff (first_name, last_name, phone_number, specialization)
40             VALUES (first_name, last_name, phone_number, specialization)
41             RETURNING staff_id INTO nstaff_id;
42
43         IF from_date IS NULL OR from_date = '' THEN
44             RAISE EXCEPTION 'Employment date cannot be null or empty.';
45         END IF;
46
47         INSERT INTO staff_position (from_date, to_date, position_id, staff_id)
48             VALUES (from_date, to_date, nposition_id, nstaff_id);
49
50     EXCEPTION
51     WHEN OTHERS THEN
52         RAISE EXCEPTION 'ERROR: %', SQLERRM;
53     END;
54 $$;

```

Wyzwalacze (Triggery)

Stworzono trzy wyzwalacze:

trigger_correct_pet_gender_case

Jeśli płeć zwierzęcia jest wprowadzana małą literą ('f', 'm'), wyzwalacz ten zmienia ją na dużą literę ('F', 'M'). Wykonuje on się przed operacją **INSERT** na tabeli **pet**.

```
-- Trigger - płeć z dużej litery

CREATE OR REPLACE FUNCTION correct_pet_gender_case()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.gender = 'f' THEN
        NEW.gender := 'F';
    ELSIF NEW.gender = 'm' THEN
        NEW.gender := 'M';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_correct_pet_gender_case
BEFORE INSERT ON pet
FOR EACH ROW
EXECUTE FUNCTION correct_pet_gender_case();

1 INSERT INTO pet
2 (pet_name, date_of_birth, gender, species_id, owner_id)
3 VALUES
4 ('Czarek', '2020-11-05', 'm', 1, 4);
5 SELECT * FROM pet WHERE pet_name='Czarek';
```

Data Output Messages Notifications

	pet_id [PK] integer	pet_name character varying (100)	date_of_birth date	gender "char"	species_id integer	owner_id integer
1	39	Czarek	2020-11-05	M	1	4

trigger_prevent_duplicate_position_names

Wyzwalacz ten nie pozwala na wstawienie duplikatów nazw stanowisk pracy. Uruchamiany jest przed wstawieniem do tabeli `position`.

```
20 -- Trigger - duplikaty w nazwie pozycji
21
22 CREATE OR REPLACE FUNCTION check_duplicate_position_names()
23 RETURNS TRIGGER AS $$
24 BEGIN
25     IF EXISTS (SELECT 1 FROM position WHERE position_name = NEW.position_name) THEN
26         RAISE EXCEPTION 'Position name % already exists', NEW.position_name;
27     END IF;
28     RETURN NEW;
29 END;
30 $$ LANGUAGE plpgsql;
31
32 CREATE TRIGGER trigger_prevent_duplicate_position_names
33 BEFORE INSERT ON position
34 FOR EACH ROW
35 EXECUTE FUNCTION check_duplicate_position_names();
36
```

```
1 INSERT INTO position(position_name) VALUES ('Weterynarz');
```

Data Output Messages Notifications

```
ERROR: Position name Weterynarz already exists
CONTEXT: PL/pgSQL function check_duplicate_position_names() line 4 at RAISE
```

trigger_check_unique_visit_date

Trigger ten sprawdza, czy na daną godzinę, dany weterynarz lub zwierzę ma już zapisaną wizytę. Jeśli tak, to nie pozwala na wstawienie wizyty. Uruchamiany jest przed operacją `INSERT` na tabeli `visit`.

```

38 -- Trigger - unikalność dat wizyt (dla zwierząt i osobno dla weterynarzy)
39
40 CREATE OR REPLACE FUNCTION check_unique_visit_date()
41 RETURNS TRIGGER AS $$
42 BEGIN
43     IF EXISTS ( -- vets
44         SELECT 1 FROM visit
45         WHERE vet_id = NEW.vet_id AND visit_date = NEW.visit_date
46     ) THEN
47         RAISE EXCEPTION 'A visit for this vet on this date and time already exists';
48     END IF;
49
50 IF EXISTS ( -- pets
51     SELECT 1 FROM visit
52     WHERE pet_id = NEW.pet_id AND visit_date = NEW.visit_date
53 ) THEN
54     RAISE EXCEPTION 'A visit for this pet on this date and time already exists';
55 END IF;
56 RETURN NEW;
57 END;
58 $$ LANGUAGE plpgsql;
59
60 CREATE TRIGGER trigger_check_unique_visit_date
61 BEFORE INSERT ON visit
62 FOR EACH ROW
63 EXECUTE FUNCTION check_unique_visit_date();

```

