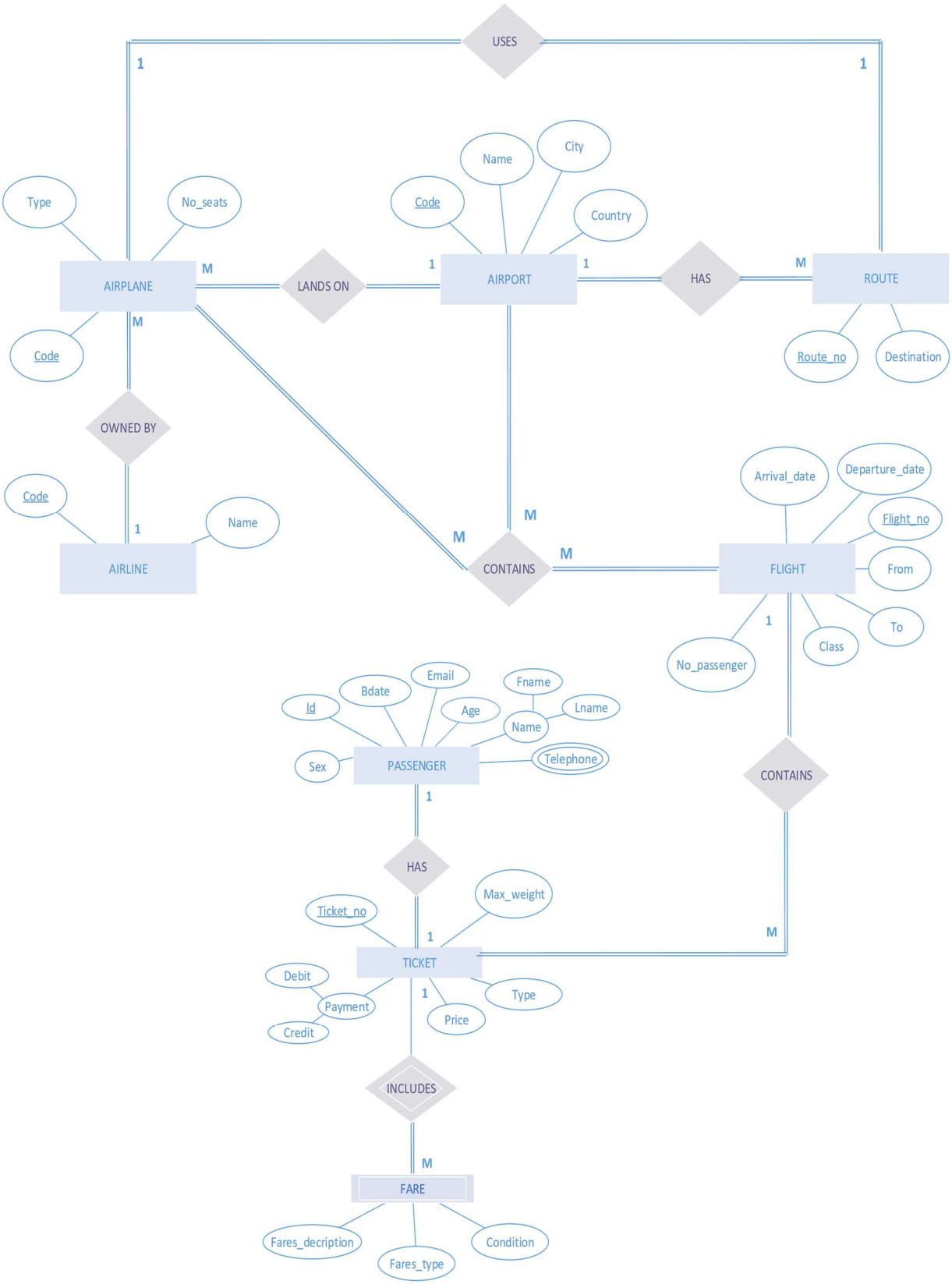
Airline Online Reservation

Name	ID	E-mail
Aalaa		
Abdelrahman		
Ahmed		
Clara		
Roudina		

Scope

The name of our database is " AIRLINE ONLINE RESERVATION DATABASE " . With the help of this system , customers can view all different flight's availability with different timings for a particular date and it also allow them to reserve a seat , cancel reservation or modify it . The database is very flexible to insert new data or modify an existence data about a passenger , add or delete columns , retrieve data about specific passenger, remove table with all constraints that make it secure (not to allow anyone to make changes in it) .


