Politechnika Wrocławska

Wydział Elektroniki

Bazy Danych 1

Baza danych MySQL na przykładzie serwisu samochodowego

Michał Droń 248832
Wiktor Danielewski 252804
Michał Maziec 252718

Prowadzący
mgr inż. Szymon Wojciechowski

11 czerwca 2021

Spis treści

1 Wstęp

1.1 Tematyka projektu

Projekt systemu dotyczy organizacji pracy oraz niezbędnej bazy danych w serwisie samochodowym. Osobą zamawiającą jest właściciel serwisu, lecz sama baza będzie dostępna (w różnym stopniu) również dla pracowników oraz klientów wspomnianego serwisu.

1.2 Założenia projektowe

1.2.1 Cele projektu

System tworzymy, aby wprowadzić ułatwienia w przedsiębiorstwie (serwisie samochodowym):

- Usprawnienie komunikacji firma-klient, np. poprzez możliwość sprawdzenia, czy samochód klienta jest już naprawiony.
- Ułatwienie zarządzania firmą dla właściciela, np. poprzez kontrolowanie bieżących wydatków w formie elektronicznej.
- Ułatwienie komunikacji właściciel-pracownik, np. poprzez przydzielanie zadań w bazie danych.

1.2.2 Aktorzy

Aktorem z największymi uprawnieniami będzie **właściciel serwisu samochodowego**. Kolejnym znaczącym aktorem będzie **pracownik serwisu**, którego zadania/możliwości zostaną wymienione w liście przypadków użycia. Ostatnim przewidywanym aktorem jest **klient**, który będzie miał dostęp np. do statusu realizowanej usługi.

1.3 Wybrane narzędzia

1.3.1 MySQL

MySQL to wolnodostępny, otwarto-źródłowy system zarządzania relacyjnymi bazami danych.
Wybraliśmy go, ponieważ jest łatwo dostępny, przyjazny dla początkujących użytkowników oraz oferuje szereg narzędzi, które ułatwiają pracę, np. interfejs graficzny dla osób, które nie czują się pewnie z obsługą konsoli.

1.3.2 MySQL Workbench

MySQL Workbench jest graficznym narzędziem do tworzenia baz danych, które oferuje możliwość: tworzenia w SQL, administracji, modyfikacji baz danych oraz swobodną edycję bazującą na systemie MySQL.

1.3.3 Mockaroo

Strona internetowa, która umożliwia generowanie danych losowych do baz danych. Okazała się być bardzo elastyczna i zawierała dużą bazę domyślnych kategorii danych, które często można było bezpośrednio wprowadzić do naszej bazy danych. Umożliwiała również generowanie danych losowych w formie gotowych zapytań.