```

1 INSERT INTO visit
2 (visit_date, reason, pet_id, vet_id)
3 VALUES ('2019-04-20 09:00:00', 'Wizyta kontrolna', 3, 6);

```

Data Output Messages Notifications

ERROR: A visit for this vet on this date and time already exists
 CONTEXT: PL/pgSQL function check_unique_visit_date() line 7 at RAISE

Automatyzacja zadania (z wykorzystaniem jobów)

W ramach automatyzacji zadania za pomocą jobów, zainstalowano [pgAgent](#) za pomocą narzędzia StackBuilder (spakowaną wraz z PostgreSQL). Stworzono job [check_integrity](#), który sprawdza klucze Foreign Key wszystkich tabel – tzn. sprawdza, czy rzeczywiście istnieje rekord do którego odnosi się dany klucz. Tzw. Foreign Key Constraints, dodane w pierwszym etapie projektu, powinny automatycznie zapobiegać występowaniu sytuacji w której jakiś klucz obcy odnosi się do nieistniejącego rekordu, lecz ten Constraint może zostać (np. chwilowo) zniesiony i mogą być naniesione zmiany na danych w tym czasie. Job [check_integrity](#) zlicza ilość błędów dla danej relacji i zapisuje je do pliku .csv.

Stworzony job ma trzy kroki. W pierwszym, tworzona jest tabela pomocnicza job_log.

Create - Step

General Code SQL

Name

create_log_table

Enabled?

☒

Kind

☒ SQL ☐ Batch

Connection type

☒ Local ☐ Remote

Database

wet_clinic

Connection string

Please specify the connection string for the remote database server. Each parameter setting is in the form keyword = value. Spaces around the equal sign are optional. To write an empty value, or a value containing spaces, surround it with single quotes, e.g., keyword = 'a value'. Single quotes and backslashes within the value must be escaped with a backslash, i.e., \" and \\. For more information, please see the documentation on [libpq connection strings](#).

Close

Reset

Save

Create - Step

General Code SQL

```

1 DROP TABLE IF EXISTS job_log;
2 CREATE TABLE job_log (
3     log_id SERIAL PRIMARY KEY,
4     check_name TEXT,
5     log_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
6     errors_found INT
7 );

```


W drugim kroku, sprawdzane są klucze obce wszystkich tabel. Wyniki są zapisywane do tabeli pomocniczej.



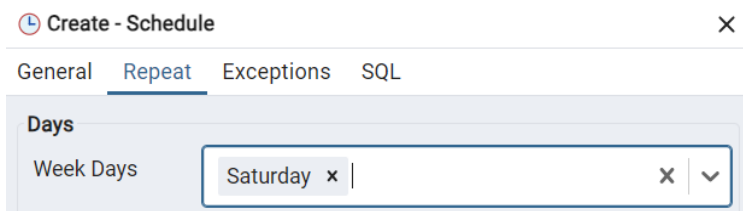
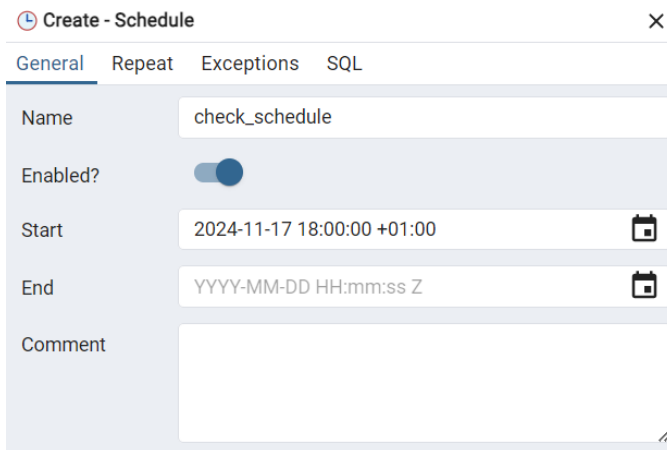
```
1
2 -- pet - species
3 WITH pet_check_species AS (
4     SELECT COUNT(*) AS n_species
5     FROM pet
6     WHERE species_id IS NOT NULL AND species_id NOT IN (SELECT species_id FROM species)
7 )
8 INSERT INTO job_log (check_name, log_time, errors_found)
9 SELECT 'Pet - FK Species check', CURRENT_TIMESTAMP, n_species
10 FROM pet_check_species;
11
12 -- pet - owner
13 WITH pet_check_owner AS (
14     SELECT COUNT(*) AS n_owner
15     FROM pet
16     WHERE owner_id IS NOT NULL AND owner_id NOT IN (SELECT owner_id FROM pet_owner)
17 )
18 INSERT INTO job_log (check_name, log_time, errors_found)
19 SELECT 'Pet - FK Owner check', CURRENT_TIMESTAMP, n_owner
20 FROM pet_check_owner;
21
22 -- pet measurement - pet
```

W trzecim kroku, wyniki zadania są zapisywane do pliku .csv.



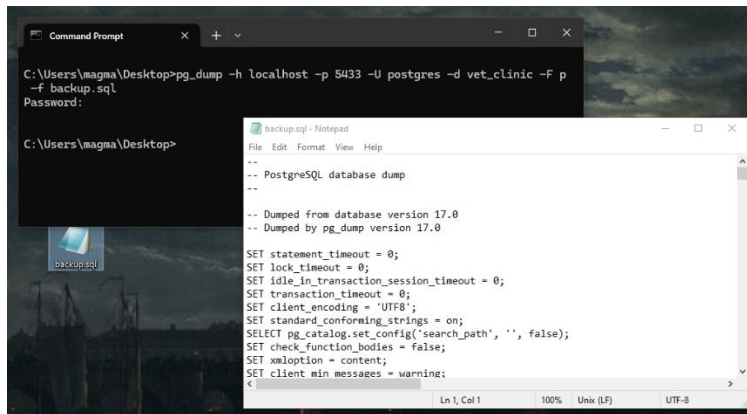
```
1
2 COPY job_log TO 'C:\Users\Public\integrity_check_results.csv' DELIMITER ',' CSV HEADER;
3
```

Następnie, ustalono kiedy i co jaki okres czasu zadanie będzie się wykonywać – raz w tygodniu, w sobotę.



Kopia zapasowa

Wykorzystano komendę `pg_dump` do stworzenia kopię zapasową bazy danych `vet_clinic`.



