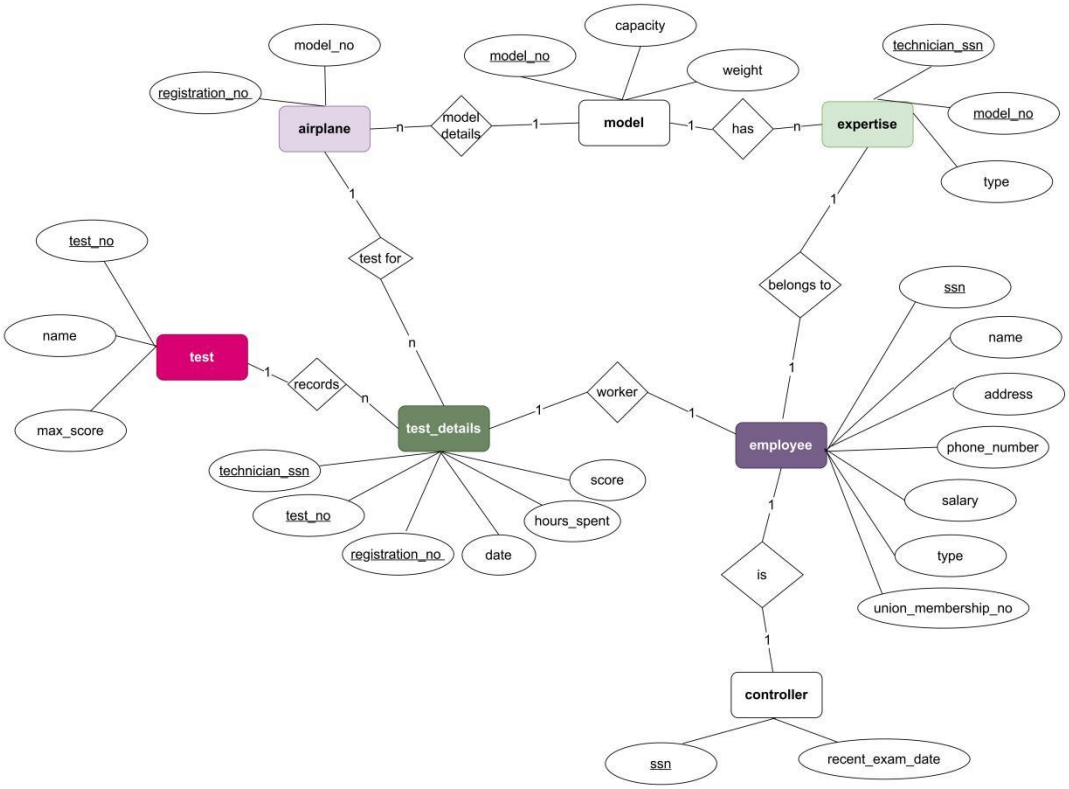
Airport Information System

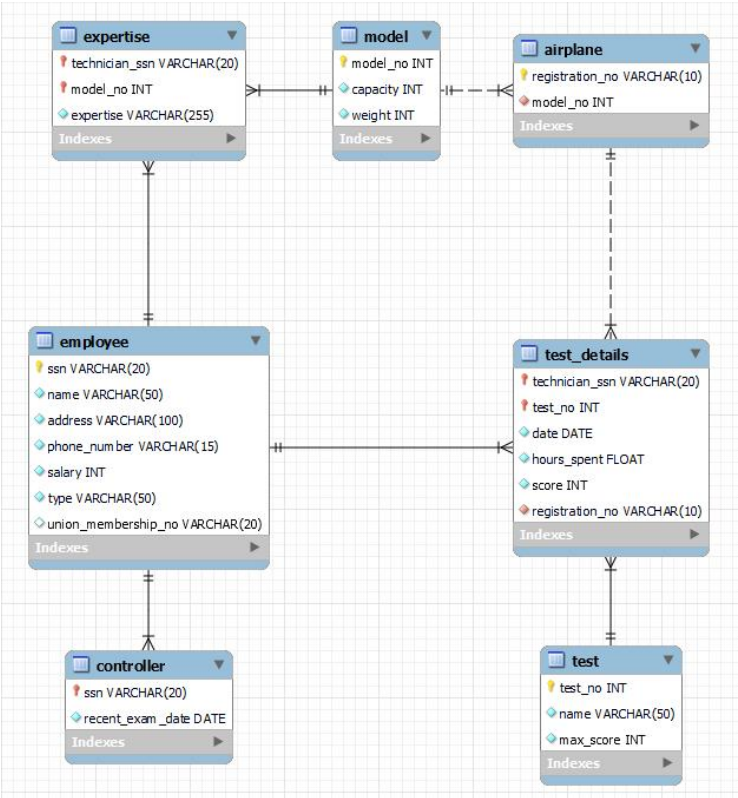
Contents

1. Introduction

The purpose of this project is to develop and implement a relational database for an airport information system. The database will store information about airplanes, models, technicians, traffic controllers, union membership, and tests. This report outlines the entities and relations involved in the database design, and the reasoning behind the choices made in terms of the models and normalization decisions.

Enhanced Entity-Relationship (EER) diagram





Entities and Relations: The database will consist of the following entities and relations:

1. airplane: registration_no (PK), model_no (FK)
2. model: model_no (PK), capacity, weight
3. expertise: technician_ssn (PK, FK), model_no (PK, FK), type
4. controller: ssn (PK,FK), recent_exam_date
5. employee: ssn (PK), name, address, phone_number, salary, type, union_membership_no
6. test: test_no (PK), name, max_score
7. test_details: technician_ssn (FK), test_no (FK), registration_no (FK), date, hours_spent, score

2. Reasoning:

Airplane and Model: An airplane is identified by its registration number, and each airplane is of a specific model. Therefore, airplane and model entities are created, where airplane has a primary key of registration number and a foreign key of model number. The model entity has a primary key of model number and contains information about capacity and weight.
Expertise: A technician is an expert on one or more plane models, and their expertise may overlap with that of other technicians. Therefore, an expertise entity is created with a composite primary key of technician SSN and model number, and an additional attribute of type to indicate the level of expertise.
Controller: Traffic controllers must have an annual medical examination, so a controller entity is created with a primary key of SSN and an attribute of recent exam date to store the most recent exam date.
Employee: All airport employees belong to a union, so an employee entity is created with a primary key of SSN and additional attributes of name, address, phone number, salary, type, and union membership number.
Test and Test Details: The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score. The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a given test. Therefore, a test entity is created with a primary key of test number and additional attributes of name and maximum score. A test details entity is also created with foreign keys of technician SSN, test number, and airplane registration number, and additional attributes of date, hours spent, and score.

Normalization: Normalization is the process of organizing a database to reduce redundancy and dependency, and to improve data integrity. The database design is normalized to third normal form (3NF).

First Normal Form (1NF): Each attribute of a relation must have atomic values. The entities created in this database design are already in 1NF since each attribute has a single value.

Second Normal Form (2NF): Every non-key attribute must be functionally dependent on the primary key. The entities created in this database design are in 2NF since each non-key attribute is functionally dependent on the primary key.

Third Normal Form (3NF): Every non-key attribute must be functionally dependent on the primary key, and no non-key attribute should be dependent on another non-key attribute. The entities created in this database design are in 3NF since there are no transitive dependencies between attributes.