USES
1
1
Type
No_seats
Code
AIRPLANE
M
LANDS ON
1
Name
City
Code
Country
AIRPORT
1
HAS
M
ROUTE
Route_no
Destination
M
OWNED BY
1
Code
Name
AIRLINE
M
M
CONTAINS
M
Arrival_date
Departure_date
Flight_no
From
To
Class
1
No_passenger
FLIGHT
Email
Bdate
Fname
Id
Age
Lname
Name
Sex
PASSENGER
Telephone
CONTAINS
1
HAS
Max_weight
Ticket_no
1
M
TICKET
Debit
Payment
Credit
1
Type
Price
INCLUDES
M
FARE
Fares_decription
Fares_type
Condition

Passenger

ID	Fname	Lname	Sex	Email	Bdate	Ticket_no

Phone

ID	Telephone

Ticket

No	Price	Credit	Debit	Flight_no	Type	Max_weight

Fare

Fares_description	Condition	Fares_type	Ticket_no

Route

No	Destination	Airport_code	Airplane_code

Airline

code	Name

Airport

City	Country	code	Name

Airplane

code	Type	No_seats	Airline_code

Flight

No	From	To	Class	Arrival_date	Depature_date	No_passenger

Plane_flight_port

Airplane_code	Airport_code	Flight_no

Airport-Airplane

Airport_code	Airplane_code

Description

Let's explain our Database in a simple way to be understandable :

First of all , We have tables connected with each other to facilitate the data access path in the database.

Tables (Entities) with their columns (Attributes) :

1. Here we have the first table which is called PASSENGER that has more than one column (attribute) : ID (which is considered to be the primary key of this table so it will be underlined) , Name (which is considered as a composite attribute of fname , lname so we write them directly as columns) , sex , BDate , Email , age , telephone (which is considered as multivalued attribute so we will take it off from this table to be in another table with the primary key) .

2. The Second table is called TICKET that has more than one column (attribute) : No (which is considered to be the primary key of this table so it will be underlined) , price , type , max_weight , payment which is considered as a composite attribute of credit , debit so we write them directly as columns) .

3. The Third table is called FARES that has more than one column (attribute) : type , description , condition . It is considered as weak entity.

4. The Fourth table is called FLIGHT that has more than one column (attribute) : No (which is considered to be the primary key of this table so it will be underlined) , class , from , to , no_of_passengers , arrival_date, departure_date .

5. The Fifth table is called AIRPLANE that has more than one column (attribute) : code (which is considered to be the primary key of this table so it will be underlined) , type , no_of_seats .

6. The Sixth table is called AIRLINE that has more than one column (attribute) : code (which is considered to be the primary key of this table so it will be underlined) , name .

7. The Seventh table is called AIRPORT that has more than one column (attribute) : code (which is considered to be the primary key of this table so it will be underlined) , name , city , country .

8. The Eighth table is called ROUTE that has more than one column (attribute) : No (which is considered to be the primary key of this table so it will be underlined) , destination .

Relations between these tables (Entities) :

1. [has] :
    - Between PASSENGER & TICKET tables .
    - Its cardinality : 1 to 1 relationship cause each passenger has only one ticket .
    - Its participation : must – must cause each passenger must have a ticket .

Because of its cardinality (1 to 1) , we take one of the primary keys of each table & put it in the other one as foreign key . Here we take Ticket_no (primary key of TICKET table) & put it in the PASSENGER table as foreign key .

2. [include] :
    - Between TICKET & FARES tables .
    - Its cardinality : 1 to M relationship cause each ticket include more than one fares .
    - Its participation : may – must cause each ticket may include fares or not .
    - It's considered as a Weak Relationship .

Because of its cardinality (1 to M) , we take the primary key of the 1-side & put it in the M-side as foreign key . Here we take Ticket_no (primary key of TICKET table) & put it in the FARES table as foreign key .

