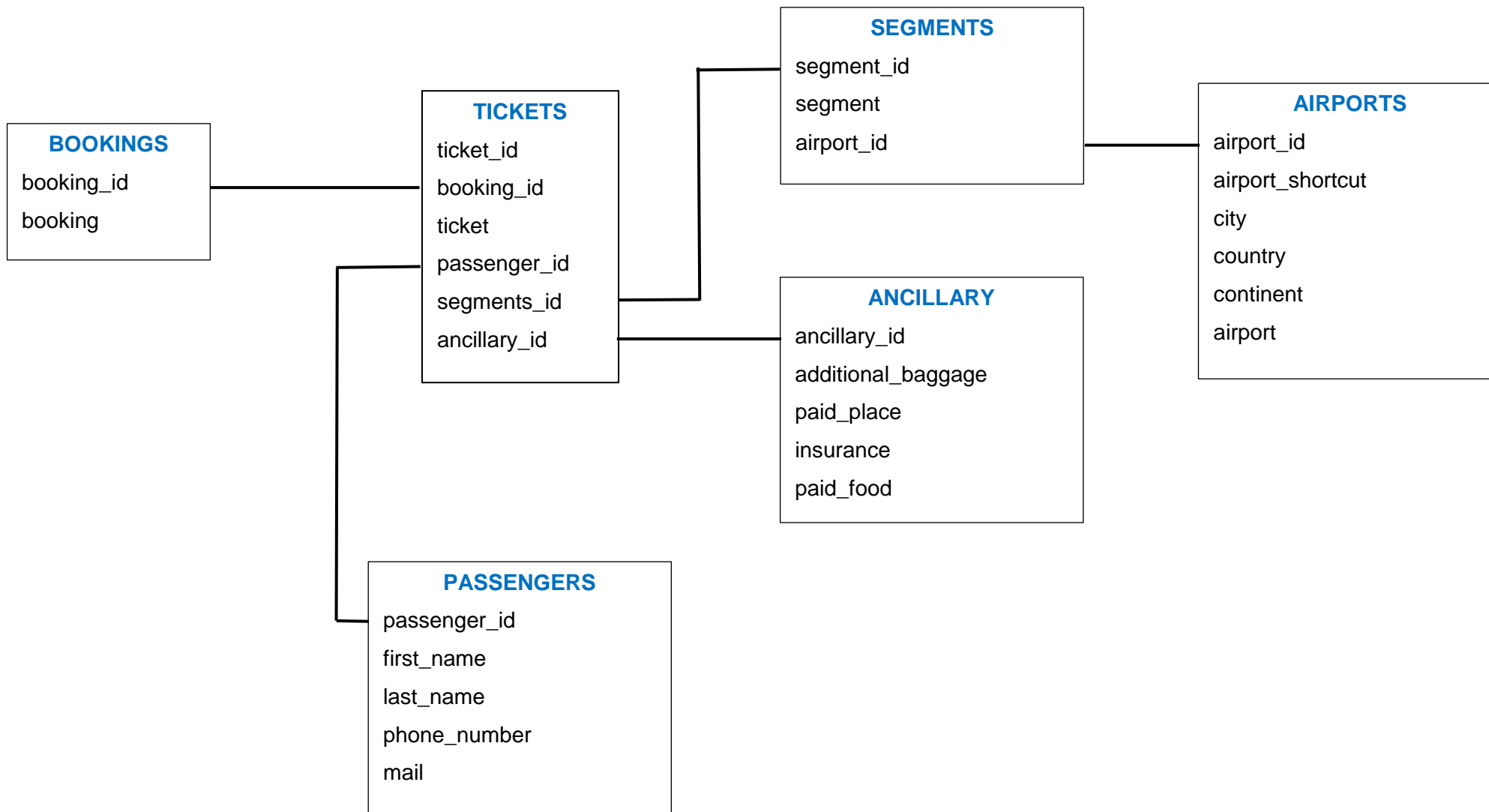


Zad. 1 Model danych



1. BOOKINGS

booking_id (INT) - prim-key, not-null, unique, auto-increment

booking (VARCHAR(10)) - not-null,

Pierwsza kolumna (booking_id) jest kluczem głównym, będącym w relacji z tabelą tickets. Każdy wiersz musi być unikalny, przyporządkowany do jednej rezerwacji. Dzięki autoinkrementacji, wartość kolejnego wiersza automatycznie zwiększa się o jeden, w porównaniu do wiersza poprzedniego.

Booking – tabela zawierająca nazwy rezerwacji, każda utworzona rezerwacja otrzymuje swój unikatowy numer(z kolumny booking_id).

