

Tomasz Mycielski

Zaawansowane systemy baz danych – projekt

Case study: Firma transportowa Eltrans

Eltrans to prężnie działająca firma transportowa, operująca zarówno na rynku krajowym, jak i międzynarodowym. Firma dysponuje liczną flotą, z czego większość stanowią tiry. Eltrans zatrudnia wielu pracowników, w większości kierowców, lecz także personel administracyjny, mechaników oraz specjalistów ds. logistyki.

Głównym wyzwaniem dla Eltrans jest efektywne zarządzanie flotą pojazdów i personelem. Firma potrzebuje narzędzia do monitorowania wydajności, optymalizacji tras, kontroli kosztów oraz wykrywania nadużyć (fraudów). Dodatkowo, Eltrans chce wzmocnić swoje możliwości analityczne, aby podejmować trafniejsze decyzje biznesowe oparte na danych.

Baza danych, która ma usprawnić działanie firmy, będzie centralnym punktem gromadzenia i analizy informacji o pracownikach, pojazdach, trasach i historycznym stanie pojazdów podczas całego cyklu ich eksploatacji. Dzięki niej, Eltrans będzie w stanie:

- Analizować pracę kierowców
- Monitorować wykorzystanie pojazdów
- Analizować opłacalność poszczególnych tras i zleceń, z uwzględnieniem specyfiki różnych typów pojazdów
- Generować raporty i analizy wspierające podejmowanie decyzji strategicznych
- Wykrywać nadużycia, których dopuszczają się pracownicy firmy

Wdrożenie tego systemu pozwoli Eltrans na zwiększenie efektywności operacyjnej, redukcję kosztów oraz poprawę jakości świadczonych usług. To z kolei przełoży się na wzmocnienie pozycji firmy na konkurencyjnym rynku transportowym, umożliwiając jej lepsze wykorzystanie zróżnicowanej floty pojazdów i efektywniejsze zarządzanie zasobami ludzkimi, oraz oczywiście zwiększenie zysków.

RDBMS

Na potrzeby tego projektu do zarządzania relacyjną bazą danych wykorzystany zostanie system **PostgreSQL**. Używałem tego systemu już wcześniej i nie miałem z nim problemu. Postgres ma ogromną społeczność, więc nawet jeśli napotkam jakiś problem, to na pewno szybko znajdę jego rozwiązanie w internecie. Korzystanie z Postgresa jest darmowe (open source), a instalacja jest dziecinnie prosta. Korzystam z **helm** (chart od Bitnami) do utworzenia instancji Postgresa na moim klastrze **k3s**, do którego dostęp mam przez WireGuard (Tailscale).

Screenshoty z działającego Postgresa:

```
postgres=# \l
```

List of databases								
Name	Owner	Encoding	Locale Provider	Collate	Ctype	Locale	ICU Rules	Access privileges
postgres	postgres	UTF8	libc	en_US.UTF-8	en_US.UTF-8			
template0	postgres	UTF8	libc	en_US.UTF-8	en_US.UTF-8			=c/postgres +
template1	postgres	UTF8	libc	en_US.UTF-8	en_US.UTF-8			postgres=CTc/postgres +
								=c/postgres +
								postgres=CTc/postgres