1.3.4 Visual Paradigm

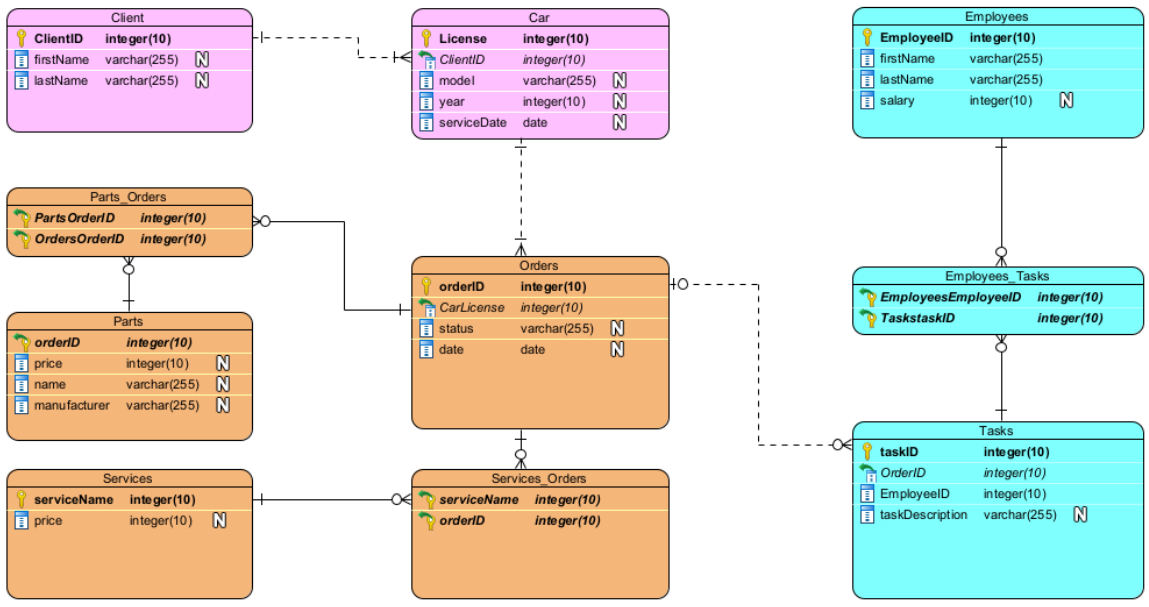
Diagramy ERD zostały utworzone w aplikacji Visual Paradigm.

2 Architektura

Bazując na głównej tematyce projektu oraz określonych przypadkach użycia, utworzyliśmy 10 tabel, które możemy podzielić na 3 pomniejsze kategorie:

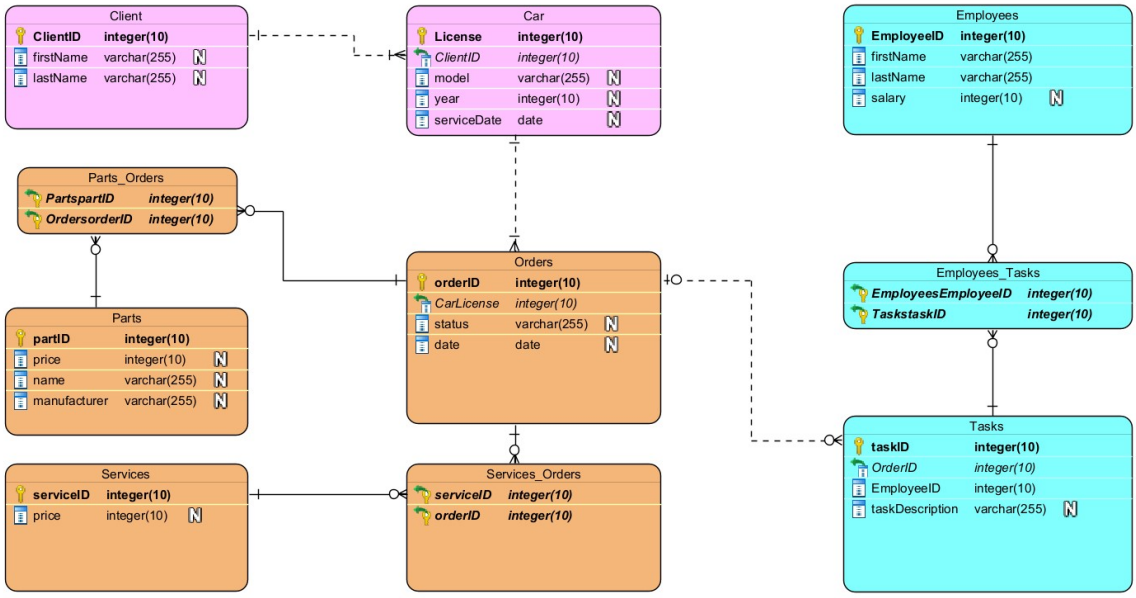
- Różową - zawiera dane dotyczące klientów oraz samochodów (które są przynależne do danych klientów)
- Pomarańczową - zawiera dane dotyczące: części, zamówień oraz usług oferowanych przez warsztat
- Turkusową - zawiera dane dotyczące pracowników oraz zadań, które muszą wykonać.

