CSE 3055 DATABASE SYSTEMS PROJECT

Travel Agency

Team Members:

Göksel Tokur - 150116049

Merve Ayer - 150119828

Furkan Aras - 150116019

Project Description:

Travel agency sites come handy in times when you need travel services. Planning for your vacation, particularly during holiday seasons can be a lot less pricing if you have contacted a reputable travel agent to help you. A travel agency database system may contain many bookings, payment and customer information. For this reason, travel agencies need to use a well-designed database system to keep a record of their customers and bookings.

The database allows for;

- Quick book hotel, flight, railway or car.
- Tracking customers' bookings.
- Keep customers' information such as phone number, address to provide easy communication between agency and the customer.
- Easy tracking of payment and income.
- Tracking booking outcomes.
- Tracking booking/payment status.
- Less payment errors.
- Booking records will be stored and can be accessed

Data and Requirement Analysis:

A company has several travel agency that has attributes of the travel agency identifier and travel agency name. A booking can be different number of services such as hotel room, airlines, cars and railway. These bookings consist of product identifier that can be service bookings. We identify the following additional attributes for bookings: customer identifier, outcome identifier, status identifier, travel agency identifier, date of booking and booking details.

A travel agency may group any number of bookings. A booking must belong to a travel agency.

Each booking must have a booking outcome that describes the success of the booking with outcome identifier, and outcome status as canceled or OK. Booking outcomes may have bookings optionally.

Each booking must have booking status attribute that has status identifier and status description such as provisional or precise. Booking status entities may have bookings optionally.

Customers can book services. A customer will be identified by a unique customer ID and consists of customer details such as name, surname and phone number. Customers may have optional many customer addresses that consist of date from and date to attributes and must have addresses entity which has address identifier and address details. Each booking is booked by at least one customer.

Payments of bookings will be consisting of payment identifier, booking identifier, payment amount, payment date and other details if it is necessary. Bookings optionally may have many payments. Payments must have booking.

Supertype services entity will be exist with subtypes: airlines, cars, hotels and railways. The identifier for a service is service id and another attribute is service details.

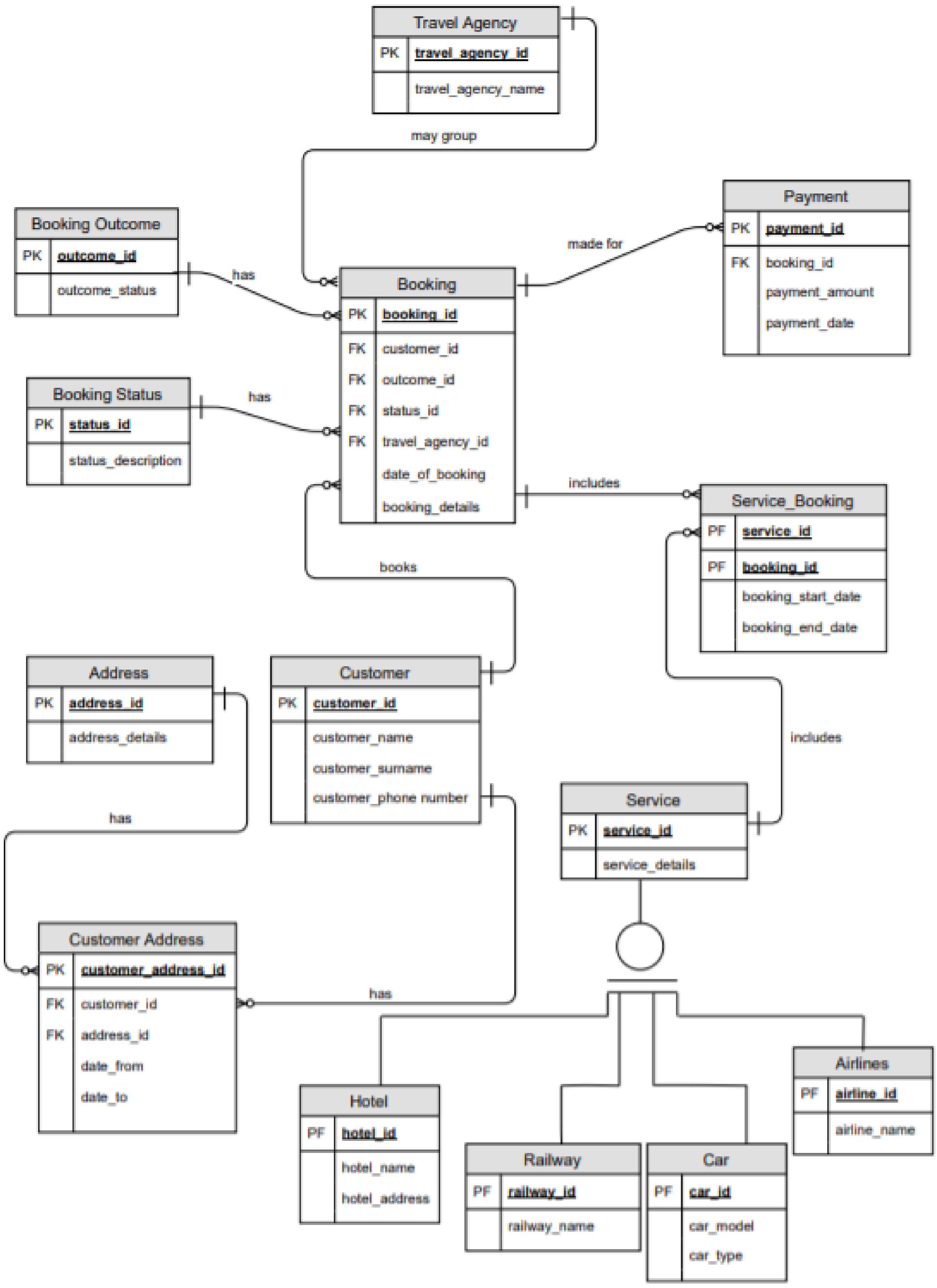
Airlines entity has an airline identifier and airline name attributes.