Nazwa TOBER oznacza lot do Berlina(to Berlin). Liczba, oznacza numer zamówienia, na przykład danego dnia, czy tygodnia. Dla przykładu wartość TOLHR_843 oznacza 843 rezerwację na lot do Londynu(lotnisko London Heathrow Airport z oznaczeniem LHR).VARCHAR(10) służy do określenia ilości znaków możliwych do wpisania. W tym przypadku nie potrzeba „zapychać” pamięci większymi wymaganiami. Kolumna nie może być pusta, oraz musi być unikalna. Miejsce docelowe lotu może się wielokrotnie powielać, o tyle numer rezerwacji jest zawsze inny, ponieważ przyjmuję, że jedna tabela odpowiada konkretnemu tygodniowi czy miesiącu a liczba rezerwacji mieści się w podanym przedziale czasowym.

2. TICKETS

ticket_id (INT) - prim-key, not-null, unique, auto-increment

booking_id (INT) - not-null

ticket (VARCHAR(8)) - not-null,

passenger_id(INT) - not-null, unique

segments_id(INT) - not-null, unique

ancillary_id(INT) - not-null, unique

Tak sam jak bilet lotniczy zawiera wszystkie najważniejsze informacje dotyczące lotu, tak podobną funkcję ma tabela TICKETS. Jest w relacji z innymi tabelami dotyczącymi: danych pasażera(id_passengers), dodatkowych usług(ancillary_id), czy odcinków podróży(segments_id).

Ticket_id to kolumna nadająca automatycznie unikalny i kolejny numer dla biletu. Nie wymaga wprowadzania dzięki opcji auto-inkrementacji, nie może się powielać(jeden konkretny bilet, na jedną osobę).

Kolumna booking_id posiada tylko jedną wartość – nie może być pusta. Może się za to powielać, ponieważ istnieje opcja kupna kilku biletów na jedną rezerwację. Nie jest kluczem głównym, ponieważ odnosi się do tabeli z rezerwacjami.

Kolumna ticket to nazwa biletu, przeznaczyłem na to 8 znaków, ponieważ nazwa składa się z dwóch Kodów IATA- lotniska z którego wyruszamy, oraz docelowego(bez uwzględniania przesiadek) oddzielonych myślnikiem, np. „WAW-MAD”.

Ostatnie trzy kolumny to klucze, które tworzą relację z odpowiadającymi im tabelami.

3. PASSENGERS

passenger_id(INT) prim-key, not-null, unique,

first_name (VARCHAR(20)) not-null,

last_name (VARCHAR(25)) not-null,

phone_number (VARCHAR(15)) not-null,

mail (VARCHAR(45)) not-null,

Tabela zawiera dane wszystkich pasażerów, którzy złożyli rezerwację. Kolumna passenger_id to jak sama nazwa wskazuje – numer id każdego pasażera, musi być unikalny. Jest kluczem głównym tej tabeli.

W przypadku Kolumny phone_number nie zastosowałem zmiennej INT, ponieważ niektóre numery mogą być wprowadzane ze znakiem „+” na początku – który nie jest integerem (na przykład numer kierunkowy do polski – zaczynający się od +48).

Jednak zdecydowanie lepszym rozwiązaniem było by ustalenie jednego sposobu wprowadzania numerów, na przykład bez znaku na początku, z numerem kierunkowym, bez spacji (48123456789)

Na adres e-mail(kolumna mail) przeznaczyłem aż 45 znaków, ponieważ zdaję sobie sprawę, że niektóre maile związane z firmami czy instytucjami są długie.

4. **ANCILLARY**

ancillary_id (INT) prim-key, not-null, unique,
additional_baggage (VARCHAR(3)) not-null,
paid_place (VARCHAR(3)) not-null,
insurance (VARCHAR(3)) not-null,
paid_food (VARCHAR(3)) not-null

Pierwsza kolumna, pełni dokładnie taką samą rolę jak kolumna passenger_id w poprzedniej tabeli. Jest to unikalny numer będący w relacji z tabelą tickets, służący do przekazywania informacji o dodatkowych udogodnieniach.

Następne cztery kolumny zawierają informację o wybraniu (wartość „YES”) bądź rezygnacji (wartość „NO”) z konkretnej usługi.

5. SEGMENTS

segment_id(INT) prim-key, not-null,

segment(VARCHAR(8)) not-null,

airport_id(INT) not-null,

Tutaj sytuacja jest nieco inna. Dlatego że dla każdego biletu może być (w teorii) nieskończona ilość przesiadek, a każdy wiersz opisuje jeden lot bez przerwy, czy przesiadki. To znaczy że mając bilet Warszawa – Madryt w jedną stronę, możemy mieć w przypadku lotu z przesiadkami dwa wiersze („WAW-BER”, oraz „BER-MAD”), dla których pierwsza kolumna przyjmuje tę samą wartość.