2.1 ERD planowane



Rysunek 1: Pierwsza wersja ERD.

2.2 ERD rzeczywiste



Rysunek 2: Finalna wersja wersja ERD.

2.3 Przypadki użycia

Aktor	Przypadek Użycia
Właściciel	1. Sprawdzenie, jakie części należy kupić
	2. Edycja listy oferowanych usług
	3. Edycja danych pracowników serwisu
	4. Przydzielanie zadań pracownikom
Pracownik	5. Edycja listy części do kupienia
	6. Ustawienie statusu realizacji danej usługi
	7. Przeglądanie zadań przypisanych do siebie
	8. Przeglądanie części dostępnych w warsztacie
	9. Aktualizacja danych związanych z przeglądem pojazdu
	10. Aktualizacja danych klientów serwisu
	11. Aktualizacja usług wchodzących w skład zamówienia
Klient	12. Sprawdzenie oferowanych przez serwis usług
	13. Sprawdzenie daty ostatniego przeglądu
	14. Sprawdzenie stanu realizacji usługi

3 Implementacja

3.1 Tworzenie bazy

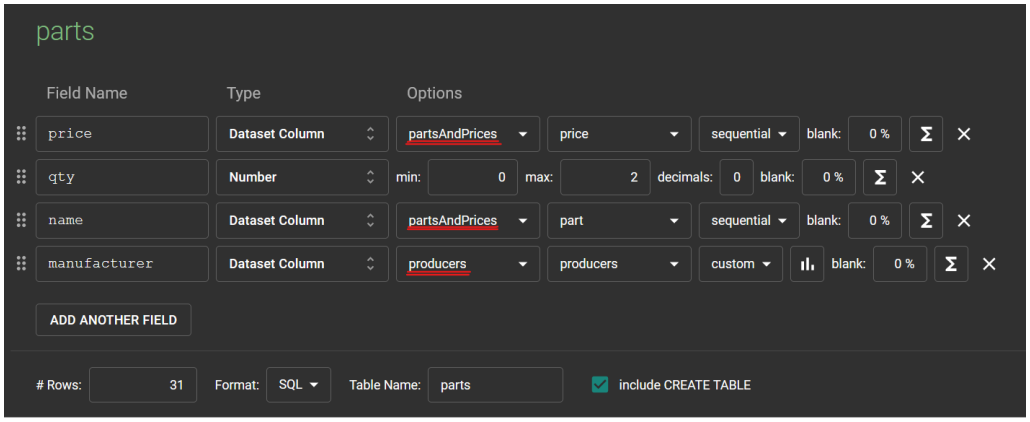
Wygenerowaliśmy skrpyt do stworzenia naszej bazy danych po stworzeniu diagramu ERD w programie Visual Paradigm. Sprawdziliśmy przy tym poprawność wygenerowanego kodu oraz dokonaliśmy kilku zmian kosmetycznych, które finalnie były przyczyną powstania drugiej, udoskonalonej wersji diagramu ERD. Dodatkowo dla naszej bazy został zapewniony skrypt, który wypełnia ją danymi, które mogłyby się pojawić w trakcie funkcjonowania serwisu samochodowego.

```
CREATE TABLE Tasks (
    taskID int(10) NOT NULL AUTO_INCREMENT,
    OrderID int(10) NOT NULL,
    EmployeeID int(10) NOT NULL,
    taskDescription varchar(255),
    PRIMARY KEY (taskID)
);
ALTER TABLE Tasks ADD CONSTRAINT FKTasks77706
FOREIGN KEY (OrderID) REFERENCES Orders (orderID);
```

Listing 1: Przykładowe zapytanie, tworzące tabelę Tasks (zadania)