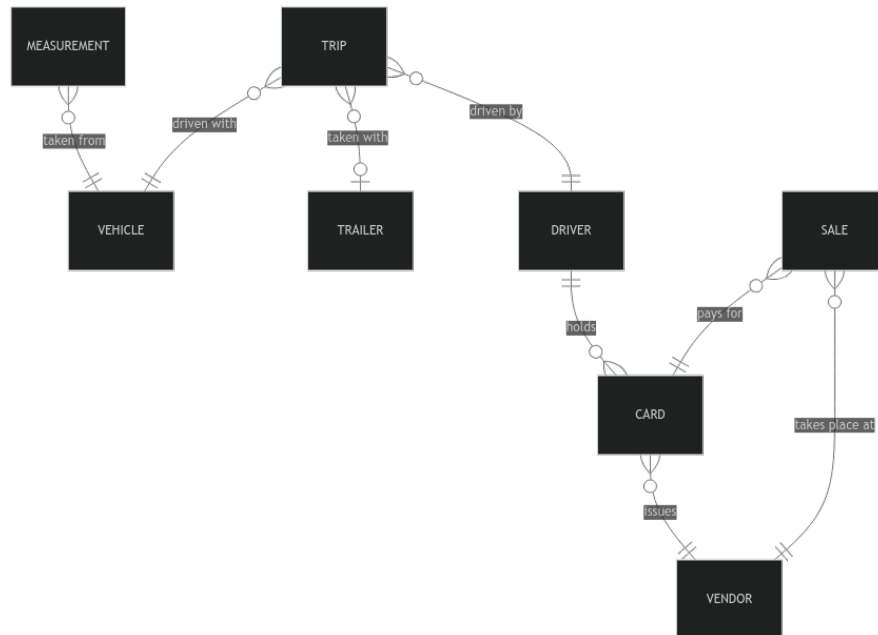
(3 rows)

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
postgres-postgresql-0	1/1	Running	0	11m

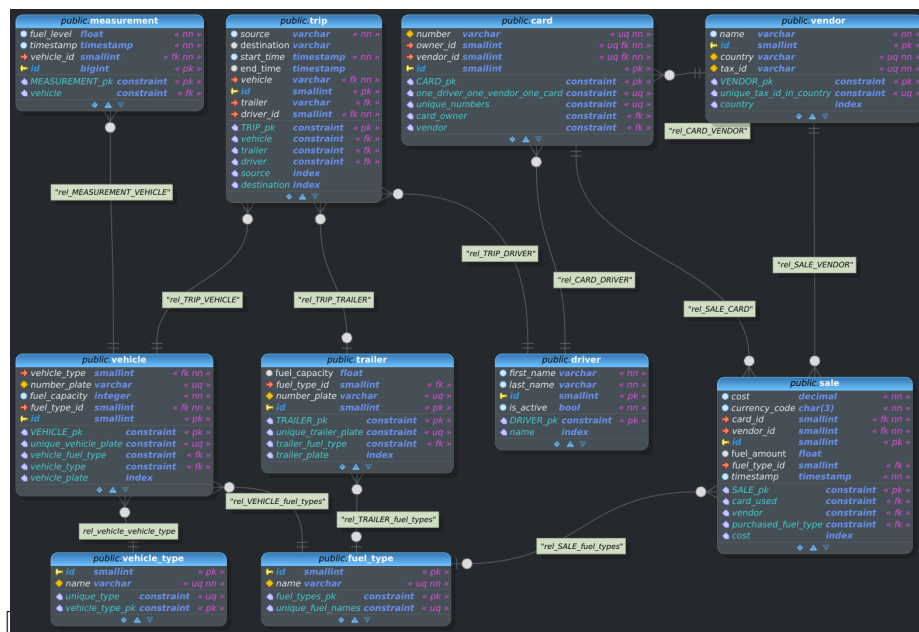
Projekt bazy danych

ERD



Implementacja bazy danych

Schemat logiczny



Wszystkie tabele są w jednym schemacie bazodanowym.

Tabele niezależne

1. measurement

Zawiera pomiary poziomu paliwa podczas jazdy samochodu.

2. trip

Zawiera informacje o trasach pokonanych przez samochody.

3. card

Zawiera informacje o kartach flotowych, których używają kierowcy podczas transakcji na stacjach paliw.

4. vendor

Zawiera informacje o dostawcach paliwa, z którymi firma ma podpisane umowy.

5. vehicle

Zawiera informacje o samochodach, którymi operuje firma.

6. trailer

Zawiera informacje o naczepach, którymi operuje firma.

7. **driver**

Zawiera informacje o kierowcach zatrudnionych w firmie.

8. **sale**

Zawiera informacje o transakcjach zakupu paliwa dokonanych przez kierowców.