Cars entity has car id, car model and car type attributes such as compact car, SUV etc.

Railways entity has railway id, railway name.

Hotels entity has hotel identifier, hotel name, address attributes.

E-R Diagram



Travel Agency
PK travel_agency_id
travel_agency_name
may group
Payment
PK payment_id
FK booking_id
payment_amount
payment_date
made for
Booking Outcome
PK outcome_id
outcome_status
has
Booking
PK booking_id
FK customer_id
FK outcome_id
FK status_id
FK travel_agency_id
date_of_booking
booking_details
Booking Status
PK status_id
status_description
has
includes
Service_Booking
PF service_id
PF booking_id
booking_start_date
booking_end_date
books
Address
PK address_id
address_details
Customer
PK customer_id
customer_name
customer_surname
customer_phone number
includes
Service
PK service_id
service_details
has
Customer Address
PK customer_address_id
FK customer_id
FK address_id
date_from
date_to
has
Hotel
PF hotel_id
hotel_name
hotel_address
Railway
PF railway_id
railway_name
Car
PF car_id
car_model
car_type
Airlines
PF airline_id
airline_name

About Tables:

- Travel_Agency Table: Contains the name and id of travel agency.

Travel_Agency				
Attribute	Data Type	Primary Key	Foreign Key	Notes
travel_agency_id	int	+		NOT NULL
travel_agency_name	Varchar(50)			NOT NULL

- Booking_Status Table: Contains the status of booking.

Booking_Status				
Attribute	Data Type	Primary Key	Foreign Key	Notes
status_id	int	+		NOT NULL
status_description	Varchar(100)			

- Booking Table: Contains the information about booking.

Booking				
Attribute	Data Type	Primary Key	Foreign Key	Notes
booking_id	int	+		NOT NULL
customer_id	int		+ (Customer)	
outcome_id	int		+ (Booking_Outcome)	
status_id	int		+ (Booking_Status)	
travel_agency_id	int		+ (Travel_Agency)	
date_of_booking	DATETİME			
booking_details	Varchar(100)			

- Customer Table: Contains customer's information.

Customer				
Attribute	Data Type	Primary Key	Foreign Key	Notes
customer_id	int	+		NOT NULL
customer_name	Varchar(20)			
customer_surname	Varchar(20)			
customer_phone_number	Varchar(20)			

- Customer_Address Table: Contains the address of customer and date time

Customer_Address				
Attribute	Data Type	Primary Key	Foreign Key	Notes
customer_address_id	int	+		NOT NULL
customer_id	int		+ (Customer)	
address_id	Int		+ (Address)	
date_from	DATETIME			
date_to	DATETIME			

- Booking_Outcome Table: States the booking outcome.

