

POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRYCZNY

KIERUNEK INFORMATYKA STOSOWANA

Etap 2

Wykonał:
Aliaksandr KAROLIK

Sprawdzający:
dr inż. Krzysztof HRYNIÓW

31 października 2021



Spis treści

1	Indeksy	2
2	Widoki	2
3	Wyzwalacze	2
4	Funkcje	4
5	Procedury	4
6	Automatyzacja cyklicznego zadania	6

1 Indeksy

Index zostały zastosowane dla tabel o nazwach **flight** oraz **passenger**. Tabela **flight** została wybrana ze względu na to, że będzie ona zawierać dane historyczne oraz aktualne dane związane z lotami. Ze względu na przechowywanie historycznych danych tabela będzie zazwyczaj wykorzystywana podczas przeprowadzania analizy popularności lotów. Kolumny, które zostały pokryte indeksami mają nazwy **departure**, **arrival** oraz **flight_number**. Kolumna **flight_number** powstała podczas poprawy pierwszego etapu projektu. Kolumny te będą zazwyczaj używane podczas sortowania i filtrowania danych, co w wyniku ma przyspieszyć wykonywania zapytań.

Jestem świadomy problemu związanego z dodawaniem nowych rekordów do tabeli, bo będzie potrzeba również aktualizacja indeksów. Jako ewentualne rozwiązanie można zrobić dodatkowy z automatyzowane zadanie, które będzie codziennie tworzyć widok zawierający wszystkie potrzebne informacje do analizy i operować na nim. W wyniku można będzie pozbyć się indeksów w tej tabeli i nie tracić w wydajności.

Dla tabeli **passenger** został wykorzystany indeks kompozytowy:

```
CREATE unique INDEX Ix_passport_surname_name ON
passenger(passport, surname, name);
```

Taki wybór był spowodowany tym, że na lotnisku jako pasażer musimy okazać paszport, który jest sprawdzany w systemie, w wyniku taka operacja ma odbywać bardzo szybko.

2 Widoki

Dla użytkownika zostały udostępnione widoki o nazwach **allFlightsWithTime** oraz **passengersWithDestinations**. Pierwszy widok udostępnia tak naprawdę najbardziej podstawowe informacje, czyli skąd do kąd jest lot oraz jest jego cenę. Taki widok może być użyty dla przekazywania informacji do frontendu na przykład. Rysunek 1 pokazuje co zawiera widok.

Drugi widok udostępnia informacje związane z pasażerami a dokładnie informacje o imieniu, nazwisku, paszporcie oraz kierunek lotów tego pasażera. Na rysunku 2 jest przedstawiona zawartość widoku.

3 Wyzwalacze

Do bazy dodano wyzwalacz dla tabeli **passenger** który jest uruchomiany przed wstawianiem danych. Celem wyzwalacza jest walidacja danych wejściowych pod względem unikalności paszportu. Czyli w przypadku próby dodawania rekordu zawierającego paszport już istniejący w bazie zostanie zwrócony kod błędu oraz komunikat o błędzie dla użytkownika i transakcja nie zostanie zakończona pomyślnie. Na rysunku 3 jest zaprezentowany wynik działania wyzwalacza

```
132
133 • CREATE VIEW allFlightsWithTime AS
134 select depart.city as depart, arrival.city as arrivals, tariff.cost as price, flight.departure as timeofdeparture,
135 flight.arrival as timeofarrival, aircompany.aircompany_name as transporter
136 from flight
137 join airRoute on flight.route_id = airRoute.airRoute_id
138 join tariff on tariff.airRoute_id = airRoute.airRoute_id
139 join aircompany on aircompany.aircompany_id = airRoute.aircompany
140 join airport as arrival on airRoute.arrivals_airport = arrival.airport_id
141 join airport as depart on airRoute.departures_airport = depart.airport_id
142 order by (depart.city);
143
144 • SELECT * FROM allFlightsWithTime;
```