Tablice słownikowe

1. **vehicle_type**
2. **fuel_type**

Odstępstwa od 3NF

- W tabeli **trip**:
 - **source** i **destination** są przechowywane jako varchar zamiast referencji do tabeli **locations**

Uzasadnienie:

- Uniknięcie skomplikowanych joinów przy wyszukiwaniu tras
- Lokalizacje mogą być bardzo zmienne i nie zawsze standardowe (np. adresy klientów)
- Dodano indeksy hash aby przyspieszyć wyszukiwanie po tych kolumnach

- W tabeli **sale**:
 - **currency_code** jest przechowywane bezpośrednio zamiast referencji do tabeli walut

Uzasadnienie:

- Lista walut jest względnie stała (ISO 4217)
- Uniknięcie dodatkowego joina przy każdym zapytaniu o sprzedaż
- Typ **char(3)** zapewnia poprawny format kodu waluty

- W tabeli **vendor**:
 - **country** jest przechowywane jako varchar zamiast referencji do tabeli krajów

Uzasadnienie:

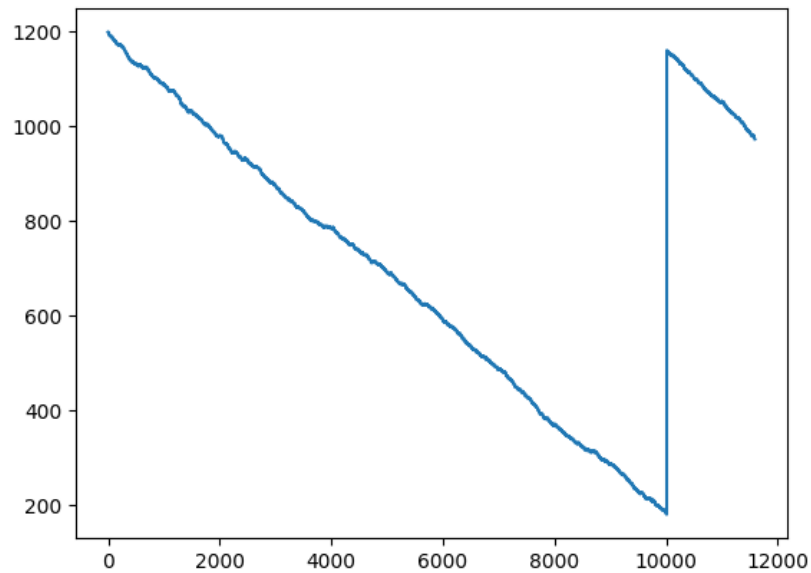
- Lista krajów jest względnie stała
- Często używane w zapytaniach (dodano indeks hash)
- Uniknięcie dodatkowego joina przy wielu zapytaniach

Istotne decyzje projektowe

- Użycie `number_plate` jako klucza obcego w powiązaniach pojazdów i naczep z przejazdami:
 - Zamiast używać `id`, użyłem numeru rejestracyjnego jako klucza obcego
 - Uzasadnienie: Szybsze wyszukiwanie po numerach rejestracyjnych (które są często używane w biznesie transportowym) bez konieczności joinów
- Separacja `vehicle_type` i `fuel_type` do osobnych tabel:
 - Zamiast przechowywać te informacje jako teksty w tabeli `vehicle`
 - Uzasadnienie: zapewnia spójność danych
- Użycie `timestamp` zamiast `datetime`:
 - Uzasadnienie: Uniknięcie problemów ze strefami czasowymi (`timestamp` przechowuje UTC)
- Struktura kart paliwowych:
 - Karta jest powiązana zarówno z kierowcą jak i vendorem
 - Constraint `one_driver_one_vendor_one_card` zapewnia że kierowca może mieć tylko jedną kartę u danego vendora
 - Uzasadnienie: Odzwierciedla rzeczywisty model biznesowy gdzie kierowcy mogą mieć różne karty od różnych dostawców

Generowanie danych

Do wygenerowania danych wykorzystałem własne skrypty SQL i Python. Aby zwiększyć realizm danych wielokrotnie wykorzystywałem losowanie z rozkładu normalnego (na przykład do generowania cen paliwa). Na szczególną uwagę zasługuje program, który generuje pomiary poziomu paliwa podczas przejazdów samochodów oraz zapisy transakcji zakupu paliwa dokonanych przez kierowców gdy poziom był niski.