3. [contain] :
    - Between FLIGHT & TICKET tables .
    - Its cardinality : 1 to M relationship cause each flight has more than one ticket .
    - Its participation : must – must cause each flight must have a ticket .

Because of its cardinality (1 to M) , we take the primary key of the 1-side & put it in the M-side as foreign key . Here we take Flight_no (primary key of FLIGHT table) & put it in the TICKET table as foreign key .

4. [contain] :
    - Between AIRPLANE , AIRPORT & FLIGHT tables (Ternary Relationship).
    - Its cardinality : M to M to M relationship cause airport has more than one airplane & can manage more than one flight .
    - Its participation : must – must-must cause airport must have more than one airplane & must manage more than one flight .

Because of its cardinality (M to M to M) , we take the primary keys of each table (airport_code , airplane_code , flight_no) & put them in a new table which is called P-F-P . All of them together will be the primary key of the new table (P-F-P table).

5. [owned by] :
    - Between AIRLINE & AIRPLANE tables .
    - Its cardinality : 1 to M relationship cause an airline own many airplanes .
    - Its participation : must – must cause an airline must own airplanes & airplanes must be owned by airline .

Because of its cardinality (1 to M) , we take the primary key of the 1-side & put it in the M-side as foreign key . Here we take airline_code (primary key of AIRLINE table) & put it in the AIRPLANE table as foreign key .

6. [land on] :
    - Between AIRPORT & AIRPLANE tables .
    - Its cardinality : 1 to M relationship cause many airplanes lands on one airports
    - Its participation : must – must cause airplane must land on airport .

Because of its cardinality (1 to M) , we take the primary key of the 1-side & put it in the M-side as foreign key , But we take both foreign keys & put them in different table . We put airplane_code & airport_code in that table , airplane_code refers to (code) in AIRPLANE table & airport_code refers to (code) in AIRPORT table .

7. [has] :
    - Between AIRPORT & ROUTE tables .
    - Its cardinality : 1 to M relationship cause an airport has many routes .
    - Its participation : must – must cause an airport must has routes .

Because of its cardinality (1 to M) , we take the primary key of the 1-side & put it in the M-side as foreign key . Here we take airport_code (primary key of AIRPORT table) & put it in the ROUTE table as foreign key .

8. [use] :
    - Between AIRPLANE & ROUTE tables .
    - Its cardinality : 1 to 1 relationship cause an airplane uses only one route .
    - Its participation : must – must cause an airplane must use a route .

Because of its cardinality (1 to 1) , we take one of the primary keys of each table & put it in the other one as foreign key so we take airplane_code & put it in ROUTE table as foreign key.

Tables after simplifying Relationships :

1. PASSENGER
    - Columns : ID , fname , lname , sex , BDate , Email ,age , Ticket_no .
    - Primary key : ID .
    - Foreign key : Ticket_no refers to (No) in TICKET table .

2. PASSENGER_TELEPHONE
    - Columns : ID , telephone .
    - Primary key : ID , telephone together
    - Foreign key : ID refers to (ID) in PASSENGER table .

3. TICKET
    - Columns : No , type , price , max_weight , credit , debit , Flight_no .
    - Primary key : No .
    - Foreign key : Flight_no refers to (No) in FLIGHT table .

4. FARES
    - Columns : type , description , condition , Ticket_no .
    - Primary key : no primary key
    - Foreign key : Ticket_no refers to (No) in TICKET table .

5. FLIGHT
    - Columns : No , class , from , to , arrival_date , departure_date , no_of_passenger
    - Primary key : No .
    - Foreign key : no foreign key

6. AIRPLANE
    - Columns : code , type , no_of_seats , airline_code .
    - Primary key : code .
    - Foreign key : airline_code refers to (code) in AIRLINE table .

7. AIRLINE
    - Columns : code , name .
    - Primary key : code .
    - Foreign key : no foreign key .

8. AIRPORT
    - Columns : code , name , city , country .
    - Primary key : code .
    - Foreign key : no foreign key .