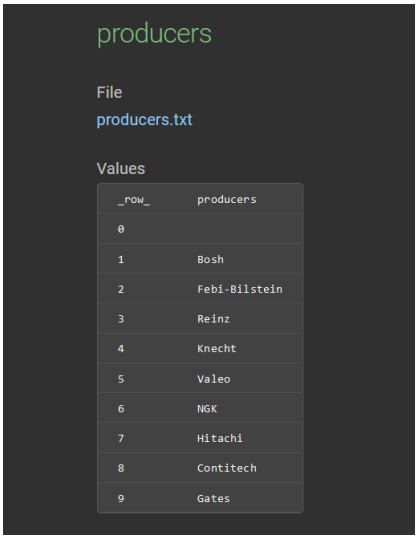
3.2 Dane

Skrypt z danymi startowymi do bazy danych był w szczególności tworzony przy użyciu strony internetowej Mockaroo. Strona ta posiada liczną bazę kategorii domyślnych oraz umożliwia generowanie gotowych zapytań, na podstawie stworzonej przez nas na stronie tabeli, która ma odzwierciedlać w mniejszym lub większym stopniu interesującą nas relacje w stworzonej przez nas bazie danych. Tabele można zapisywać, co jest wielkim atutem, ponieważ umożliwia nam to tworzenie tabel z kluczami obcymi. Na stronie możemy również dodawać swoje własne zbiory danych w postaci pliku csv, co jest bardzo przydatne, gdy nasz atrybut jest specyficzny i nie możemy znaleźć odpowiedniej dla niego kategorii. Na poniższym obrazku możemy zobaczyć przykład utworzonej tabeli na stronie, która ma odzwierciedlać relację w naszej bazie danych o nazwie *parts*.



Rysunek 3: Odpowiednik relacji *parts* utworzony na stronie

Podkreślone na czerwono słowa to nazwy zbiorów danych, które zostały wprowadzone przez nas poprzez pliki csv. Jeden plik csv może, ale nie musi zawierać więcej niż jedną kolumnę, dlatego też, zaraz obok mamy do wyboru interesująca nas w danym momencie kolumnę. Na obrazku poniżej przykład utworzonego przez nas zbioru o nazwie *producers*.



Rysunek 4: Zbiór danych o nazwie *producers*

Strona pobrała od nas plik *producers.txt* (z odpowiednio sformatowanymi danymi) i właściwie go zinterpretowała, dzięki temu uzyskaliśmy swój własny zbiór danych, którego możemy używać do wypełniania atrybutów w relacjach.

Gdy nasza tabela jest już gotowa, strona zapyta się o liczbę danych, jaką chcemy uzyskać, w jakim formacie chcemy mieć te dane (m.in. sql) oraz umożliwi nam wygenerowanie zapytania, które pozwala na utworzenie odpowiednika tabeli (stworzonego na stronie), w naszej bazie danych.

Preview

```
create table parts (
        price VARCHAR(3),
        qty INT,
        name  VARCHAR(3),
        manufacturer VARCHAR(2)
);
insert into parts (price, qty, name , manufacturer) values (4000, 2, 'Shor Engine', 'Febi-Bilstein');
insert into parts (price, qty, name , manufacturer) values (2000, 1, 'Engine block', 'Valeo');
insert into parts (price, qty, name , manufacturer) values (250, 1, 'Engine block mounting parts', 'Febi-Bilstein');
insert into parts (price, qty, name , manufacturer) values (100, 2, 'Timing case', 'NGK');
insert into parts (price, qty, name , manufacturer) values (400, 1, 'Oil level indicator', 'Gates');
insert into parts (price, qty, name , manufacturer) values (1100, 0, 'Cylinder head', 'Febi-Bilstein');
insert into parts (price, qty, name , manufacturer) values (25, 0, 'Cylinder head attached parts', 'Hitachi');
insert into parts (price, qty, name , manufacturer) values (1200, 1, 'Cylinder head cover', 'Febi-Bilstein');
insert into parts (price, qty, name , manufacturer) values (200, 1, 'Belt Drive-Vibration Damper', 'Bosh');
insert into parts (price, qty, name , manufacturer) values (200, 2, 'Belt Drive Water Pump/Alternator', 'Bosh');
```

Rysunek 5: Przykład wygenerowanych danych w formacie sql

Najczęściej wygenerowany zapytanie tworzące tabelę nie jest zbytnio przydatne, ponieważ zawiera wiele niedociągłości, a więc nie korzystaliśmy z niego.