Widoczny na wykresie szum został wprowadzony do danych celowo w celu symulacji niedokładności przyrządów pomiarowych oraz zmiennego spalania pojazdu podczas podróży.

Użytkownicy

Stworzyłem użytkowników symbolizujących działy w firmie.

```
CREATE USER financial WITH PASSWORD 'financial';
CREATE USER compliance WITH PASSWORD 'compliance';
```

Następnie nadałem im dostęp do widoków, które zostały stworzone dla ich działów.

```
GRANT SELECT ON <widok> TO <uzytkownik>
REVOKE ALL ON SCHEMA public FROM <uzytkownik>
```

Testy dostępów

```
> PGPASSWORD=financial psql -h raspberrypi -p 30956 -U financial -d eltrans
psql (14.13 (Homebrew), server 17.0)
WARNING: psql major version 14, server major version 17.
        Some psql features might not work.
Type "help" for help.

eltrans=> select * from fuel_cost_per_driver_hour limit 10;
 driver_id | first_name | last_name | fuel_spending_per_hour
-----+-----+-----+-----
          10 | Ewa       | Nowak     | 89.69
          14 | Tomasz    | Woźniak   | 94.65
           8 | Marek     | Michalski | 110.19
           4 | Paweł     | Kowalski  | 147.46
           5 | Joanna    | Kowalczyk | 151.26
          13 | Marek     | Kozłowski | 153.24
           1 | Elżbieta  | Wiśniewska | 155.02
           9 | Józef     | Nowakowski | 163.58
           3 | Joanna    | Kowalska  | 168.10
           6 | Elżbieta  | Adamczyk  | 168.58
(10 rows)

eltrans=> select * from driver;
ERROR: permission denied for table driver
eltrans=>
```

```
> PGPASSWORD=compliance psql -h raspberrypi -p 30956 -U compliance -d eltrans
psql (14.13 (Homebrew), server 17.0)
WARNING: psql major version 14, server major version 17.
        Some psql features might not work.
Type "help" for help.






eltrans=> select * from suspected_fraudsters;
 first_name | last_name | vehicle_reg | timestamp | fuel_stolen
-----+-----+-----+-----+-----
Paweł      | Kowalski  | WAGVE03     | 2024-01-01 16:36:00 | 70.81
Aleksandra | Kowalczyk | WALWR09     | 2024-01-01 16:11:30 | 274.45
Aleksandra | Kowalczyk | WALWR09     | 2024-01-04 15:28:45 | 66.64
Tomasz     | Woźniak   | WAMDC09     | 2024-01-04 16:52:09 | 119.28
Elżbieta   | Wiśniewska | WAMOA01     | 2024-01-01 16:53:54 | 60.21
Adam        | Wiśniewski | WANUY09     | 2024-01-01 16:24:18 | 192.40
Adam        | Wiśniewski | WANUY09     | 2024-01-07 16:47:09 | 220.61
Marek      | Kozłowski | WAOTM08     | 2024-01-04 15:39:51 | 134.67
Jan         | Mazur     | WATUW05     | 2024-01-07 16:05:21 | 179.74
Elżbieta   | Adamczyk  | WAXIU05     | 2024-01-04 15:04:09 | 52.69
Katarzyna  | Pawłowska | WAXJH03     | 2024-01-01 15:53:21 | 79.20
Katarzyna  | Pawłowska | WAXJH03     | 2024-01-04 16:28:15 | 94.00
(12 rows)

eltrans=> select * from driver;
ERROR: permission denied for table driver
eltrans=>
```

