Patryk Marszałek, Paweł Nowak, Maciej Rut.

Naszym celem było stworzenie bazy danych biura podróży, zawierającej dane o klientach, wycieczkach etc. W bazie znajdują się następujące tabele:

1. Clients – tabela z informacjami o klientach.
2. Statutes – tabela przechowująca statusy upoważniające do zniżek.
3. Clinets\_statuses – tabela z informacjami, który klient ma jakie statusy.
4. Countries – kraje, przez które mogą przebiegać wycieczki.
5. Cities – miasta, powiązane z krajami.
6. Attractions – atrakcje, powiązane z krajami.
7. Trips – lista wycieczek oferowanych przez biuro.
8. Trip\_dates – lista dat, kiedy odbywają się konkretne wycieczki (jedna wycieczka może odbywać się kilka razy).
9. Client\_trips – tabela opisująca jakie wycieczki zarezerwowali klienci (z tabeli trip\_dates) i ile już zapłacili.
10. Pilots – prowadzący wycieczki.
11. Pilot\_warrants – tabela z informacjami, do prowadzenia których wycieczek upoważniony jest pilot.
12. Travels – tabela opisująca środki transportu pomiędzy miastami i ich ceny.
13. Travels\_price\_changes – tabela przechowująca informacje o czasowych zmianach cen podróży (z travels).
14. Trip\_routes – tabela opisująca przebieg wycieczki (z jaki środków transportu korzysta).
15. Discounts – tabela opisująca zniżki (niezależnie czy dotyczą środków transportu czy atrakcji), każda zniżka jest przypisana do statusu (czyli komu się należy).
16. Attraction\_discounts – tabela opisująca jakie zniżki są dostępne dla danej atrakcji.
17. Travels\_discounts – analogicznie co wyżej, dla środków transportu.
18. Opinions – tabela z opiniami wystawionymi przez klientów, anonimowo bądź nie.
19. Trip\_attractions – tabela przechowująca listę atrakcji odwiedzanych podczas wycieczki, z informacją czy są one obowiązkowe czy opcjonalne.

W bazie są stworzone enumy attractions\_type i travel\_type służące do opisu typu atrakcji lub rodzaju środka podróży (używane odpowiednio w attractions i travels).

Funkcje:

- real\_price() – funkcja licząca cenę konkretnej wycieczki (z datą) dla konkretnego klienta uwzględniając jego zniżki i okresowe zmiany cen.

- max\_price() – funkcja szacująca cenę wycieczki (bez zniżek i zmian okresowych).

- sum\_discounts() – funkcja, która dla klienta i wycieczki oblicza, ile oszczędza dzięki swoim statusom.

- cash\_back() – funkcja, obliczająca jaką część wpłaconych pieniędzy odzyska klient w wypadku rezygnacji z udziału w wycieczce (im później rezygnuje, tym mniej odzyskuje).

Powyższe funkcje te są używane w aplikacji.

- travels\_both\_directions() – funkcja ułatwiające wypełnianie tabeli travels - dodaje podróż oraz symetryczną podróż w drugą stronę.

-City\_id(), Country\_id() – funkcje zwracające id odpowiednio miasta lub kraju po otrzymaniu jego nazwy (ułatwiały wprowadzanie danych).

Widoki:

-payments – widok z listą wszystkich klientów, zakupionych przez nich wycieczek, ich cen oraz informację o płatnościach dokonanych przez klienta.

-pilot\_trips – lista pilotów i ich wycieczek (które prowadzą).

-trip\_atractions\_list – widok z listą wycieczek i ich atrakcji.

-client\_discounts – widok z listą klientów, ich wycieczek i ile oszczędzają na zniżkach, dla danej wycieczki.

-client\_cash\_back – lista klientów, ich wycieczek i ile otrzymają zwrotu, jeśli zrezygnują.

-client\_opinions – lista klientów (z danymi) i wystawionych przez nich opinii.

Widoki są użyteczne w aplikacji.

Założone triggery:

-przy usuwaniu kraju usuwane są również jego miasta.

-przy usuwaniu miasta usuwane są jego atrakcje.

-przy usuwaniu atrakcji są one usuwane z też tabel attraction\_discounts i trip\_attractions.

-usunięcie klienta usuwa też jego rezerwacje wycieczek.

-usuniecie zniżki z tabeli discounts usuwa ją również z attraction\_discounts i travel\_discounts.