3. Create table script:

```
CREATE TABLE model (
   model_no INT PRIMARY KEY,
   capacity INT NOT NULL,
   weight INT NOT NULL
);

CREATE TABLE airplane (
   registration_no VARCHAR(10) PRIMARY KEY,
   model_no INT NOT NULL,
   FOREIGN KEY (model_no) REFERENCES model(model_no)
);

CREATE TABLE expertise (
   technician_ssn VARCHAR(20) NOT NULL,
   model_no INT NOT NULL,
   expertise VARCHAR(255) NOT NULL,
   PRIMARY KEY (technician_ssn, model_no),
   FOREIGN KEY (technician_ssn) REFERENCES employee(ssn),
   FOREIGN KEY (model_no) REFERENCES model(model_no)
);

CREATE TABLE controller (
   ssn VARCHAR(20) PRIMARY KEY,
   recent_exam_date DATE NOT NULL,
      FOREIGN KEY (ssn) REFERENCES employee(ssn)
);

CREATE TABLE employee (
   ssn VARCHAR(20) PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   address VARCHAR(100) NOT NULL,
   phone_number VARCHAR(15) NOT NULL,
   salary INT NOT NULL,
      type VARCHAR(50) NOT NULL,
   union_membership_no VARCHAR(20) DEFAULT NULL
);

CREATE TABLE test (
   test_no INT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   max_score INT NOT NULL
);
```

```
CREATE TABLE test_details (
   technician_ssn VARCHAR(20) NOT NULL,
   test_no INT NOT NULL,
   date DATE NOT NULL,
   hours_spent FLOAT NOT NULL,
   score INT NOT NULL,
   registration_no VARCHAR(10) NOT NULL,
   PRIMARY KEY (technician_ssn, test_no),
   FOREIGN KEY (technician_ssn) REFERENCES employee(ssn),
   FOREIGN KEY (test_no) REFERENCES test(test_no),
      FOREIGN KEY (registration_no) REFERENCES airplane(registration_no)
);
```

Results:

Insert datas script:

```
-- Insert data for the model table
INSERT INTO model (model_no, capacity, weight)
VALUES
      (1, 200, 5000),
      (2, 300, 7500),
      (3, 400, 10000),
      (4, 250, 6000),
      (5, 350, 8000);

-- Insert data for the airplane table
INSERT INTO airplane (registration_no, model_no)
VALUES
      ('ABC123', 1),
      ('DEF456', 2),
      ('GHI789', 3),
      ('JKL012', 1),
      ('MNO345', 2);

-- Insert data for the employee table
INSERT INTO employee (ssn, name, address, phone_number, salary, type,
union_membership_no)
VALUES
      ('111-11-1111', 'John Smith', '123 Main St', '555-1234', 60000, 'technician', 100),
      ('222-22-2222', 'Jane Doe', '456 Oak Ave', '555-5678', 70000, 'technician', 200),
      ('333-33-3333', 'Bob Johnson', '789 Maple St', '555-9012', 80000, 'technician', 300),
      ('444-44-4444', 'Sarah Lee', '321 Elm St', '555-3456', 90000, 'technician', 400),
      ('555-55-5555', 'Tom Smith', '654 Birch Ln', '555-7890', 100000, 'technician', 500),
```

```
        ('666-66-6666', 'Allen', '120 Birch Ln', '555-7982', 100500, 'controller', 600),
        ('777-77-7777', 'Hiton', '119 Birch Ln', '555-8413', 89000, 'controller', 700);

-- Insert data for the expertise table
INSERT INTO expertise (technician_ssn, model_no, expertise)
VALUES
    ('111-11-1111', 1, 'e1'),
    ('111-11-1111', 2, 'e1'),
    ('222-22-2222', 2, 'e2'),
    ('333-33-3333', 3, 'e3'),
    ('555-55-5555', 4, 'e3'),
        ('444-44-4444', 5, 'e5');

-- Insert data for the controller table
INSERT INTO controller (ssn, recent_exam_date)
VALUES
        ('666-66-6666', '2022-04-15'),
    ('777-77-7777', '2022-09-17');

-- Insert data for the test table
INSERT INTO test (test_no, name, max_score)
VALUES
    (1, 'Flight test', 100),
    (2, 'Maintenance test', 90),
    (3, 'Emergency procedures test', 80),
        (4, 'Navigation test', 95),
        (5, 'Communication test', 85);

-- Insert data for the test_details table
INSERT INTO test_details (technician_ssn, test_no, date, hours_spent, score, registration_no)
VALUES
('111-11-1111', 1, '2022-01-01', 2.5, 90, 'ABC123'),
('222-22-2222', 1, '2022-01-01', 2.5, 100, 'DEF456'),
('111-11-1111', 2, '2022-02-15', 3.0, 85, 'DEF456'),
('222-22-2222', 2, '2022-03-01', 2.5, 90, 'DEF456'),
('333-33-3333', 3, '2022-04-01', 4.0, 80, 'GHI789'),
('555-55-5555', 4, '2022-05-15', 3.5, 92, 'JKL012'),
('111-11-1111', 4, '2022-05-15', 3.5, 95, 'JKL012'),
('444-44-4444', 5, '2022-06-02', 3.6, 85, 'MNO345');
```