Przykładowe zapytania

Którzy kierowcy wykonali najwięcej tras w danym miesiącu?

```
SELECT driver.first_name,  
       driver.last_name,  
       count(*) AS trips_taken  
FROM driver  
JOIN trip ON driver.id=trip.driver_id  
WHERE trip.start_time>='2024-01-01'  
      AND trip.end_time<'2024-02-01'  
GROUP BY driver.first_name,  
         driver.last_name  
ORDER BY trips_taken DESC  
LIMIT 10;
```

#	 A-Z first_name 	A-Z last_name 	123 trips_taken 
1	Zbigniew 	Nowak	3
2	Adam	Kowalczyk	3
3	Łukasz	Szymański	3
4	Joanna	Wiśniewska	3
5	Marcin	Kwiatkowski	3
6	Marcin	Woźniak	2
7	Adam	Kowalski	2
8	Andrzej	Lewandowski	2
9	Grzegorz	Wójcik	2
10	Katarzyna	Zielińska	2

Jaka jest średnia cena oleju napędowego w Polsce?

W tym zapytaniu wykorzystane zostało **podzapytanie**.

```
SELECT ft.name AS "Nazwa paliwa",  
       ROUND(CAST(sum(cost) / sum(fuel_amount) AS numeric), 2) AS "Cena paliwa [EUR]"  
FROM sale s  
JOIN fuel_type ft ON ft.id = s.fuel_type_id  
WHERE vendor_id IN  
      (SELECT id  
       FROM vendor  
       WHERE country = 'Poland')
```



```
AND ft.name = 'diesel'
GROUP BY ft.name;
```

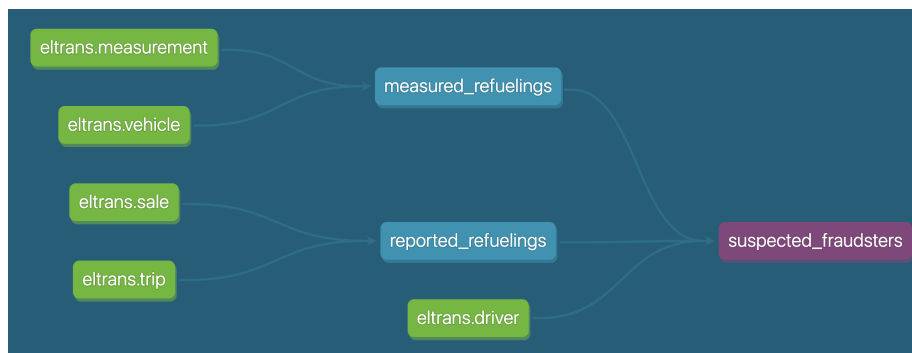
#	A-Z Nazwa paliwa ↓↑	123 Cena paliwa [EUR] ↓↑
1	diesel	1.5

Perspektywy

Utworzyłem sześć perspektyw.

Wykrywanie fraudów

Kierowcy kupują paliwo na stacjach paliw przy użyciu firmowych kart flotowych. Kierowca może próbować zatankować część paliwa do własnego kanistra aby w ten sposób “dorobić” sobie bonus do pensji. Jest to oczywiście kradzież. W celu jej wykrywania stworzone zostały perspektywy.



```
SELECT * FROM reported_refuelings;
```

#	123 fuel_added ↓↑	timestamp ↓↑	A-Z vehicle_reg ↓↑	123 driver_id ↓↑
1	1249.16	2024-01-01 16:53:55.000	WAMOA01	1
2	754.14	2024-01-04 14:56:43.000	WAMOA01	1
3	939.3	2024-01-07 15:21:34.000	WAMOA01	1
4	1381.07	2024-01-01 16:24:19.000	WANUY09	2
5	921.61	2024-01-04 15:23:46.000	WANUY09	2
6	1409.22	2024-01-07 16:47:10.000	WANUY09	2
7	1175.58	2024-01-01 16:42:01.000	WAYHI08	3
8	1045.44	2024-01-04 15:22:16.000	WAYHI08	3
9	1061.8	2024-01-01 16:36:01.000	WAGVE03	4
10	841.77	2024-01-04 14:59:01.000	WAGVE03	4
11	975.54	2024-01-01 15:26:46.000	WALNO01	5
12	862.92	2024-01-04 15:13:19.000	WALNO01	5

```
SELECT * FROM measured_refuelings;
```

