Project: Design, Development, and Implementation of a Relational Database

Airport Information System

May 2nd 2016

ECE567/CSC423 – Database Design & Management Spring 2016

Saad Sadiq – C11954772

Table of Contents

Objective

Develop an airport information system to organize all information about all the airplanes stationed and maintained at the airport in the XYZ county. You need to specify the assumptions that are used in your design, if there is any. The relevant information is as follows:

• Every airplane has a registration number, and each airplane is of a specific model.

• The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.

• A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.

• Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.

• Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.

• All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by a social security number.

• 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. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score the airplane received on the test.

Introduction

In this project we have been assigned the task to develop an airport information system to organize all information about all the airplanes stationed and maintained at the airport in the XYZ county. We need to develop both the Conceptual and logical diagrams while following the necessary design steps. Finally we implement the designed database in the ORACLE database using SQL and run 10 test case queries using Embedded SQL. As a first step it is appropriate to specify the assumptions that are used in our design

Following are the constraints and assumptions

- We assume that there is only one union and each employee is member of that union. Therefore the union name is not worth mentioning. If there are more than one unions, then it would be necessary to create another entity called 'Union' that will hold the different unionNames and their disctinctive Ids.
- There can be several airplanes of the same model, for example, there can be several Boeing 737 airplanes with different registration numbers.
- Each technician has an expertise of at least one airplane.
- A traffic controller cannot not also work as a technician, or vice versa.
- Each airworthiness test is performed by one technician only on one aircraft only using one test type only.
- If there are many technicians, there may be some technicians who have never conducted any test.
- There is at least one airplane of each model type. Otherwise why would the model be listed in the models table
- The FAA tests have a general list and it is not mandatory that each test have at least one plane linked to it. There may be tests that are not required in the region.
- Every airplane has a Boolean attribute called 'airworthiness'. If the airplane fails its test, then the FAA can request the airplane's airworthiness to be changed to False so it is grounded.
- Each test must be conducted by a technician who is an expert on that model.
    - For a technician to be “conducting” a test, and for a technician to “be expert” for a given test and model are two distinct relationships. We have no ER constraints that establishes such value-based relationships among 2 distinct relationships of “is-expert” versus “conducts-tests” and verifying that the matching technician entity exists in both cases

1. Conceptual ER Diagram

The objective at this stage i.e. the conceptual design stage is to present information about the database model that suitably provides information about the airplanes stationed and maintained at the airport into a database management system to improve the organization at the airport. To accomplish that the top down database design methodology have been used for designing the conceptual ERD as the basis of this database project. The description of necessary entities and constraints have been given in the conceptual diagram whereas the details of the attributes will be presented in the logical design stage.

Identification of the relations (entity types)

Entity Name	Description	Occurrence
Airplane	Every airplane has a registration number, and each airplane is of a specific model	Each airplane will be of one specific model
AirplaneModel	There are a number of airplane models, and each model is identified by a model number and has a capacity and a weight	There can be several airplanes of the same model
Employee	General item describing all staff employed by the airport	Each member of the staff works at one particular branch
Technician	A number of technicians work at the airport. Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians	
TrafficController	There is another type of employee at the airport called the traffic controllers.	
TestType	The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy.	Each property as a single owner and is available at one specific brach, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.
TestsConducted	The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a given test	

Identify relationship types and their participation and cardinality constraints.

Entity Name	Multiplicity	Relationship	Multiplicity	Entity name
Airplane	0..* 0..*	Has types takes part in	1..1 1..*	**Model** **TestsConducted**
AirplaneModel	1..1	Belongs to	0..*	**Airplane**
Employee	1..1 1..1	Has type Has type	0..1 0..1	**Technician** **TrafficController**
Technician	0..1 0..*	Belongs to holds conducts	1..1 1..*	**Employee** **TechnicianExpertise** **TestConducted**
TechnicianExpertise	1..*	Includes	0..*	**Technician**
TrafficController	0..1	Belongs to	1..1	**Employee**
TestType	1..1	Used by	0..*	**TestsConducted**
TestConducted	0..* 1..* 0..*	Conducted by performed on by following	1..1 1..1 1..1	**Technician** **Airplane** **TestType**

Identification of attributes and association of attributes with entity or relationship types.

Entity Name	Attributes	Description	Data Type & Length	Nulls	Multi-valued
Airplane	regNo airworthy	Uniquely identifies every airplane Boolean status of the airplane	Integer 1 character Y or N	No No	No No
AirplaneModel	modelNo capacity weight	Uniquely identifies every model shows seating capacity weight of each plane	Integer Integer Integer	No	No No No
Employee	ssn unionMemNo	Uniquely identifies a member of staff identifies the union membership	Integer Integer	No No	no No
Technician	name salary phone address	Name of the technician monthly salary	30 variable characters 10 digit number 10 digit number 50 variable characters	No No No	No No No No
TechnicianExpertise	comments yearsSinceExpert	Recommendations etc year when he became an expert	100 variable char 4 digit number	No	No No
TrafficController	examDate medExamScore	Date on which exam was taken score he got on the medical test	Date integer	No	No No
TestType	FAANo name maxScore	Uniquely identify each FAA test unique name of the test maximum allowed score of each	Integer 25 variable characters integer	No No	No No No
TestConducted	hours score testDate	Number of hours test was performed score achieved on the test date of the test	Integer Integer date	No	No No No

Determination of candidate and primary key attributes of entity types.