Booking_Outcome				
Attribute	Data Type	Primary Key	Foreign Key	Notes
outcome_id	int	+		NOT NULL
Outcome_status	Varchar(100)			

- Payment Table: Includes the cost of the reservations and when to pay.

Payment				
Attribute	Data Type	Primary Key	Foreign Key	Notes
Payment_id	int	+		NOT NULL
Booking_id	int		+ (booking)	
Payment_amount	float			
Payment_date	DATETIME			
Other_details	Varchar(100)			NULL

- Address Table: Includes addres id and address details.

Address				
Attribute	Data Type	Primary Key	Foreign Key	Notes
address_id	int	+		NOT NULL
address_details	Varchar(100)			

- Service Table: Contains service id and details.

Service				
Attribute	Data Type	Primary Key	Foreign Key	Notes
Service_id	int	+		NOT NULL
Service_details	Varchar(100)			

- Service_Booking Table: Includes when the booking starts and ends.

Service_Booking				
Attribute	Data Type	Primary Key	Foreign Key	Notes
Service_id	int	+	+ (Service)	NOT NULL
booking__id	int	+	+ (Booking)	NOT NULL
Booking_start_date	DATETIME			
Booking_end_date	DATETIME			

- Hotel Table: Contains hotel identifier, hotel name, address attribute.

Hotel				
Attribute	Data Type	Primary Key	Foreign Key	Notes
hotel_id	int	+	+ (Service)	NOT NULL
Hotel_name	Varchar(60)			
Hotel_address	Varchar(100)			

- Airlines Table: Contains airline identifier and airline name attributes.

Airlines				
Attribute	Data Type	Primary Key	Foreign Key	Notes
airline_id	int	+	+ (Service)	NOT NULL
airline_name	Varchar(100)			

• Railway Table: Contains railway identifier and railway name attributes.

Railway				
Attribute	Data Type	Primary Key	Foreign Key	Notes
Railway_id	int	+	+ (Service)	NOT NULL
Railway_name	Varchar(60)			

• Car Table: Contains car id, car model and car type attributes.

Car				
Attribute	Data Type	Primary Key	Foreign Key	Notes
car_id	int	+	+ (Service)	NOT NULL
car_model	Varchar(60)			
car_type	Varchar(60)			

• Booking_log Table : Keeps a record of the transactions in the Booking Table.

Booking_log				
Attribute	Data Type	Primary Key	Foreign Key	Notes
booking_id	Varchar(8)			NOT NULL
performed_action	Varchar(10)			
action_date	DATETIME			

Constraints:

PRIMARY KEY Constraint:

• travel_agency_id uniquely identify each row in a Travel_Agency Table. Therefore, it is primary key.

• outcome_id uniquely identify each row in a Booking_Outcome Table. Therefore, it is primary key.

- status_id uniquely identify each row in a Booking_Status Table. Therefore, it is primary key.
- customer_id uniquely identify each row in a Customer Table. Therefore, it is primary key.
- booking_id uniquely identify each row in a Booking Table. Therefore, it is primary key.
- payment_id uniquely identify each row in a Payment Table. Therefore, it is primary key.
- service_id uniquely identify each row in a ServiceTable. Therefore, it is primary key.
- service_id and booking_id are both uniquely identifies each row in a Service_Booking Table. Therefore, they are primary keys.
- address_id uniquely identify each row in a Address Table. Therefore, it is primary key.
- customer_address_id uniquely identify each row in a Customer_Address Table. Therefore, it is primary key.
- hotel_id uniquely identify each row in a Hotel Table. Therefore, it is primary key.
- airline_id uniquely identify each row in a Airlines Table. Therefore, it is primary key.
- car_id uniquely identify each row in a Car Table. Therefore, it is primary key.
- railway_id uniquely identify each row in a Railway Table. Therefore, it is primary key.