#	A-Z vehicle_reg ↓↑	123 fuel_added ↓↑	timestamp ↓↑
1	WAMOA01	1188.9534073964683	2024-01-01 16:53:54.000
2	WAMOA01	754.0765134427759	2024-01-04 14:56:42.000
3	WAMOA01	938.4067551316404	2024-01-07 15:21:33.000
4	WANUY09	1188.66922327344	2024-01-01 16:24:18.000
5	WANUY09	922.1991233817599	2024-01-04 15:23:45.000
6	WANUY09	1188.6094756769883	2024-01-07 16:47:09.000
7	WAYHI08	1132.7650542949686	2024-01-01 16:42:00.000
8	WAYHI08	1003.6869848406427	2024-01-04 15:22:15.000
9	WAGVE03	990.9931671729654	2024-01-01 16:36:00.000
10	WAGVE03	841.5433555682264	2024-01-04 14:59:00.000
11	WALNO01	975.6183475425687	2024-01-01 15:26:45.000
12	WALNO01	863.3450326906885	2024-01-04 15:13:18.000
13	WALNO01	1007.9772090774426	2024-01-07 15:12:54.000

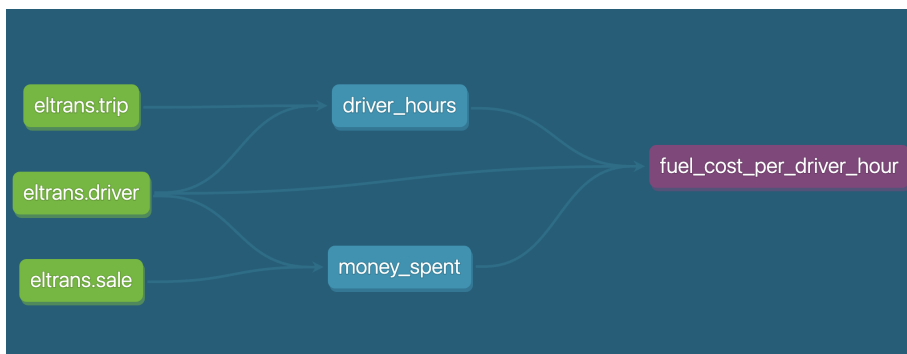
```
SELECT * FROM suspected_fraudsters;
```

Enter a SQL expression to filter results, e.g. column_name = 'v'

#	A-Z first_name ↓↑	A-Z last_name ↓↑	A-Z vehicle_reg ↓↑	timestamp ↓↑	123 fuel_stolen ↓↑
1	Paweł	Kowalski	WAGVE03	2024-01-01 16:36:00.000	70.81
2	Aleksandra	Kowalczyk	WALWR09	2024-01-01 16:11:30.000	274.45
3	Aleksandra	Kowalczyk	WALWR09	2024-01-04 15:28:45.000	66.64
4	Tomasz	Woźniak	WAMDC09	2024-01-04 16:52:09.000	119.28
5	Elżbieta	Wiśniewska	WAMOA01	2024-01-01 16:53:54.000	60.21
6	Adam	Wiśniewski	WANUY09	2024-01-01 16:24:18.000	192.4
7	Adam	Wiśniewski	WANUY09	2024-01-07 16:47:09.000	220.61
8	Marek	Kozłowski	WAOTM08	2024-01-04 15:39:51.000	134.67
9	Jan	Mazur	WATUW05	2024-01-07 16:05:21.000	179.74
10	Elżbieta	Adamczyk	WAXIU05	2024-01-04 15:04:09.000	52.69
11	Katarzyna	Pawłowska	WAXJH03	2024-01-01 15:53:21.000	79.2
12	Katarzyna	Pawłowska	WAXJH03	2024-01-04 16:28:15.000	94