Airport_id obsługuje relacje z tabelą zawierającą informacje o konkretnym lotnisku. Wprowadzane tutaj id odwołuje się do lotniska na którym czeka nas następne lądowanie.

6. AIRPORTS

airport_id(INT) prim-key, not-null, unique, auto-increment

airport_shortcut(VARCHAR(3)) not-null,

city(VARCHAR(30)) not-null,

country(VARCHAR(20)) not-null,

continent(VARCHAR(15)) not-null,

airport(VARCHAR(120)) not-null,

airport_id to klucz, dzięki któremu możemy przyporządkować lotnisko, do lotu, nie może się dublować, każde lotnisko ma swój własny numer, przysypisywany automatycznie.

airport_shortcut – kod IATA lotniska

Poniżej przedstawiam przykładowe wartości, które umieściłem w tabelach.:

```
INSERT INTO bookings(booking) VALUES ("TOBER_842"), ("TOLHR_843"), ("TOMAD_844");

INSERT INTO tickets(booking_id,ticket,passenger_id,segments_id,ancillary_id) VALUES
(1,"WAW-BER",1,1,1),
(2,"WAW-LHR",2,2,2),
(3,"WAW-MAD",3,3,3);

INSERT INTO passengers(passenger_id,first_name,last_name,phone_number,mail) VALUES
(1,"Kacper","Płotkowiak","721247749","kacperowsky14@gmail.com"),
(2,"Jan","Kowalski","123 456 789","j.kowalski@gmail.com"),
(3,"Adam","Przykładowy","+48 123 456 999","adamprzykladowy@gmail.com");

INSERT INTO ancillary(ancillary_id,additional_baggage,paid_place,insurance,paid_food) VALUES
(1,"YES","NO","YES","YES"),
(2,"NO","NO","YES","NO"),
(3,"NO","NO","YES","YES");
```

c.d:

```
INSERT INTO segments(segment_id,segment,airport_id) VALUES
(1,"WAR-BER",2),
(2,"WAW-LHR",3),
(3,"WAW-MAD",4),
(4,"BER-WAW",1),
(5,"LHR-WAW",1),
(6,"MED-WAW",1),

(7,"BER-LHR",3),
(8,"BER-MAD",4),
(9,"MAD-BER",2),
(10,"LHR-BER",2),

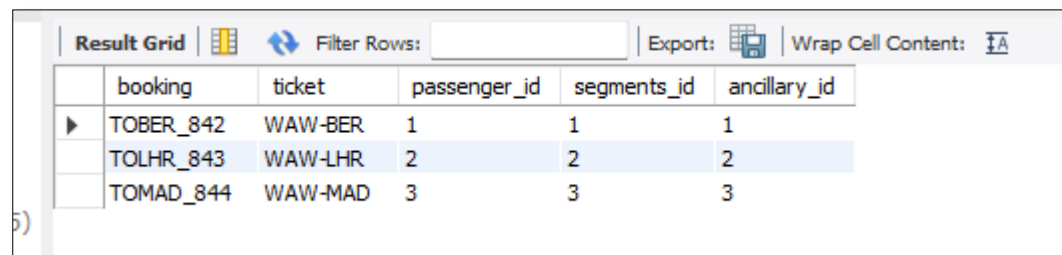
(11,"MAD-LHR",3),
(12,"LHR-MAD",4);

INSERT INTO airports(airport_id,airport_shortcut,city,country,continent,airport) VALUES
(1,"WAW","Warsaw","Poland","Europe","Warsaw Chopin Airport"),
(2,"BER","Berlin","Germany","Europe","Berlin Brandenburg Airport"),
(3,"LHR","London","England","Europe","London Heathrow Airport"),
(4,"MAD","Madrid","Spain","Europe","Adolfo Suárez Madrid-Barajas Airport");
```


Za pomocą tak zaprojektowanej bazy danych, możemy pozyskiwać dowolne informacje dotyczące lotu, rezerwacji czy biletu, na przykład:

```
SELECT booking, ticket, passenger_id, segments_id, ancillary_id
FROM bookings
INNER JOIN tickets
ON bookings.booking_id = tickets.ticket_id;
```

wynik:



The screenshot shows a database query result grid with the following columns: booking, ticket, passenger_id, segments_id, and ancillary_id. The grid contains three rows of data. The first row is TOBER_842, WAW-BER, 1, 1, 1. The second row is TOLHR_843, WAW-LHR, 2, 2, 2. The third row is TOMAD_844, WAW-MAD, 3, 3, 3. The grid is titled 'Result Grid' and has a 'Filter Rows' field and an 'Export' button. The 'Wrap Cell Content' option is also visible.

	booking	ticket	passenger_id	segments_id	ancillary_id
▶	TOBER_842	WAW-BER	1	1	1
	TOLHR_843	WAW-LHR	2	2	2
	TOMAD_844	WAW-MAD	3	3	3