3.3 Zapytania testowe

Do każdego przypadku użycia zostało stworzone zapytanie testowe. Niektóre wymagały jedynie insertów do tabel obsługujących połączenia wiele do wielu, inne to bardziej skomplikowane zapytania korzystające z podzapytań i agregacji.

```
SELECT
        Parts.name AS "Part",
        Parts.manufacturer AS "Manufacturer"
FROM Parts_Orders
    LEFT JOIN Parts
    ON Parts.partID = Parts_Orders.PartspartID
    WHERE Parts_Orders.OrdersorderID IN (
        SELECT orderID
        FROM Orders
        WHERE status <> 'Repaired'
    ) AND Parts.qty=0 ;
```

Listing 2: Przykładowe zapytanie testowe - PU nr 1 (Sprawdzenie, jakie części należy kupić)

```sql
SELECT
        Parts.name AS "Part",
        Parts.manufacturer AS "Manufacturer"
FROM Parts_Orders
    LEFT JOIN Parts
    ON Parts.partID = Parts_Orders.PartspartID
    WHERE Parts_Orders.OrdersorderID IN (
        SELECT orderID
        FROM Orders
        WHERE status <> 'Repaired'
    ) AND Parts.qty=0 ;
```

Listing 3: Przykładowe zapytanie testowe - PU nr 7 (Przeglądanie zadań przypisanych do siebie)

4 Podsumowanie

Uważamy, że wykonaliśmy nasze zadanie poprawnie, ponieważ udało nam się uzyskać zamierzony rezultat i spełnić wszystkie warunki określone przez prowadzącego. Sama baza danych działa poprawnie oraz wszystkie przypadki użycia zostały zaimplementowane poprawnie, o czym świadczy chociażby zatwierdzenie ich funkcjonalności podczas jednego ze spotkań konsultacyjnych.

4.1 Wprowadzone zmiany

Różnice pomiędzy pierwotną a finalną wersją diagramu ERD nie są aż tak diametralne. Główną przyczyną naniesionych zmian była niezgodność nazw kluczy oraz drobna nieścisłość w jednej z relacji, które wykryliśmy podczas generowania danych do naszej bazy. Przykładowo, w przypadku nazewnictwa dokonaliśmy następujących zmian:

1. Zamiana **PartsOrderID** na **PartspartID** w tabeli Parts_Orders
2. Zamiana **orderID** na **partID** w tabeli Parts
3. Zamiana **serviceName** na **serviceID** w tabeli Services