Koszt paliwa na godzinę jazdy kierowcy

Firma chce nagradzać kierowców którzy wybierają tanie stacje i nie mają ciężkiej nogi. W tym celu skorzystać można z perspektyw, które informują o średnim koszcie paliwa spalonego przez danego kierowcę na godzinę jazdy.



```
SELECT * FROM driver_hours;
```

#	A-Z first_name ↓↑	A-Z last_name ↓↑	123 hours_driven ↓↑	123 driver_id ↓↑
1	Elżbieta	Wiśniewska	26.75263501749999999	1
2	Adam	Wiśniewski	26.24400638444...	2
3	Joanna	Kowalska	18.769775878888889	3
4	Paweł	Kowalski	18.493900615277778	4
5	Joanna	Kowalczyk	26.494782386944444	5
6	Elżbieta	Adamczyk	17.443545604722222	6

```
SELECT * FROM money_spent;
```

#	A-Z first_name ↓↑	A-Z last_name ↓↑	123 driver_id ↓↑	123 total_spending ↓↑
1	Elżbieta	Wiśniewska	1	4147.2
2	Adam	Wiśniewski	2	5603.63
3	Joanna	Kowalska	3	3155.21
4	Paweł	Kowalski	4	2727.02
5	Joanna	Kowalczyk	5	4007.63
6	Elżbieta	Adamczyk	6	2940.67
7	Katarzyna	Pawłowska	7	3862.43

```
SELECT * FROM fuel_cost_per_driver_hour;
```

#	123 driver_id ↓↑	A-Z first_name ↓↑	A-Z last_name ↓↑	123 fuel_spending_per_hour ↓↑
1	10	Ewa	Nowak	89.69
2	14	Tomasz	Woźniak	94.65
3	8	Marek	Michalski	110.19
4	4	Paweł	Kowalski	147.46
5	5	Joanna	Kowalczyk	151.26
6	13	Marek	Kozłowski	153.24
7	1	Elżbieta	Wiśniewska	155.02
8	9	Józef	Nowakowski	163.58
9	2	Adam	Wiśniewski	166.37

Indeksy

W bazie stworzyłem indeksy aby przyspieszyć niektóre zapytania, które dają istotne informacje z punktu widzenia biznesowego. Tam, gdzie zapytania mają formę porównania do konkretnej wartości (na przykład numer rejestracyjny) wykorzystany został indeks typu hash. W przeciwnym wypadku (oraz w indeksie złożonym z kilku kolumn ponieważ takich nie obsługuje hash) wykorzystałem indeks btree.

trip

Utworzone zostały indeksy typu hash na kolumnach **source** i **destination** aby przyspieszyć zapytania o miejsca, gdzie jeżdżą pojazdy.

vendor

Utworzony został indeks typu hash na kolumnie **country** aby przyspieszyć zapytania o kraje w których zarejestrowane są działalności dostawców paliwa.

sale

Utworzony został indeks btree na kolumnie `cost` zawierający wartości `cost`, `fuel_amount` i `vendor_id` aby przyspieszyć zapytania o średnie ceny paliwa u dostawców.

driver

Utworzony został indeks złożony typu btree na kolumnach `first_name` i `last_name` aby przyspieszyć zapytania o kierowców przy użyciu ich imion (w przeciwieństwie do ich id).

vehicle i trailer

W obu tabelach utworzono indeksy typu hash na kolumnach z numerami rejestracyjnymi.

Benchmark

Bez indeksu na kolumnie `timestamp` w `measurement` wykonanie zapytania `SELECT min(m.timestamp) FROM measurement m` zajmowało około 900 milisekund. Po dodaniu indeksu btree na tę kolumnę czas wykonania tego zapytania spadł do 12 milisekund! To redukcja o niemal **99%**!