Result Grid

#	depart	arrivals	price	timeofdeparture	timeofarrival	transporter
1	Barcelona	London	100	2021-08-23 06:45:56	2021-08-23 12:45:56	Ryanair
2	Dubai	Paris	80	2021-07-23 07:45:56	2021-07-23 11:45:56	Flydubai
3	Dubai	Barcelona	20	2021-01-23 12:45:56	2021-01-23 15:45:56	Flydubai
4	London	Barcelona	300	2021-05-23 12:45:56	2021-05-23 18:45:56	Wizzair
5	Warsaw	Paris	700	2021-02-23 13:45:56	2021-02-23 17:45:56	Ryanair

Rysunek 1: Widok allFlightsWithTime

```
146 • CREATE VIEW passengersWithDestinations AS
147 select passenger.name, passenger.surname, passenger.passport, depart.city as depart, arrival.city as arrivals from passenger
148 join ticket on ticket.passenger = passenger_id
149 join tariff on tariff.tariff_id = ticket.tariff
150 join airRoute on airRoute.airRoute_id = tariff.airRoute_id
151 join airport as arrival on airRoute.arrivals_airport = arrival.airport_id
152 join airport as depart on airRoute.departures_airport = depart.airport_id
153 order by passenger.surname;
154
155 • select * from passengersWithDestinations;
```

Result Grid

#	name	surname	passport	depart	arrivals
1	Wadsworth	Conring	1YVHZ8BA9A	Dubai	Barcelona
2	Wadsworth	Conring	1YVHZ8BA9A	Warsaw	Paris
3	Miles	Grafhom	WAUKF78P49	Barcelona	London
4	Lewie	Hearnes	2G4WD58236	Dubai	Barcelona
5	Lanae	Mousedall	JH4CU2E60A	Dubai	Barcelona
6	Lanae	Mousedall	JH4CU2E60A	London	Barcelona
7	Willamina	Noods	WBASP2C55C	Dubai	Paris

Rysunek 2: Widok passengersWithDestinations

```
273
274 • insert into passenger(name,surname, passport) values('Lewie','Hearnes','2G4WD58236');
```

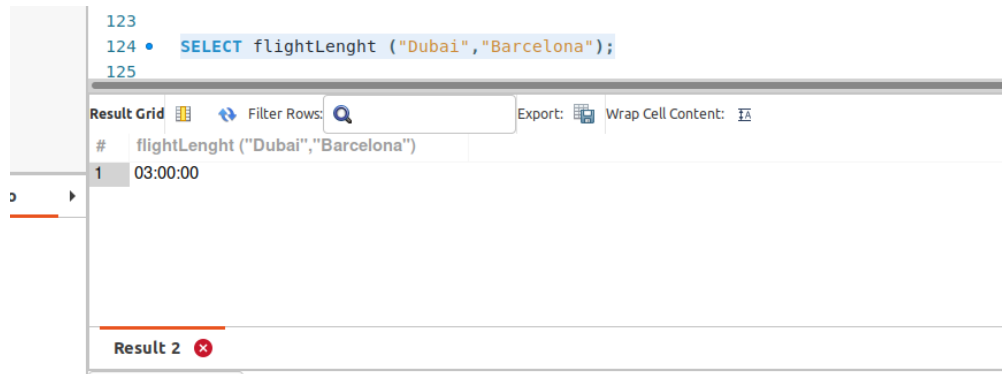
Action Output

#	Time	Action	Message	Duration / Fetch
1	19:45:42	use airlines	0 row(s) affected	0,00075 sec
2	19:45:46	Insert into passenger(name,surname, passport) values(...	Error Code: 1644. Passport must be unique. This passenger already in database	0,018 sec

Rysunek 3: Wynik działania wyzwalacza

4 Funkcje

Do bazy została dodana funkcja o nazwie **flightLenght** która przyjmuje jako argumenty nazwy miast do których odbywa się lot i wylicza długość lotu. Do wyliczania długości lotu została wykorzystana funkcja **TIMEDIFF**. Wynik działania jest przedstawiony na rysunku 4