```
C:\Users\magma\Desktop>pg_dump -h localhost -p 5433 -U postgres -d vet_clinic -F p
-f backup.sql
Password:

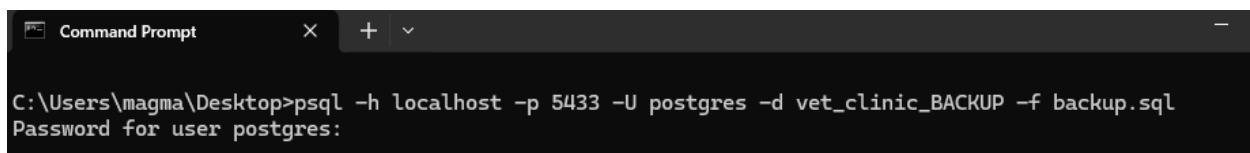
C:\Users\magma\Desktop>

backup.sql - Notepad
File Edit Format View Help
--
-- PostgreSQL database dump
--

-- Dumped from database version 17.0
-- Dumped by pg_dump version 17.0

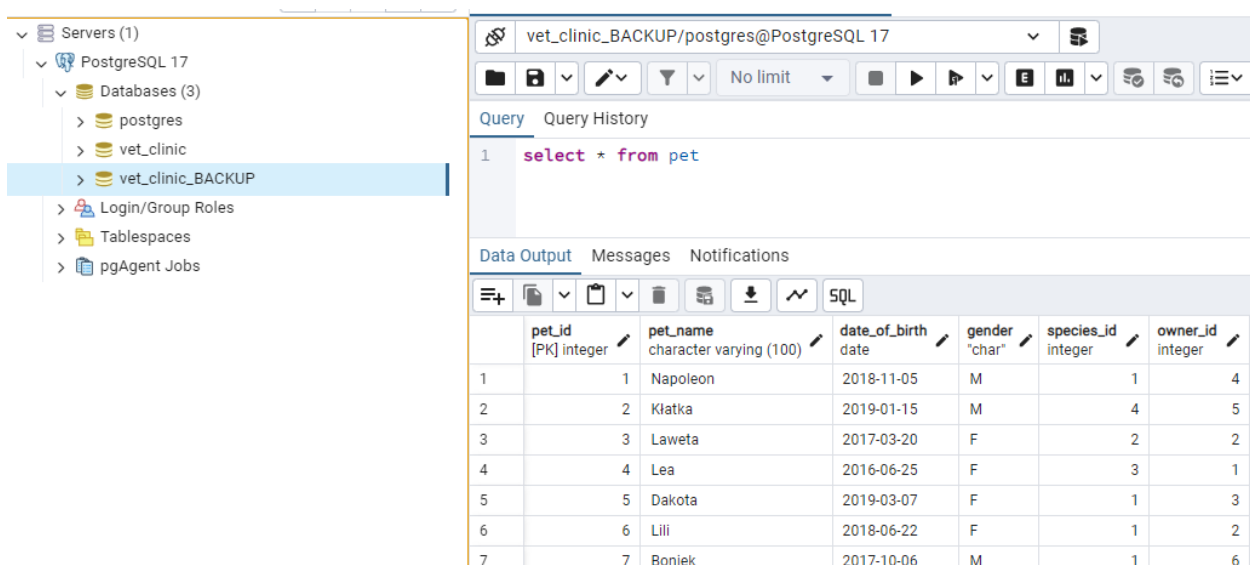
SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET transaction_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
<
```

Stworzono przy użyciu narzędzia pgAdmin nową bazę danych `vet_clinic_BACKUP` w celu sprawdzenia poprawności kopii zapasowej. Następnie, wykorzystano `backup.sql` do odtworzenia bazy danych `vet_clinic` w `vet_clinic_BACKUP`.



