

# Aryan Shrivastava

## 1. MiniWorld

The mini-world Airport Management System (AMS) is designed to manage and streamline critical airport operations by tracking essential details such as flight schedules, passenger information, and airport routes.

Users of the above database are:

- Passengers - To book, reschedule and cancel their flight.
- Airline Employees - To manage flight schedules, modify bookings, and analyze flight traffic patterns.

At the end of the report, we will be able to:

- Be able to see our flight details in real time as a user
- Change or reschedule a flight as a member of staff
- Quickly check for any flight
- Be able to identify which routes have more or less traffic
- Be able to identify ways to make flight operations more efficient

## 2 Database Requirements:

Also showing foreign keys to represent certain relationships

### 1. Entities:

Strong Entities:

<attribute\_name> <data\_type> <constraints (if any)>

- Airport      (**Ap\_name** string PRIMARY KEY,  
                  **City** string NOT NULL,

**Country** string NOT NULL

)

- Airline (**Airline\_name** string UNIQUE NOT NULL,  
**Airline\_ID** string PRIMARY KEY)

(Assumption: Every airline has a unique Airline\_name)

- Flight\_Employee (**Emp\_id** string PRIMARY KEY,  
**Phone\_no** string multi-values NOT NULL,  
**EName** composite(**F\_name** string NOT NULL, **M\_name** string, **L\_name** string),  
**Jobtype** string NOT NULL,  
**Salary** float,  
**Bdate** DATE NOT NULL,  
**Age** int DERIVED as in Year(Bdate)-Year(CurrentDate)  
**Works\_For** string FOREIGN KEY referencing  
Airline.Airline\_ID  
**ManagerID** string FOREIGN KEY referencing  
Flight\_Employee.Emp\_id  
)

Here Phone\_no is a multi-valued attribute, Age is a derived attribute and Ename is a composite attribute consisting of F\_name, M\_name and L\_name. Manager\_id is a foreign key for the **Manager** relation

- Airplane (

**AirplaneID** string PRIMARY KEY,

**Aircraft\_Model** string NOT NULL,

**Flying\_Hours** float,

**SeatingCapacity** int NOT NULL

)

- Passenger (**PID** string PRIMARY KEY,

**Phone\_no** string multi-values NOT NULL,

**PName** composite(**F\_name** string NOT NULL, **M\_name** string,  
**L\_name** string),

**Bdate** DATE NOT NULL,

**Age** int DERIVED as in Year(Bdate)-Year(CurrentDate)

BoardingFlight

)

Here Phone\_no is a multi-valued attribute and PName is composite attribute consisting of F\_name, M\_name and L\_name.

Sub-classes of Passenger: First\_Class\_Passenger,  
Business\_Class\_Passenger, Student\_Passenger.

The subclassing relationship is partial. All the subclasses inherit the attributes of Passenger. The additional attributes are given below

First\_Class\_Passenger (

**FirstClassServices** string,

**PrivateSuite** Boolean NOT NULL,

**SleepPodAvailability** Boolean NOT NULL

)

Business\_Class\_Passenger (

**BusinessClassServices** string;

**BusinessLoungeAccess** boolean NOT NULL;

**PriorityBoarding** boolean NOT NULL

**WorkstationAvailability** boolean NOT NULL

Weak Entities:

- Dependents (

**Name** string NOT NULL,

**DOB** Date NOT NULL,

**Relationship** string,

**Depends\_on** string FOREIGN KEY referencing  
Flight\_Employee.Emp\_id

)

Here Name is a partial key and we are assuming that every  
Flight\_Employee has a dependent.

- Flight (

**Flight\_code** string NOT NULL,

**Source** string FOREIGN KEY referencing Airport.Ap\_name,

**Destination** string FOREIGN KEY referencing Airport.Ap\_name,

**AirplaneID** FOREIGN KEY referencing Airplane.AirplaneID NOT  
NULL,

**AirlineID** FOREIGN KEY referencing Airline.AirlineID NOT NULL,  
**Pilot** FOREIGN KEY referencing Flight\_Employee.Emp\_id NOT NULL,  
**Arrival** TIMESTAMP NOT NULL,  
**Departure** TIMESTAMP NOT NULL,  
**Duration** DERIVED as in Arrival-Departure  
 )

This table is for Flight Journeys.

Here Duration is a derived attribute.

- Baggage (
 

**Baggageld** char[2] NOT NULL,  
**Weight** float NOT NULL,  
**BelongsTo** string FOREIGN KEY referencing Passenger.PID  
 )

Baggageld is a partial key.