The screenshot shows a SQL query editor with the following code:

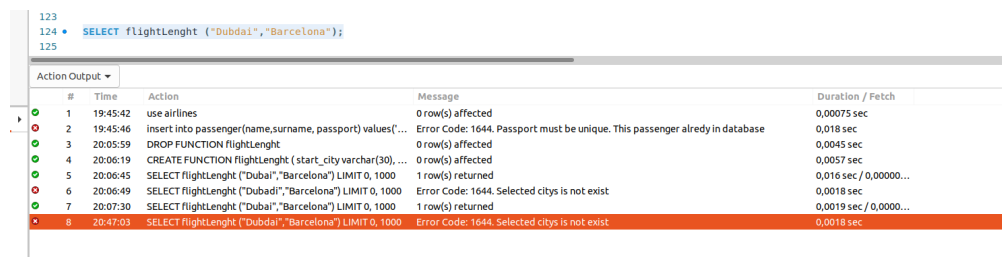
```
123
124 • SELECT flightLenght ("Dubai", "Barcelona");
125
```

Below the code, the 'Result Grid' is displayed with the following data:

#	flightLenght ("Dubai", "Barcelona")
1	03:00:00

At the bottom, there is a 'Result 2' status bar with a red 'x' icon, indicating an error.

(a) Dane poprawne



The screenshot shows the 'Action Output' window with the following data:

#	Time	Action	Message	Duration / Fetch
1	19:45:42	use airlines	0 row(s) affected	0,00075 sec
2	19:45:46	insert into passenger(name,surname,passport) values('...'	Error Code: 1644. Passport must be unique. This passenger already in database	0,018 sec
3	20:05:59	DROP FUNCTION flightLenght	0 row(s) affected	0,0045 sec
4	20:06:19	CREATE FUNCTION flightLenght (start_city varchar(30), ...	0 row(s) affected	0,0057 sec
5	20:06:45	SELECT flightLenght ("Dubai", "Barcelona") LIMIT 0, 1000	1 row(s) returned	0,016 sec / 0,00000...
6	20:06:49	SELECT flightLenght ("Dubai", "Barcelona") LIMIT 0, 1000	Error Code: 1644. Selected city is not exist	0,0018 sec
7	20:07:30	SELECT flightLenght ("Dubai", "Barcelona") LIMIT 0, 1000	1 row(s) returned	0,0019 sec / 0,00000...
8	20:47:03	SELECT flightLenght ("Dubai", "Barcelona") LIMIT 0, 1000	Error Code: 1644. Selected city is not exist	0,0018 sec

(b) Dane nie poprawne

Rysunek 4: Wynik działania funkcji flightLenght

5 Procedury

Dla użytkowników bazy zostały przygotowane procedury o nazwach **getraiseBy-NameAndSurname** **createNewRoute**. Pierwsza procedura pozwala sprawdzić dla pasażera jego trasę taka procedura będzie dosyć często wykorzystywana na lotnisku przez personal. Na rysunku 5 został zaprezentowany wynik działania procedury.

Druga procedura umożliwia stworzenia nowej trasy, jako argumenty przyjmuje ona nazwy lotnisk. Jeżeli baza nie zawiera podanych lotnisk, zostają one dodane do tabeli **airport**. Aby kontrolować transakcje został stworzony handler wewnątrz procedury, który będzie przywracać transakcję. Handler zdefiniowany został przy pomocy następującej komendy:

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
```

procedures	89	
NewRoute	90	CALL getraiseByNameAndSurname("Lewie", "Hearnes");
seByNameAr	91	
s	92	
	93	

#	name	surname	depart	arrivals	departure_time	arrival_time
1	Lewie	Hearnes	Dubai International Airport	Barcelona-El Prat Airport	2021-01-23 12:45:56	2021-01-23 15:45:56

Rysunek 5: Wynik działania procedury getraiseByNameAndSurname

Na rysunku 6 są przedstawione wyniki działania procedury przy podaniu testowych nazw lotnisk. Jak widać zostały one dodane do tabeli **airport** oraz powstał wpis w tabeli **airRoute**. Na rysunku 7 widać wynik działania procedury przy podaniu błędnego argumentu w wyniku działania procedury zwrócony jest kod błędu dla użytkownika.

#	airport_id	city	airport_name
1	1	Dubai	Dubai International Airport
2	2	Barcelona	Barcelona-El Prat Airport
3	3	Warsaw	Warsaw Chopin Airport
4	4	London	London Heathrow Airport
5	5	Paris	Charles de Gaulle Airport
6	6		Test1
7	7		Test2

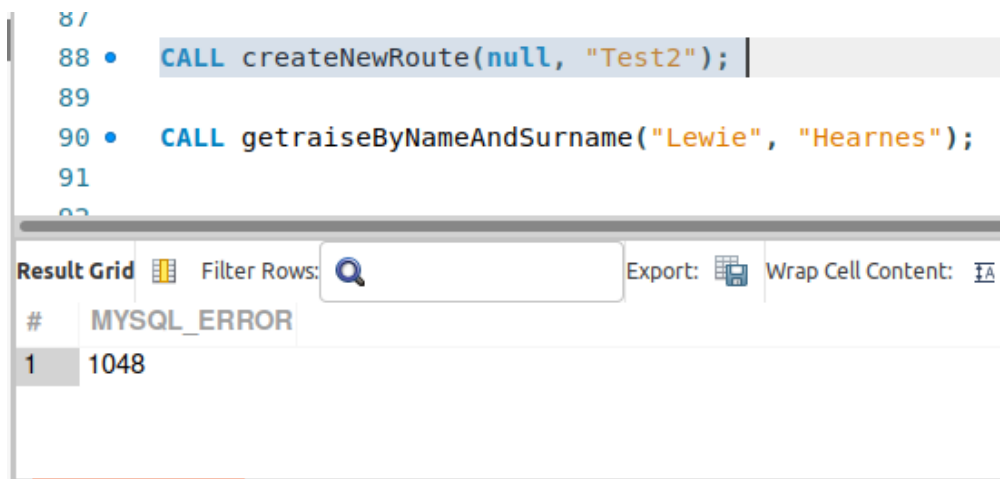
#	airRoute_id	plane_id	departures_airport	arrivals_airport	aircompany
1	1	1	2	2	2
2	2	4	3	5	1
3	3	2	4	2	3
4	4	5	1	5	2
5	5	1	2	4	1
6	6		6	7	

#	Time	Action	Message
8	13:38:00	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
9	13:38:01	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
10	13:38:02	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
11	13:38:03	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
12	13:38:03	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
13	13:38:04	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
14	13:38:04	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
15	13:38:10	CALL createNewRoute("Test1", "Test2")	0 row(s) affected
16	13:38:16	SELECT * FROM airlines.airport LIMIT 0, 1000	7 row(s) returned

#	Time	Action	Message
8	13:38:00	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
9	13:38:01	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
10	13:38:02	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
11	13:38:03	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
12	13:38:03	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
13	13:38:04	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
14	13:38:04	Get schemata	Error Code: 2013 Lost connection to MySQL server at 'reading in'
15	13:38:10	CALL createNewRoute("Test1", "Test2")	0 row(s) affected
16	13:38:18	SELECT * FROM airlines.airport LIMIT 0, 1000	7 row(s) returned
17	13:38:43	SELECT * FROM airlines.airRoute LIMIT 0, 1000	6 row(s) returned

(a) Tabela airport po wykonaniu procedury (b) Tabela airRoute po wykonaniu procedury

Rysunek 6: Wynik działania createNewRoute. Dane poprawne



Rysunek 7: Wynik działania createNewRoute. Dane nie poprawne

6 Automatyzacja cyklicznego zadania

W projekcie zostało stworzone automatyczne zadanie celem, którego jest codzienne zbieranie statystyki ilości osób podróżujących dla każdego lotu w ciągu dnia. Zebrane statystyki następnie są eksportowane do plików csv, które umieszczane są w folderze `/usr/local/reports/` który później może być użyty jako volume, aby nie tracić je po usunięciu kontenera.

Aby móc tworzyć cykliczne zadania w MySQL, musiałem ustawić odpowiednie zmienne w bazie oraz kontenerze. W bazie zostały ustawione następujące zmienne:

```
SET GLOBAL event_scheduler = ON;
SET @@GLOBAL.event_scheduler = ON;
SET GLOBAL event_scheduler = 1;
SET @@GLOBAL.event_scheduler = 1;
```

Następnie w docker compose dodano zmienną środowiskową, która odpowiada za wskazania folderu, do którego MySQL ma uprawnienia do zapisywania plików:

```
--secure-file-priv='/usr/local/reports/'
```

Poniżej jest przedstawiony kod samego joba.

```
SET @TS = DATE_FORMAT(NOW(), '%Y_%m_%d_%H_%i_%s');
SET @FOLDER = '/usr/local/reports/';
SET @PREFIX = 'report';
SET @EXT = '.csv';
```

```
set @QUERY = "select CURRENT_DATE as fly_date, flight.flight_number as flight_number,
count(*) as number_pass from passenger
join ticket on ticket.passenger = passenger_id
join flight on ticket.flight = flight.flight_id
WHERE departure > DATE_SUB(NOW(), INTERVAL 24 HOUR) AND departure <= NOW()
group by flight.flight_number INTO OUTFILE '";

SET @CMD = CONCAT(@QUERY,@FOLDER,@PREFIX,@TS,@EXT,
"' FIELDS ENCLOSED BY '\"',
" TERMINATED BY ','",
" ESCAPED BY '\"',
" LINES TERMINATED BY '\\n';");

PREPARE statement FROM @CMD;
EXECUTE statement;
```

Zapytanie wykonywane wewnątrz zadania jest konstruowane przy pomocy metody concat ze względu na chęć tworzenia unikalnych plików. Tworzenie unikalnych plików było dosyć przydatne podczas testowania poprawności implementacji.

Dla przetestowania poprawności działania cykliczność została zmieniona na jedną minutę. Dla przetestowania zostały również dodane dane testowe zawierające informacje o lotach w dzień testu. Na rysunku 8 są zaprezentowane wyniki testów.

```
• use airlines;
• insert into flight( route_id, plane, pilot, departure, arrival, flight_number) values(1, 2, 1, '2021-10-29 12:45:56', '2021-10-29 15:45:56', '1G6D357V58');
• insert into flight( route_id, plane, pilot, departure, arrival, flight_number) values(2, 3, 3, '2021-10-29 12:43:56', '2021-10-29 12:45:56', '1G6D357V53');
• insert into ticket(passenger, tariff, flight) values(1,1,6);
• insert into ticket(passenger, tariff, flight) values(2,1,6);
• insert into ticket(passenger, tariff, flight) values(4,1,7);
```

(a) Dodawanie testowych danych do bazy

```
root@51d3fa6a8230:/usr/local/reports# cat report_2021_10_29_14_55_00.csv
root@51d3fa6a8230:/usr/local/reports# ls
report_2021_10_29_14_55_00.csv
root@51d3fa6a8230:/usr/local/reports# ls
report_2021_10_29_14_55_00.csv
root@51d3fa6a8230:/usr/local/reports# ls
report_2021_10_29_14_55_00.csv
root@51d3fa6a8230:/usr/local/reports# ls
report_2021_10_29_14_55_00.csv report_2021_10_29_14_56_00.csv
root@51d3fa6a8230:/usr/local/reports# cat report_2021_10_29_14_56_00.csv
"2021-10-29","1G6D357V53","1"
"2021-10-29","1G6D357V58","2"
root@51d3fa6a8230:/usr/local/reports# ^C
root@51d3fa6a8230:/usr/local/reports# exit
```

(b) Wynik działania zadania

Rysunek 8: Wynik działania zadania