```
C:\Users\magma\Desktop>psql -h localhost -p 5433 -U postgres -d vet_clinic_BACKUP -f backup.sql
Password for user postgres:
```

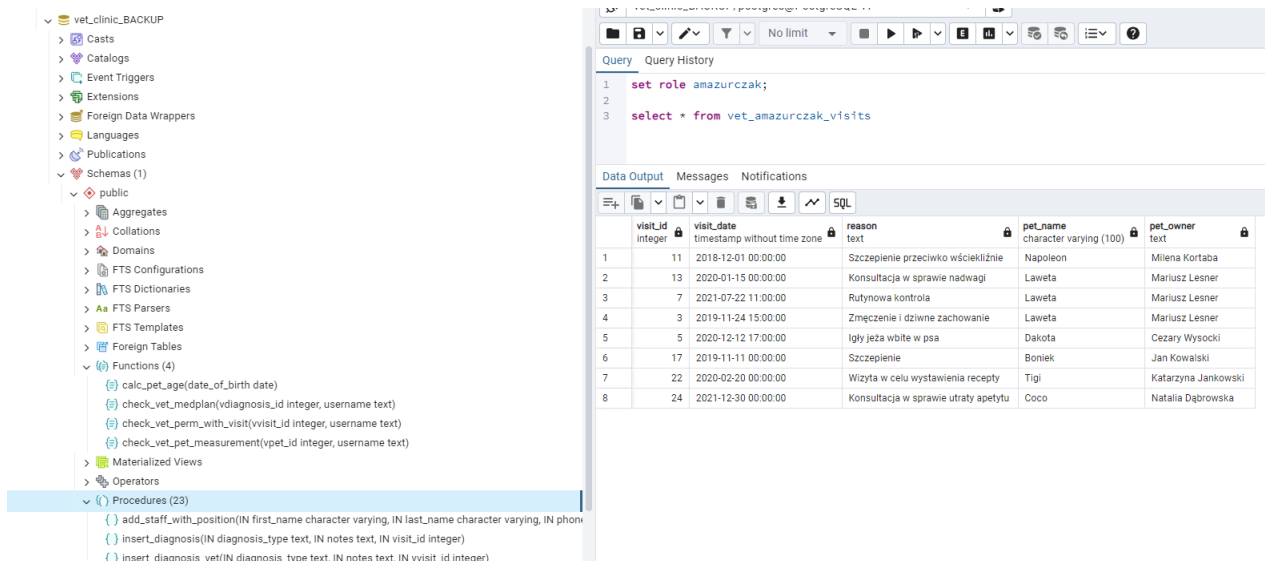
Sprawdzono, że tabele wraz z danymi zostały poprawnie odtworzone:



The screenshot shows the pgAdmin interface with the 'vet_clinic_BACKUP' database selected. The 'Query' tab is active, displaying the query `select * from pet`. The 'Data Output' tab shows the results of the query in a table format.

	pet_id [PK] integer	pet_name character varying (100)	date_of_birth date	gender "char"	species_id integer	owner_id integer
1	1	Napoleon	2018-11-05	M	1	4
2	2	Klatka	2019-01-15	M	4	5
3	3	Laweta	2017-03-20	F	2	2
4	4	Lea	2016-06-25	F	3	1
5	5	Dakota	2019-03-07	F	1	3
6	6	Lili	2018-06-22	F	1	2
7	7	Boniek	2017-10-06	M	1	6

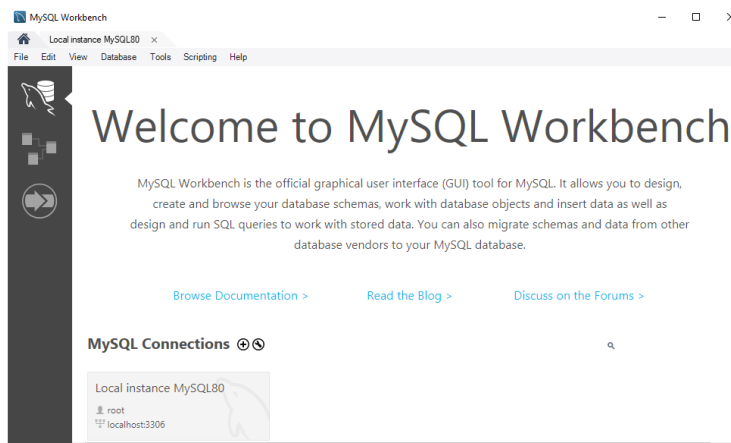
Inne elementy bazy, takie jak widoki, procedury, funkcje również zostały odtworzone:

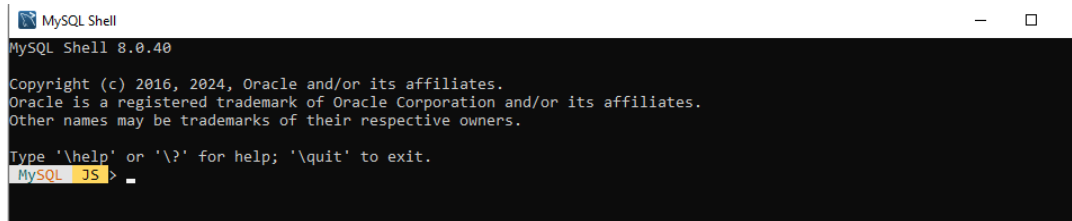


Przy użyciu komendy `pg_dump`, użytkownicy i ich uprawnienia nie są kopiowane do kopii zapasowej. Gdybym chciała odtworzyć moją bazę danych w innym klastrze, musiałabym za pomocą komendy `pg_dumpall` z flagą `-r` skopiować użytkowników do innego pliku i ich odtworzyć potem za pomocą komendy `psql` w terminalu (podobna struktura komendy to tej użytej przy odtworzeniu całej bazy). Alternatywą byłoby stworzenie kopii zapasowej całego klastra za pomocą komendy `pg_dumpall`, w której role byłyby już zawarte (ale stworzyłoby to też kopię zapasową wszystkich innych baz w klastrze). Ponieważ odtworzyłam bazę w tym samym klastrze co oryginalna, nie było potrzeby odtwarzania ról / użytkowników.

Przygotowanie innego systemu bazodanowego

Jako drugi system bazodanowy, wybrano MySQL. Pobrano i zainstalowano najnowszą wersję z strony producenta (8.0.40).