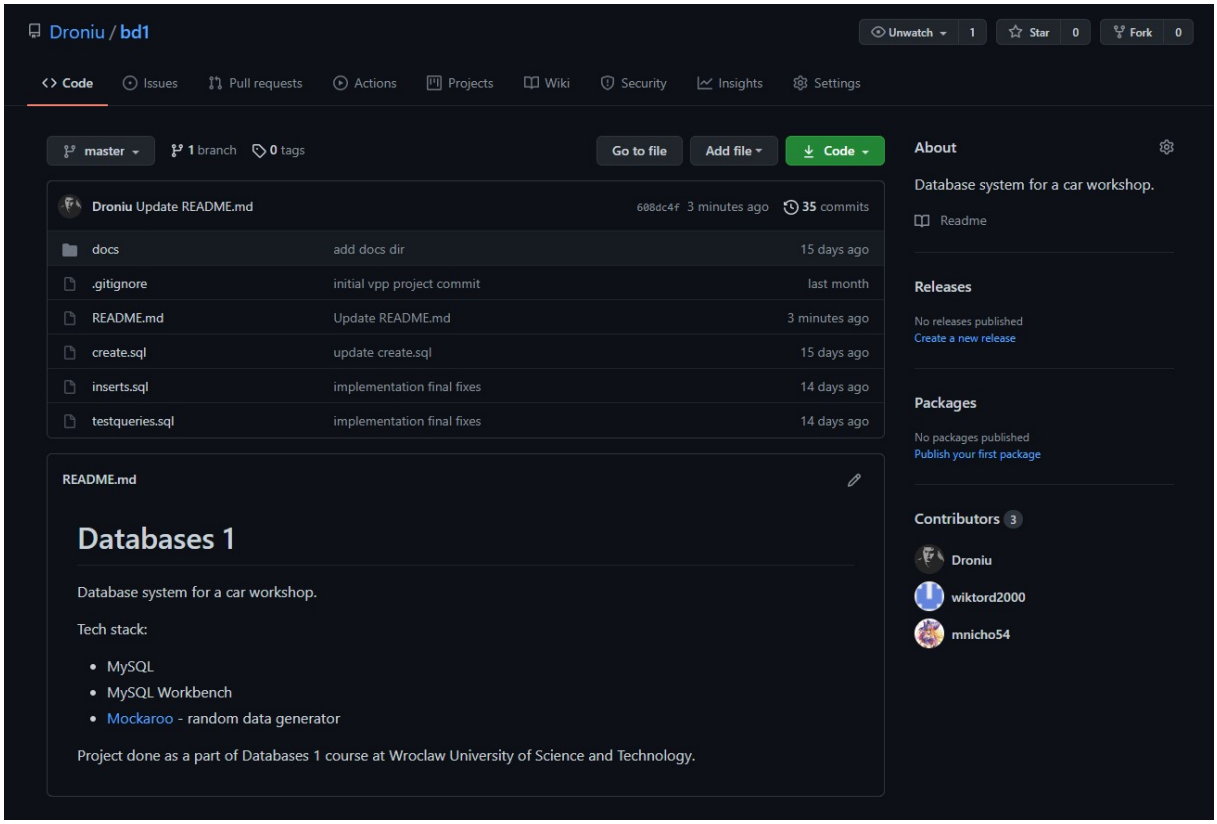
4.2 Możliwość rozbudowy

Wykreowana przez nas baza danych jest otwarta na dalszą rozbudowę. Zwróciliśmy na to uwagę, ponieważ jest bardzo ważny aspekt tego typu bazy, szczególnie w sektorze biznesowym. Przykładową możliwością rozbudowy mogłaby być, np. implementacja odpowiednich typów bądź rang pracowników. Niemniej, uważamy, że baza danych w obecnej wersji spełnia swoją zamierzoną funkcjonalność, która już pozwala na dokonanie naprawdę wielu czynności (zawartych chociażby w przypadkach użycia).

4.3 Wnioski

Utworzona przez nas baza danych spełniła wcześniej założone przypadki użycia. Do jej implementacji w szczególności pomocny okazał się być program Visual Paradigm, który umożliwił nam, na podstawie gotowego diagramu ERD, wygenerować zapytania ją tworzące. Oczywiście zapytania te nie były idealne, ale były dobrym fundamentem, na którym mogliśmy swobodnie pracować i ulepszać nasz projekt.

Cały projekt dostępny jest na serwisie GitHub pod linkiem:
`https://github.com/Droniu/bd1`



Rysunek 6: Repozytorium projektu na GitHubie.

5. Źródła

[1] Wykłady dr inż. Dariusza Jankowskiego

[2] Dokumentacja MySQL 8.0
`https://dev.mysql.com/doc/refman/8.0/en/`

[3] Generator danych
`https://www.mockaroo.com/`

[4] Podręcznik użytkownika Visual Paradigm
`https://www.visual-paradigm.com/support/documents/`