4. Query application:

```
# Import the necessary libraries
import mysql.connector

# Connect to the database
conn = mysql.connector.connect(host='square.law.miami.edu', user='zxg287', password='~Csc7887', database='zxg287_FINAL_PROJECT')
c = conn.cursor()

# Define the menu options
while True:
    print("\nMenu Options:")
    print("1. Insert a new technician into the database.")
    print("2. Delete an existing airplane from the database.")
    print("3. Update the expertise of an existing technician.")
    print("4. List the details of the technician whose salary is greater than the average of the salary of all technicians.")
    print("5. List all the model numbers that a given technician has the expertise, along with their capacity and weight information.")
    print("6. List the total number of technicians who are experts in each model.")
    print("7. List the details (test number, test name, maximum score, etc.) of the FAA tests for a given airplane, sorted by the maximum scores.")
    print("8. List the most recent annual medical examination and his/her union membership number for each traffic controller.")
    print("9. List the total number of tests done by each technician for a given airplane.")
    print("10. List the name of the technician, the registration number of the airplane, and the FAA number of those tests done between September 2005 and December 2005, sorted by the FAA numbers.")
    print("0. Exit")

    # Get the user's input
    choice = int(input("Enter your choice: "))

    # Execute the corresponding menu option
    if choice == 1:
        # Insert a new technician into the database
        ssn = input("Enter the technician's ssn: ")
        name = input("Enter the technician's name: ")
        address = input("Enter the technician's address: ")
        phone = input("Enter the technician's phone: ")
        salary = float(input("Enter the technician's salary: "))
        model_no = int(input("Enter the work for model_no: "))
        expertise = input("Enter the technician's expertise: ")
```

```
            union_membership_no = int(input("Enter the union_membership_no: "))

            c.execute("INSERT INTO employee VALUES ('"+ssn+"', '"+name+"', '"+address+"', '"+phone+"', "+str(salary)+", 'technician', "+str(union_membership_no)+")")
            c.execute("INSERT INTO expertise VALUES ('"+ssn+"', "+str(model_no)+", '"+expertise+"')")
            conn.commit()
            print("Technician added successfully.")

        elif choice == 2:
            # Delete an existing airplane from the database
            registration_number = input("Enter the registration number of the airplane you want to delete: ")
            c.execute("DELETE FROM airplane WHERE registration_no='"+registration_number+"'")
            conn.commit()
            print("Airplane deleted successfully.")

        elif choice == 3:
            # Update the expertise of an existing technician
            ssn = input("Enter the ssn of the technician: ")
            expertise = input("Enter the new expertise of the technician: ")
            model_no = input("Enter the work for model_no: ")
            c.execute("UPDATE expertise SET expertise='"+expertise+"' WHERE technician_ssn='"+ssn+"' and model_no="+model_no)
            conn.commit()
            print("Expertise updated successfully.")

        elif choice == 4:
            # List the details of the technician whose salary is greater than the average of the salary of all technicians
            c.execute("SELECT * FROM employee WHERE  type='technician' and salary >(SELECT AVG(salary) FROM employee WHERE  type='technician')")
            rows = c.fetchall()
            for row in rows:
                print(row)

        elif choice == 5:
            # List all the model numbers that a given technician has the expertise, along with their capacity and weight information
            ssn = input("Enter the name of the ssn: ")
            c.execute("SELECT model_no, capacity, weight FROM model WHERE model_no IN (SELECT model_no FROM expertise WHERE technician_ssn='"+ssn+"')")
            rows = c.fetchall()
            for row in rows:
```

```
            print(row)