```
MySQL Shell
MySQL Shell 8.0.40

Copyright (c) 2016, 2024, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS >
```

Analiza różnic pomiędzy dialektami

Dialekty SQL PostgreSQL i MySQL są podobne, lecz istnieje kilka znaczących różnic.

- PostgreSQL posiada typ SERIAL służący do autoinkrementacji wartości danej kolumny, a MySQL ma słowo-klucz AUTO_INCREMENT. W bazie danych vet_clinic, SERIAL jest używany w kluczach prywatnych wszystkich tabel, a więc ich typ będzie musiał być zmieniony na INT AUTO_INCREMENT w ramach poprawek. MySQL, w przeciwieństwie do PostgreSQL, nie wspiera tzw. Identity Column, lecz ta funkcjonalność nie jest używana w mojej bazie i nie będzie wchodziła w zakres potrzebnych poprawek do migracji
- PostgreSQL wspiera typ Boolean a MySQL wspiera TINYINT(1). Niektóre funkcje w bazie vet_clinic używają lub zwracają typ Boolean, więc będzie on musiał być zmieniony
- Składnia procedur i funkcji różni się między PostgreSQL i MySQL. Np. w MySQL blok DECLARE jest poniżej słowa-klucz BEGIN, a w PostgreSQL jest powyżej. Ponadto, MySQL nie ma opcji SECURITY DEFINER i używa DELIMITER, podczas gdy PostgreSQL domyślnie używa \$\$.
- Obsługa błędów w PostgreSQL i MySQL się różni. W PostgreSQL, można użyć RAISE EXCEPTION, a w MySQL – SIGNAL SQLSTATE.

Narzędzia do migracji danych

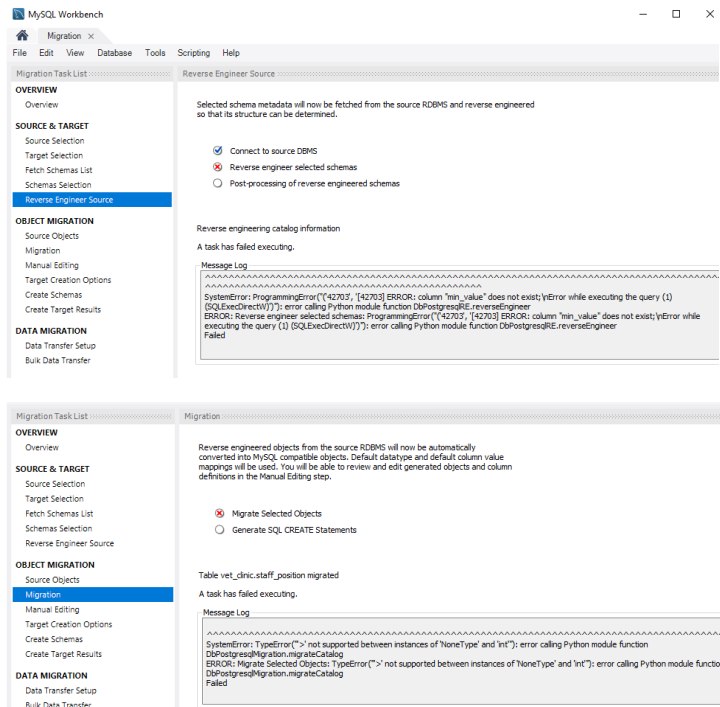
Narzędzia ETL:

- MySQL Workbench Migration Wizard – jest to narzędzie podpięte do MySQL Workbench. Ponieważ jest to narzędzie wbudowane już do Workbench’a, w pierwszej kolejności zdecydowałam się na wykonanie migracji za jego pomocą. Niestety okazało się wadliwe, a jedyne znalezione rozwiązania na problemy wiązały się z modyfikacją kodu źródłowego Workbench’a.
- Estuary Flow – jest to narzędzie ETL działające z przeglądarki. Niestety, jest to narzędzie bardziej przystosowane do pracy z bazami danych postawionymi na serwerach – gdy w ramach wstępnego zapoznania się z narzędziem chciałam przetestować połączenie z moją bazą, okazało się, że Estuary nie jest zbyt przystosowany do pracy z serwerem na localhost i próba połączenia zwróciła błąd. Jest to narzędzie płatne, lecz oferuje darmowy okres próbny.
- Talend – jest to bardzo szeroko rozbudowane narzędzie ETL. Również jest narzędziem płatnym, lecz też oferuje bezpłatny okres próby. Narzędzie to jest bardzo rozbudowane i jego funkcjonalności wykraczają szeroko poza kwestię migracji danych. Nie wybrano tego narzędzia z powodu wysokiego progu wstępu i braku przystępnej instrukcji do migracji danych za jego pomocą.

- DBeaver – jest to narzędzie do administracji baz danych. Posiada on bezpłatną wersję Community i dobrą dokumentację. Wybrano to narzędzie, ponieważ po wstępnym wglądzie w jej funkcjonalności, zadanie wykonania migracji okazało się bardzo przystępne.

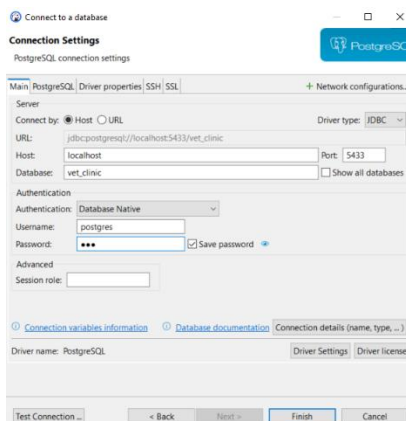
Migracja danych

Na początku, spróbowano przeprowadzić migrację za pomocą narzędzia MySQL Workbench Migration Wizard, lecz próby te kończyły się błędami z którymi nie mogłam sobie poradzić:



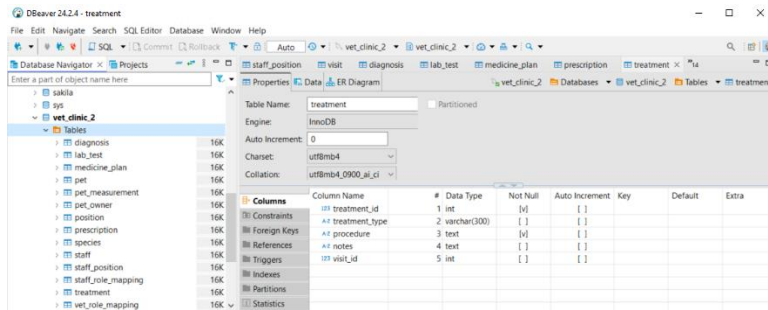
Spróbowano więc migracji za pomocą DBeaver. Skorzystano z instrukcji na oficjalnej stronie DBeaver: <https://dbeaver.com/docs/dbeaver/Data-migration/>

Połączono się z bazami danych w PostgreSQL i MySQL:



(Dla bazy danych MySQL wyglądało to analogicznie.)

Eksportowano dane z PostgreSQL do MySQL zgodnie z instrukcją. Przeniesiono dane do bazy vet_clinic_2 w MySQL dla zachowania czytelności:



Zweryfikowano, że oprócz definicji tabel, dane również zostały przeniesione:

```
1 • select * from vet_clinic_2.prescription;  
2
```

Result Grid				
prescription_id	prescription_date	medicine_name	medicine_amount	visit_id
1	2019-11-24	Lasix	10	3
2	2019-11-24	Benazepril	4	3
3	2019-11-24	Soloxine	14	3
4	2021-07-22	Lasix	20	7
5	2021-07-22	Benazepril	8	7
6	2021-07-22	Soloxine	28	7
7	2020-12-12	Amoksylicyna	1	5
8	2021-07-26	Amoksylicyna	8	8
9	2020-08-05	Benadryl	2	18
10	2020-02-20	Buspirone	4	22
11	2021-12-30	Amoksylicyna	1	24
12	2020-09-30	Albuterol	10	16
NULL	NULL	NULL	NULL	NULL

Migracja – poprawki

Migracja wymagała wprowadzenie szeregu poprawek:

- Klucz prywatny nie był oznaczony jako klucz prywatny
- Dodanie klauzuli NOT NULL do poszczególnych pól
- Dodanie opcji Auto-increment do ID

- Klucz obcy nie był oznaczony jako klucz obcy

```
1 -- select * from vet_clinic_2.species;  
2 • describe vet_clinic_2.species;
```

Field	Type	Null	Key	Default	Extra
species_id	int	NO		NULL	
species_name	varchar(100)	YES		NULL	

Wprowadzono odpowiednie poprawki za pomocą DBeaver (poprzez zapytania SQL):

```
1 -- select * from vet_clinic_2.species;  
2 • describe vet_clinic_2.species;  
3  
4 -- ALTER TABLE vet_clinic_2.species  
5 -- ADD PRIMARY KEY (species_id);  
6  
7 -- ALTER TABLE vet_clinic_2.species  
8 -- MODIFY species_id INT AUTO_INCREMENT;  
9  
10 -- ALTER TABLE vet_clinic_2.species  
11 -- MODIFY species_name varchar(100) NOT NULL;
```

Field	Type	Null	Key	Default	Extra
species_id	int	NO	PRI	NULL	auto_increment
species_name	varchar(100)	NO		NULL	

```

15
16 • ALTER TABLE vet_clinic_2.pet_measurement
17   MODIFY pet_id int NOT NULL;
18
19 • ALTER TABLE vet_clinic_2.pet_measurement
20   ADD FOREIGN KEY (pet_id) REFERENCES pet(pet_id)
21   ON UPDATE CASCADE
22   ON DELETE NO ACTION;
23
24
25
26

```

Result Grid

Field	Type	Null	Key	Default	Extra
measurement_id	int	NO	PRI	NULL	auto_increment
measurement_date	date	NO		NULL	
weight	decimal(5,2)	NO		NULL	
height	decimal(5,2)	NO		NULL	
pet_id	int	NO	MUL	NULL	

DBeaver skutecznie przeniósł tabele oraz dane, lecz nie przeniósł wszystkiego innego (widoków, procedur, triggerów, itp), więc trzeba było to zrobić ręcznie.

Należało stworzyć użytkowników:

```

3 • CREATE USER 'amazurczak'@'%' IDENTIFIED BY '123';
4 • CREATE USER 'jsierakowski'@'%' IDENTIFIED BY '123';
5 • CREATE USER 'switowska'@'%' IDENTIFIED BY '123';
6 • CREATE USER 'jmrotek'@'%' IDENTIFIED BY '123';
7 • CREATE USER 'ajanowicz'@'%' IDENTIFIED BY '123';
8
9 • GRANT USAGE ON *.* TO 'amazurczak'@'%';
10 • GRANT USAGE ON *.* TO 'jsierakowski'@'%';
11 • GRANT USAGE ON *.* TO 'switowska'@'%';
12 • GRANT USAGE ON *.* TO 'jmrotek'@'%';
13 • GRANT USAGE ON *.* TO 'ajanowicz'@'%';

```

Składnia do stworzenia widoków była taka sama jak dla bazy w PostgreSQL. Podobnie, składnia do utworzenia indeksów również była taka sama jak dla PostgreSQL. Składnia wyzwalaczy miała pewne pomniejsze różnice.

Należało zmodyfikować kod tworzący funkcję i procedury przed wstawieniem ich do bazy w MySQL:


```

1  DELIMITER $$
2
3  CREATE FUNCTION check_vet_perm_with_visit(vvisit_id INT, username VARCHAR(255))
4  RETURNS BOOLEAN
5  DETERMINISTIC
6  SQL SECURITY DEFINER
7  BEGIN
8      DECLARE current_vet_id INT;
9      DECLARE visit_exists BOOLEAN DEFAULT FALSE;
10
11      SELECT staff_id INTO current_vet_id
12      FROM staff_role_mapping
13      WHERE role_name = username;
14
15      IF current_vet_id IS NULL THEN
16          SIGNAL SQLSTATE '45000'
17          SET MESSAGE_TEXT = 'Role does not have an ID.';
18      END IF;
19
20      SELECT EXISTS (
21          SELECT 1
22          FROM visit

```

Output

#	Time	Action	Message
✓ 22	12:52:55	grant execute on function check_vet_perm_with_visit to jsierakowski	0 row(s) affected
✓ 23	12:52:55	grant execute on function check_vet_perm_with_visit to switowska	0 row(s) affected
✓ 24	12:52:55	grant execute on function check_vet_perm_with_visit to jmrotek	0 row(s) affected
✓ 25	12:52:55	grant execute on function check_vet_perm_with_visit to ajanowicz	0 row(s) affected
✓ 26	12:54:29	CREATE FUNCTION check_vet_perm_with_visit(vvisit_id INT, username VARCHAR(255)) RETURNS BOOLEAN	0 row(s) affected

```

1
2 • grant execute on function check_vet_perm_with_visit to amazurczak;
3 • grant execute on function check_vet_perm_with_visit to jsierakowski;
4 • grant execute on function check_vet_perm_with_visit to switowska;
5 • grant execute on function check_vet_perm_with_visit to jmrotek;
6 • grant execute on function check_vet_perm_with_visit to ajanowicz;
7
8
9

```

Output

#	Time	Action	Message
✓ 23	12:52:55	grant execute on function check_vet_perm_with_visit to switowska	0 row(s) affected
✓ 24	12:52:55	grant execute on function check_vet_perm_with_visit to jmrotek	0 row(s) affected
✓ 25	12:52:55	grant execute on function check_vet_perm_with_visit to ajanowicz	0 row(s) affected
✓ 26	12:54:29	CREATE FUNCTION check_vet_perm_with_visit(vvisit_id INT, username VARCHAR(255)) RETURNS BOOLEAN	0 row(s) affected
✓ 27	12:55:51	grant execute on function check_vet_perm_with_visit to amazurczak	0 row(s) affected
✓ 28	12:55:51	grant execute on function check_vet_perm_with_visit to jsierakowski	0 row(s) affected
✓ 29	12:55:51	grant execute on function check_vet_perm_with_visit to switowska	0 row(s) affected
✓ 30	12:55:51	grant execute on function check_vet_perm_with_visit to jmrotek	0 row(s) affected
✓ 31	12:55:51	grant execute on function check_vet_perm_with_visit to ajanowicz	0 row(s) affected

```

1  DELIMITER $$
2
3  CREATE PROCEDURE insert_pet_owner(
4      first_name VARCHAR(100),
5      last_name VARCHAR(100),
6      phone_number TEXT,
7      email_address VARCHAR(100)
8  ) SQL SECURITY DEFINER
9  BEGIN
10     IF first_name IS NULL OR first_name = '' THEN
11         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'First name cannot be null or empty.';
12     END IF;
13
14     IF last_name IS NULL OR last_name = '' THEN
15         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Last name cannot be null or empty.';
16     END IF;
17
18     IF phone_number IS NULL OR phone_number = '' THEN
19         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Phone number cannot be null or empty.';
20     END IF;
21
22     INSERT INTO pet_owner (first_name, last_name, phone_number, email_address)
23     VALUES (first_name, last_name, phone_number, email_address);
24
25 END $$
26
27 DELIMITER ;

1  DELIMITER $$
2  CREATE FUNCTION calc_pet_age(date_of_birth DATE)
3  RETURNS INT
4  DETERMINISTIC
5  BEGIN
6      DECLARE pet_age INT;
7      IF date_of_birth IS NULL THEN
8          RETURN NULL;
9      END IF;
10     SET pet_age = TIMEDIFF(YEAR, date_of_birth, CURDATE());
11     RETURN pet_age;
12 END $$
13 DELIMITER ;

```

Nie wszystkie wnioski wyciągnięte w analizie różnic okazały się poprawne. Rzeczywiście, składnia funkcji i procedur są inne, lecz okazało się, że MySQL pozwala na użycie Boolean – jest to synonim TINYINT(1).

Poprawne działanie funkcjonalności po migracji

Po wprowadzeniu poprawek, zweryfikowano poprawne działanie wszystkich funkcjonalności.

Trigger

Zweryfikowano działanie wszystkich trzech zaimplementowanych triggerów:

```

1
2 • INSERT INTO pet
3   (pet_name, date_of_birth, gender, species_id, owner_id)
4   VALUES
5   ('Cezar', '2019-11-08', 'm', 1, 4);
6
7 • select * from pet where pet_name = 'Cezar';
8

```

result Grid | | Filter Rows: | Edit: | Export/Imp

pet_id	pet_name	date_of_birth	gender	species_id	owner_id
37	Cezar	2019-11-08	M	1	4
NULL	NULL	NULL	NULL	NULL	NULL

```

1
2 • INSERT INTO position
3   (position_name)
4   VALUES
5   ('Weterynarz');
6

```

Output

Action Output

#	Time	Action	Message
1	20:39:39	INSERT INTO pet (pet_name, date_of_birth, gender, species_id, owner_id) VALUES ('Cezar', '2019-11-08', 'm', 1, 4)	1 row(s) affected
2	20:39:54	select * from pet where pet_name = 'Cezar' LIMIT 0, 1000	1 row(s) returned
3	20:40:35	select * from pet LIMIT 0, 1000	37 row(s) returned
4	20:41:41	INSERT INTO position (position_name) VALUES ('Weterynarz')	Error Code: 1644. Position name already exists

```

2 • INSERT INTO visit
3   (visit_date, reason, pet_id, vet_id)
4   VALUES
5   ('2019-04-20 09:00:00', 'Szczepienie psa', 1, 6);
6

```

Output

Action Output

#	Time	Action	Message
✓ 1	20:39:39	INSERT INTO pet (pet_name, date_of_birth, gender, species_id, owner_id) VALUES ('Cezar', '2019-11-08', 'm', 1, 4)	1 row(s) affected
✓ 2	20:39:54	select * from pet where pet_name = 'Cezar' LIMIT 0, 1000	1 row(s) returned
✓ 3	20:40:35	select * from pet LIMIT 0, 1000	37 row(s) returned
✗ 4	20:41:41	INSERT INTO position (position_name) VALUES ('Weterynarz')	Error Code: 1644. Position name already exists
✗ 5	20:42:43	INSERT INTO visit (visit_date, reason, pet_id, vet_id) VALUES ('2019-04-20 09:00:00', 'Szczepienie psa', 1, 6)	Error Code: 1644. A visit for this vet on this date and time already exists

UDF

Zweryfikowano działanie zaimplementowanej funkcji:

```
1
2 • select pet_name, calc_pet_age(date_of_birth) from pet;
3
4
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
pet_name	calc_pet_age(date_of_birth)			
Napoleon	6			
Kłatka	5			
Laweta	7			
Lea	8			
Dakota	5			
Lili	6			
Boniek	7			

Procedura

Zweryfikowano działanie zaimplementowanej procedury:

```
2 • call add_staff_with_position('Agnieszka','Nowakowska','111-567-899',null,'Księgowa','2019-02-01',null);
3 • select * from staff where first_name='Agnieszka';
4
5
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
staff_id	first_name	last_name	phone_number	specialization	
16	Agnieszka	Nowakowska	111-567-899	NULL	
NULL	NULL	NULL	NULL	NULL	

```
2 • call add_staff_with_position('Agnieszka','Nowakowska','111-567-899',null,'Księgowa','2019-02-01',null);
3 • select * from position;
4
5
```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
position_id	position_name				
1	Recepcjonist/ka				
2	Weterynarz				
3	Sprzątac				
4	Technik sprzętowy				
11	Ochroniarz				
12	Księgowa				
NULL	NULL				

Magdalena Markowicz 310836

Widok użytkownika

Zweryfikowano widoki użytkowników:

```
1 • select * from vet_clinic_2.vet_amazurczak_visits
```

visit_id	visit_date	reason	pet_name	pet_owner
3	2019-11-24 15:00:00	Zmęczenie i dziwne zachowanie	Laweta	Mariusz Lesner
5	2020-12-12 17:00:00	Igły jeża wbite w psa	Dakota	Cezary Wysocki
7	2021-07-22 11:00:00	Rutynowa kontrola	Laweta	Mariusz Lesner
11	2018-12-01 00:00:00	Szczepienie przeciwko wściekliznie	Napoleon	Milena Kortaba
13	2020-01-15 00:00:00	Konsultacja w sprawie nadwagi	Laweta	Mariusz Lesner
17	2019-11-11 00:00:00	Szczepienie	Boniek	Jan Kowalski
22	2020-02-20 00:00:00	Wizyta w celu wystawienia recepty	Tigi	Katarzyna Jankowski
24	2021-12-30 00:00:00	Konsultacja w sprawie utraty apetytu	Coco	Natalia Dąbrowska

Procedura wstawiania

Zweryfikowano również procedury wstawiania danych:

```
1 • call vet_clinic_2.insert_pet_measurement_vet('2020-04-20',39.3,70.4,1);
2 • select * from vet_clinic_2.vet_amazurczak_pets;
```

pet_id	pet_name	species	date_of_birth	gender	latest_weight	latest_height	pet_owner
5	Dakota	Pies	2019-03-07	F	32.30	75.20	Cezary Wysocki
3	Laweta	Kot	2017-03-20	F	9.75	32.40	Mariusz Lesner
1	Napoleon	Pies	2018-11-05	M	39.30	70.40	Milena Kortaba