-usunięcie wycieczki z trips usuwa wszystkie jej terminy z trip\_dates.

-przy podawaniu PESELU sprawdzana jest cyfra kontrolna (można zastanawiać się nad sensownością peselu w potencjalnie międzynarodowej bazie danych, ale przyjmijmy pewne uproszczenie rzeczywistości).

-przy insertowaniu (lub updatowaniu) wycieczki do trip\_dates jeśli podamy pilota to zostanie sprawdzone, czy nie ma on już wycieczki o kolidującym terminie, gdy nie podamy pilota, zostanie on znaleziony i dopasowany automatycznie (oczywiście w niekolidujący sposób oraz wycieczki są rozdzielane mniej więcej równomiernie między różnych pilotów).

-przy usunięciu pilota, który miał zaplanowane pewne wycieczki, baza podejmie próbę automatycznego przypisania nowych pilotów do tych wycieczek.

-podobnie przy usuwaniu pilotom uprawnień z pilot\_warrants.

-triggery sprawdzające spójność danych w tabeli trip\_routes (czy kolejne transporty rzeczywiście wyjeżdżają z miasta, w którym aktualnie jesteśmy i czy w zaplanowana trasa nie jest dłuższa od długości wycieczki podanej w trips).

-trigger pozwalający na usuwanie kolejnych transportów z trip\_routes tylko od końca (by nie rozspójnić danych).

-trigger sprawdzający, czy nie usuwamy transportu z travels używanego w jakiejś wycieczce. W wypadku gdy usunięcie się powiedzie, usuwamy również informacje z travel\_price\_changes i travel\_discounts.

-trigger sprawdzający, czy na jedną wycieczkę nie próbujemy zapisać więcej osób, niż wyspecyfikowane w tabeli trips (kolumna max\_people).

-trigger sprawdzający, czy insertując bądź updatując informacje w trip\_attractions nie próbujemy przypisać do wycieczki atrakcji z miasta, którego nigdy ona nie odwiedza.

-trigger zabraniający zmiany id klienta w tabeli opinions.

-Kilka triggerów blokujących updaty: istnieją krotki, których update mógłby poważnie zaburzyć spójność bazy (np. from\_city lub to\_city w travels). Takie updaty są zablokowane, w razie pomyłki należy najpierw użyć delete a później insert (te opatrzone są odpowiedznimi triggerami).

-trigger nie pozwalający dać czemuś zniżki większej niż tego cena.

Co do komentarzy po etapie pierwszym:

-zostały dodane informacje o kolejności przebiegu trasy wycieczki i w którym dniu odbywają się przejazdy,

-została dodana tabela trip\_dates (modelująca to, że zazwyczaj biura podróży oferują wycieczki w różnych terminach).

-została dodana tabela travel\_price\_changes (pojawił się zarzut, że ceny lotów w dłuższym okresie czasu nie kosztują tyle samo).

-zostały dodane uprawnienia pilotów (pilot\_warrants).

-zostały dodane opinie klientów.

Napotkane trudności:

Operacja update – baza mogłaby zostać przez nią rozspójniona w sposób omijający wszelkie constrainty założone na tabelach (np. klucze obce). Natomiast pisanie odpowiednich triggerów do wszystkich możliwych updatów byłoby bardzo skomplikowane. Dlatego na niektórych krotkach zostały założone triggery w ogóle blokujące pewne updaty. W takich wypadkach należy używać najpierw delete a później insert (te już można łatwiej zabezpieczyć).

Aplikacja:

Aplikacja w języku Java. Główna klasa Main. Wymaga JavaFX SDK

<https://gluonhq.com/products/javafx/>

Wymagania odpalania z opcją VM

--module-path [ścieżka do biblioteki JavaFX] --add-modules javafx.controls,javafx.fxml.

Po odpaleniu pojawi się okienko łączenia z PSQL (należy wpisać nazwę użytkownika i hasło).

Następnie można zalogować się lub założyć konto.

W aplikacji można wyświetlić listę wycieczek (klikając w przycisk trips), po kliknięciu w nazwę wycieczki otworzy się okno z jej terminami i pod spodem krótkim opisem.

Po zalogowaniu mamy możliwość zapisu na wycieczki w różnych terminach, wyświetlania zakupionych wycieczek, otrzymanych zniżek, statusów oraz napisania (pozytywnej) opinii.

Istnieje plik run.sh wystarczy w terminalu wpisać “bash run.sh” testowane na Ubuntu 20.04.