NOT NULL Constraint:

- Travel_agency_id is the primary key in the Travel_Agency Table, so it is NOT NULL.
- outcome_id is the primary key in the Booking_Outcome Table, so it is NOT NULL.
- status_id is the primary key in the Booking_Status Table, so it is NOT NULL.
- customer_id is the primary key in the Customer Table, so it is NOT NULL.
- booking_id is the primary key in the Booking Table, so it is NOT NULL.
- payment_id is the primary key in the PaymentTable, so it is NOT NULL.
- service_id is the primary key in the Service Table, so it is NOT NULL.
- service_id and booking_id are primary keys in the Service_Booking Table, so they are NOT NULL.
- address_id is the primary key in the Address Table, so it is NOT NULL.

• customer_address_id is the primary key in the Customer_Address Table, so it is NOT NULL.

• hotel_id is the primary key in the HotelTable, so it is NOT NULL.

• railway_id is the primary key in the RailwayTable, so it is NOT NULL.

• car_id is the primary key in the CarTable, so it is NOT NULL.

• airline_id is the primary key in the Airline Table, so it is NOT NULL.

UNIQUE Constraint:

• Travel_agency_id only have unique value and not have duplicate data in the Travel_Agency Table so it is UNIQUE.

• outcome_id only have unique value and not have duplicate data in the Booking_Outcome Table so it is UNIQUE.

• status_id only have unique value and not have duplicate data in the Booking_Status Table so it is UNIQUE.

• customer_id only have unique value and not have duplicate data in the Customer Table so it is UNIQUE.

• booking_id only have unique value and not have duplicate data in the Booking Table so it is UNIQUE.

• payment_id only have unique value and not have duplicate data in the Payment Table so it is UNIQUE.

• service_id only have unique value and not have duplicate data in the Service Table so it is UNIQUE.

• service_id and booking_id are both only have unique value and not have duplicate datain the Service_Booking Table so they are UNIQUE.

• address_id only have unique value and not have duplicate data in the Address Table so it is UNIQUE.

• customer_address_id only have unique value and not have duplicate data in the Customer_Address Table so it is UNIQUE.

• hotel_id only have unique value and not have duplicate data in the Hotel Table so it is UNIQUE.

• airline_id only have unique value and not have duplicate data in the Airlines Table so it is UNIQUE.

• car_id only have unique value and not have duplicate data Car Table so it is UNIQUE.

• railway_id only have unique value and not have duplicate data in the Railway Table so it is UNIQUE.

FOREIGN KEY Constraint:

• travel_agency_id in Booking Table is the primary key in the Travel Agency Table. Therefore it is a foreign key in Booking Table. customer_id in Booking Table is the primary key in the Customer Table. Therefore it is foreign key in the Booking Table. outcome_id in Booking Table is the primary key in the Booking_Outcome Table. Therefore it is foreign key in the Booking Table. status_id in Booking Table is the primary key in the Booking_Status Table. Therefore it is foreign key in the Booking Table.

• booking_id in Payment Table is the primary key in the Booking Table. Therefore it is a foreign key in the Payment Table.

• service_id in Service_Booking Table is the primary key in the Service Table. Therefore it is a foreign key in the Service_Booking Table. booking_id in Service_Booking Table is the primary key in the Booking Table. Therefore it is a foreign key in the Service_Booking Table.

• customer_id in Customer_Address Table is the primary key in the Customer Table. Therefore it is a foreign key in the Customer_Address Table. addres_id in Customer_Address Table is the primary key in the Address Table. Therefore it is a foreign key in the Customer_Address Table.

• hotel_id in Hotel Table is the primary key in the Hotel Table. Hotel Table is subtype of the Service Table. hotel_id references service_id in Service Table. Therefore hotel_id is foreign key in the Hotel Table.

• airline_id in Airlines Table is the primary key in the Airlines Table. Airlines Table is subtype of the Service Table. airline_id references service_id in Service Table. Therefore airline_id is foreign key in the Airlines Table.

• railway_id in Railway Table is the primary key in the Railway Table. Railway Table is subtype of the Service Table. railway_id references service_id in Service Table. Therefore railway_id is foreign key in the Railway Table.

• car_id in Car Table is the primary key in the Car Table. Car Table is subtype of the Service Table. car_id references service_id in Service Table. Therefore car_id is foreign key in the Car Table.

CHECK Constraint:

• Create a table Service_Booking and specifies the condition for the field booking_end_date and booking_start_date as CHECK (booking_end_date >= booking_start_date).That is, does not allow booking_start_date to be greater than booking_end_date.