9. ROUTE
    - Columns : No , destination , airport_code , airplane_code .
    - Primary key : No .
    - Foreign key : airport_code refers to (code) in AIRPORT table , airplane_code refers to (code) in AIRPLANE table.

10. PLANE-FLIGHT-PORT
    - Columns : airport_code , flight_no , airplane_code .
    - Primary key : airport_code , flight_no & airplane_code together .
    - Foreign key : airport_code refers to (code) in AIRPORT table , flight_no refers to (No) in FLIGHT table , airplane_code refers to (code) in AIRPLANE table .

11. AIRPORT_AIRPLANE
    - Columns : airport_code , airplane_code .
    - Primary key : Both .
    - Foreign key : airport_code refers to (code) in AIRPORT table , airplane_code refers to (code) in AIRPLANE table.

Queries :

1- **First query :** select all from passenger **,** ticket where price > 2000.
(discuss the operational operator bigger than 2000) .

2- **Second query:** select flight number , no of flights that travels in 2020 .
(discuss the aggregate function count that counts the number of flights that travel during 2020) .

3- **Third query :** select fare type , fname as first name, lname as last name , birthdate where birthyaear >= 2000 .
(discuss the inner join that is used to join data in more than one table . Here we join FARES with TICKET with PASSENGER by fare.ticket_no = ticket.t_no & ticket.t_no = passenger.ticket_no) .

4- **Forth query :** select route number , destination , class ,flight number where class of flight = ' A ' &group by flight number .
(discuss the inner join that is used to join data in more than one table . Here we join ROUTE with AIRPLANE with PLANE_FLIGHT_PORT by route.plane_code = airplane.plane_code & airplane.plane_code = plane_flight_port.plane_code & flight.flight_no = plane.flight.port.plane_code) .

5- **Fifth query :** select number of flights , flight no that travels to ' BOM ' group by flight no .
(discuss count aggregate function) .

6- **Sixth query :** select fname , lname as passenger name that flight travel to ' FRA ' , order by passenger name .
(discuss the inner join that is used to join data in more than one table . Here we join FLIGHT with TICKET with PASSENGER by passenger.ticket_no = ticket.t_no & ticket.flight_no = flight.flight_no).

7- **Seventh query :** select flight numbers that arrived in April & count there numbers group by flight number .
(discuss count aggregate function to count no of flights in April) .

8- **Eighth query :** select airline code , airline name , flight no that travelled at 02 .
(discuss the inner join that is used to join data in more than one table . Here we join AIRLINE with AIRPLANE with FLIGHT with PLANE_FLIGHT_PORT by airline.airline_code = airplane.airplane_code & plane_flight_port.plane_code = airplane.airplane_code & flight.flight_code = plane_flight_port.flight_no , where date = 02) .

9- **Ninth query :** select * from passenger whose name contains letter ' a ' .
(discuss alphabetical letters to select the name of passenger whose fname has letter ' a ' in his name).

10-Tenth query : select ticket type, fname , lname as full name whose gender is male .

(discuss concatenation between two attribute to be one attribute & discuss inner join between TICKET with PASSENGER by ticket.t_no = passenager.ticket_no where passenager's gender is male ' M ').

From 11 → 14 : Nested Queries

11-Eleventh query : select fname , lname as passenger name whose flight is booked by credit card.

(discuss inner join between PASSENGER with TICKET by passenger.ticket_no = ticket.t_no).

12-twelvth query : select fname , lname as full name , ticket no whose ticket included fares without duplication .

13- Thirteenth query : select all from ticket that has maximimum ticket price .

(discuss maximum price from the TICKET tabe).

14-Fourteenth query : select all from passenger whose sum of their tickets is bigger than 3500 then group by them by ticket no .

(discuss aggregate function that return total summation of their tickets).

15- Fifteenth query : select fname , lname as full name whose fname contains ' a ' , lname contains ' m ' & gender is female .

(discuss inner join between PASSENGER with TICKET by ticket.t_no = passenger.ticket_no where gender is female ' F ' , fname contains letter ' a ' , lname contains letter ' m ').

16- Sixteenth query : select all from ticket , fares whose ticket no has fares & gender is female .

(discus nested query & full outer join between TICKET with FARES by ticket.t_no = fare.ticket_no) .