    elif choice == 6:
        # List the total number of technicians who are experts in each model
        c.execute("SELECT model_no, COUNT(*) FROM expertise GROUP BY model_no")
        rows = c.fetchall()
        for row in rows:
            print(row)

    elif choice == 7:
    # List the details (test number, test name, maximum score, etc.) of the FAA tests for a given airplane, sorted by the maximum scores
    registration_number = input("Enter the registration number of the airplane: ")
    c.execute("SELECT t.test_no, t.`name`, a.maxscore FROM (SELECT d.test_no, MAX(d.score) AS maxscore FROM test_details d WHERE d.registration_no = '"+registration_number+"' GROUP BY d.test_no) a LEFT JOIN test t ON t.test_no = a.test_no ORDER BY a.maxscore")
    rows = c.fetchall()
    for row in rows:
        print(row)      elif choice == 8:
        # List the most recent annual medical examination and his/her union membership number for each traffic controller
        c.execute(
            "SELECT c.*,e.union_membership_no FROM `controller` c LEFT JOIN employee e on c.ssn = e.ssn")
        rows = c.fetchall()
        for row in rows:
            print(row)

    elif choice == 9:
        # List the total number of tests done by each technician for a given airplane
        registration_number = input("Enter the registration number of the airplane: ")
        c.execute(
            "SELECT d.technician_ssn, count(d.date) FROM test_details d where d.registration_no='"+registration_number+"' GROUP BY d.technician_ssn")
        rows = c.fetchall()
        for row in rows:
            print(row)

    elif choice == 10:
        # List the name of the technician, the registration number of the airplane, and the FAA number of those tests done between September 2005 and December 2005, sorted by the FAA numbers
        c.execute(
            "SELECT e.`name`, t.registration_no, t.test_no FROM test_details t LEFT JOIN employee e on t.technician_ssn = e.ssn WHERE t.date BETWEEN '2005-09-01' AND '2005-12-31'
```

```
ORDER BY t.test_no")
            rows = c.fetchall()
            for row in rows:
                print(row)

        elif choice == 0:
            # Exit the program
            conn.close()
            print("Goodbye!")
            break

        else:
            print("Invalid choice. Please try again.")
```

5. Conclusion

In conclusion, this report outlines the entities and relationships involved in the design of a relational database for an airport information system. The database will store information about airplanes, airplane models, technicians, air traffic controllers, union memberships, tests, and test details. Each of these entities will have its own table in the database, with relationships between them established using foreign keys.

Normalization is a crucial consideration when designing a relational database. The goal of normalization is to reduce data redundancy and enhance data integrity. In our design, we have adhered to the rules of normalization up to the third normal form (3NF). In the first normal form (1NF), we have ensured that each attribute in every table contains only atomic values, thereby avoiding the storage of multiple values in a single attribute.

In the second normal form (2NF), we have ensured that each non-key attribute in every table is functionally dependent on the entire primary key. This means that we have avoided storing non-key attributes that are only dependent on part of the primary key. In the third normal form (3NF), we have ensured that each non-key attribute in every table is not transitively dependent on the primary key. This means that we have avoided storing non-key attributes that are dependent on other non-key attributes.

Overall, the design of this airport information system database is well-organized, adheres to good database design principles, and is effectively normalized. By implementing this design, the airport can efficiently and effectively manage information about airplanes, technicians, air traffic controllers, union memberships, tests, and test details, enabling them to make informed decisions about the maintenance and safety of their operations.