DEFAULT Constraint:

• Create a table named Booking and specify the default value for the field date_of_booking as GETDATE(). If date_of_booking is not entered, the default value is assign as the date for that day.

• Create a table named Payment and specify the default value for the field other_details as NULL. If other_details is not entered, the default value is assign as the NULL.

Indices:

We have primary keys for each table, so that means we directly have one clustered index for each table.

```
CREATE INDEX index_fullname
ON Customer (customer_name, customer_surname);
```

• This index was created to reach the full name directly.

Triggers:

```
CREATE TRIGGER insert_Booking_Log ON Booking
AFTER INSERT AS
BEGIN

    INSERT INTO Booking_Log(booking_id, performed_action, action_date)
    select i.booking_id, 'Insert', GETDATE() FROM inserted i;

END;
```

• The generated trigger, records the insert operations made in the Booking Table and the time it was made into the Booking_log Table.

```
create TRIGGER delete_Booking_Log ON Booking
AFTER DELETE AS
BEGIN
       DECLARE @hasPayment int;
       DECLARE @hasServiceBooking int;

       select @hasPayment=COUNT(*) from Payment
       where Payment.booking_id=(select booking_id from deleted)

       select @hasServiceBooking=COUNT(*) from Service_Booking
       where Service_Booking.booking_id=(select booking_id from deleted)

      IF ((@hasPayment) > 0 OR (@hasServiceBooking) > 0)
       BEGIN
   RAISERROR('You cannot delete corresponding booking it has records.',16, 1)
       ROLLBACK TRANSACTION --rollback delete operation
       END

        IF ((@hasPayment) = 0 AND (@hasServiceBooking) = 0)
       BEGIN
   INSERT INTO Booking_Log(booking_id, performed_action, action_date)
    select i.booking_id, 'Delete', GETDATE() FROM deleted i;
       END

END;
```

• This trigger keeps the delete operations made in the Booking Table and the time it was made into the Booking_log Table.

```
CREATE TRIGGER update_Booking_Log ON Booking
AFTER UPDATE AS
BEGIN

    INSERT INTO Booking_Log(booking_id, performed_action, action_date)
    select i.booking_id, 'Update', GETDATE()   FROM inserted i;

END;
```

• The trigger, records the update operations made in the Booking Table and the time it was made into the Booking_log Table.

Stored Procedures:

```
CREATE PROCEDURE BookingsAboveSpecificNumberOfDays
       @numberOfDay int
AS
SELECT booking_id, DATEDIFF(day , booking_start_date, booking_end_date) AS
number_of_days, booking_start_date, booking_end_date
FROM Service_Booking AS s
WHERE DATEDIFF(day , booking_start_date, booking_end_date) >= @numberOfDay;


exec BookingsAboveSpecificNumberOfDays 4;
```

- This stored procedure shows bookings over a specified number of days. This stored procedure shows bookings over 4 days.

```
CREATE PROCEDURE GetGreaterPayments
       @amount int
AS
SELECT *
FROM Payment AS p
WHERE p.payment_amount >= @amount;


exec GetGreaterPayments 500;
```

- The stored procedure shows payment amount over 500.

Views:

```
CREATE VIEW Paid_Payments AS
SELECT payment_id, other_details, payment_amount, payment_date
FROM Payment
WHERE other_details = 'Paid';
```

- This view was created to kept bookings that have been paid.

```
CREATE VIEW Revenue_of_Each_Agency AS
SELECT B.travel_agency_id, SUM(P.payment_amount) AS TotalAmount
FROM Booking AS B
LEFT OUTER JOIN Payment AS P ON P.booking_id = B.booking_id
GROUP BY  B.travel_agency_id;
```

- This view was created to kept the revenue of each agency.

Computed Column:

```
ALTER TABLE dbo.Service_Booking ADD number_of_days AS DATEDIFF(day ,
booking_start_date, booking_end_date);
```

• This code automatically calculates the number of days from the day difference between booking_start_date and booking_end_date.

Identity Column:

```
customer_id int NOT NULL UNIQUE IDENTITY (1,1)

booking_id int NOT NULL UNIQUE IDENTITY (1,1)
```

• With the IDENTITY keyword, customer_id and booking_id are automatically incremented from 1 onwards.