Here the Baggagelds are not unique but when combined with the PID of the passengers the rows are unique.

Relationships along with the min-max constraints:

- **Worksfor**: relationship between Flight\_Employee(1,1) and Airline(1, N).

Complete participation of Flight\_Employee and Airline.

- **Operates:** relationship between Airport(1,N) and Airline(1,M)

Complete participation of Airport and Airline.

Cannot be implemented using a foreign key. Needs it's own table as it is a many to many relation.

- **Luggage:** relationship between Baggage(0,N) and Passenger(1,1)

Complete participation of Baggage and partial participation of passenger.

- **Depends on:** relationship between Dependents(1,1) and Flight\_Employee(0,N)

Complete participation of Dependents.

- **Flight\_details:** defining relationship for the weak entity Flight.

Relation between Flight(1,1), Airplane(0,P), Flight\_Employee(0,R),Airport(0,Q)

Complete participation of Flight.

- **BoardingFlight:** relationship between Flight(0,N) and Passengers(1,1)
- **Manager:** relationship between Flight\_Employee (in the role of manager) (0,N) and Flight\_Employee (in the role of subordinate) (0,1)

Cardinality: (0,N):(0,1)

**Assumption - An employee has at most one manager.**

## Functional Requirements:

### 1.Retrieval:

#### 1. Selection Query:

Retrieve complete data tuples of all employees working at a specific airport

```
SELECT * FROM Flight_Employee
```

```
WHERE WorksFor = SELECT Airline_ID  
FROM Airlines  
WHERE Airline_name = "Rajiv Gandhi International Airport";
```

## **2. Projection Query:**

Retrieve names of all airlines operating at a specific airport

```
SELECT Airline_ID  
FROM Airline  
WHERE AP_Name = 'valmiki international airport'
```

## **3. Aggregate Function:**

Calculate the average salary of all employees at a specific airport.

```
SELECT AVG(Salary)  
FROM Employee  
WHERE WorksFor = SELECT Airline_ID  
FROM Airlines  
WHERE Airline_name = "Rajiv Gandhi International Airport";
```

We can also do this via Joins

```
SELECT AVG(Salary)
```

FROM Employee

WHERE WorksFor = SELECT Airline\_ID

FROM Airlines

WHERE Airline\_name = "Rajiv Gandhi International Airport";

#### **4. Search Query:**

Find all passengers with names containing "VED MAURYA" ;

SELECT \* FROM Passenger WHERE PName LIKE 'VED MAURYA';

#### **5. Analysis Reports:**

1. Number of flights operated by each airline at a specific airport.

SELECT Airline.Airline\_name, COUNT(Flight.Flight\_Code)

FROM Flight

JOIN Airline ON Flight.Airline\_ID = Airline.Airline\_ID

WHERE Flight.AP\_Name = 'Specific Airport Name'

GROUP BY Airline.Attribute;

2. Total baggage weight for each passenger on a specific flight.

SELECT Passenger.PName, SUM(Baggage.Weight)

FROM Baggage

JOIN Passenger ON Baggage.PID = Passenger.PID



WHERE Flight\_Code = 'Specific Flight Code'

GROUP BY Passenger.PName;

## 2.Modification:

### 1.INSERT

Add a new flight ensuring no violation of integrity constraints.

INSERT INTO Flight (123, 2023101006, "Rajiv gandhi international airport", "valmiki international airport" , 2023-10-23 2:00 AM, 2023-10-23 4:00 AM) ;

If there is an integrity violation - like a primary key entry is NULL, or duplicate or if a foreign pointer does not exist, the DB will reject this transaction.

### 2.UPDATE

Update the salary of a specific employee

UPDATE Flight\_Employee SET Salary = 10000 WHERE Emp\_id = 12000;

If the update statement causes entity/domain integrity or key constraint violation, then the transaction will not be performed.

For a foreign key, we can specify the update to be cascaded or instead be set to NULL in case updation of parent attribute causes referential integrity violations

### 3.Deletion:

Delete a specific baggage record

DELETE FROM Baggage WHERE Baggage\_ID = "3678";

Delete an Airline that has gone bankrupt.

DELETE FROM Airline WHERE Airline\_name="Go First";

When DELETE causes a referential integrity constraint, we can tell the Database to either reject the operation, set the referencing variables to NULL or cascading delete them (dangerous) while specifying the foreign key.