Employee

Primary key: ssn
Candidate Key: ssn, unionMemNo

Technician

Primary key: ssn (FK Reference to Employee)
Candidate key: ssn, phone
Other attributes: name, salary, address

TechnicianExpertise

Primary key: ssn (FK Ref. To Technician), modelNo (FK Ref. To AirplaneModel)
Candidate key: ssn, modelNo
Other attributes: comments, yearSinceExpert

TrafficController

Pirmary key: ssn (FK Ref. To Employee), examDate
Candidate key: ssn, examDate
Other attributes: medExamScore

Airplanes

Primary key: regNo
Candidate key: regNo
Other attributes: airworthy, modelNo (FK Ref. Airplane Model)

AirplaneModel

Primary key: modelNo
Candidate key: modelNo
Other attributes: capacity, weight

TestType

Primary key: FAANo
Candidate key: FAANo, name
Other attributes: maxScore

TestConducted

Primary key: regNo(FK Ref. Airplane), modelNo,ssn (FK Ref. TechnicianExpertise), testNo(FK Ref. TestType)
Candidate key: regNo, modelNo, ssn, testNo
Other attributes: score, testDate, hours

Determination of specialization/generalization and categorization relationships, whenever it is appropriate.

In this section we perform the generalization over Employee, Technician and TrafficController relations. The Technician and TrafficController entities are generalized to the Employee relation. We apply here generalization because from the problem statement we knew that there are only two types of employees, and generalization implies mandatory participation in one of the subclasses, therefore it was a good fit. In this relation, Employee is the superclass while Technician and TrafficController are the subclasses. Participation is required in atleast one of the subclasses. i.e. the employee has to belong in either Technician or TrafficController thus participation is mandatory. For cardinality constraints, it will be a definite disjoint constraint. There will be no overlap between technicians and traffic-controllers. An employee will either be a technician or a traffic-controller but not both.

Enhanced Entity-Relationship (EER) diagram to reflect the requirements

2. Develop a logical data model based on the following requirements:

Refinement of the conceptual model - including a refined Enhanced Entity-Relationship (EER) diagram.

We create relations for the logical data model to represent the entities, relationships, and attributes that have been identified. Uptill this stage the attributes were not added to the EER diagram, we will add them now.

Derivation of relations from the refined conceptual model.

In this step, we derive relations for the logical data model to represent the entities, relationships, and attributes. We describe the composition of each relation using a Database Definition Language (DBDL) for relational databases. Using the DBDL, we first specify the name of the relation followed by a list of the relation's simple attributes enclosed in brackets. We then identify the primary key and any alternate and/or foreign key(s) of the relation. In deciding where to place the foreign key we must first identify the 'parent' and 'child' entities in the relationship. The parent entity refers to the entity that posts a copy of its primary key into the child entity to act as a foreign key.

(1) Strong entity types

List of the entities for which we can assert the primary keys: Airplane, TestType, Employee, AirplaneModel

(2) weak entity types

List of the entities for which we readily dont have the primary key:
TestCondcuted, TechnicianExpertise, TrafficController, Technician

(3) One-to-many relationships

For 1..* relationships, we put '1' i.e. the parent entiry primary key into the '*' child table as a FK. Relations include AirplaneModel & Airplane

(4) One-to-One binary relationships

(a) With mandatory participation on both sides:

In this case we combine the entities involved into one relation. If there is a primary key in the other table, it is used as an alternative key. But we dont have any such case

(b) mandatory participation on one side:

The entity that has optional participation in the relationship. A copy of the parent entity primary key is placed in the child entity as a foreign key and serves also as a primary key. But dont have such case.

(c) Optional participation on both sides

In this case we arbitrarily choose any one as the parent, but we dont have any such case here.

(5) **One-to-One recursive**

We dont have any recursive relationships

(6) Super class / subclass relationship types

For the Employee, Technician and TrafficController entities we have the superclass / subclass relationship. Here we have a case of 'Many relations' i.e. Mandatory participation and disjoint cardinality. Therefore we include a Primary key copy of the superclass in each subclass.

(7) many-to-many binary relationships

We post a copy of the PK of the entities that participate in the relationship into the new relation to act as foreign keys. One or both of these foreign keys will also form the primary key of the new relation. (case of technical expertise). We have the case of TechnicianExpertise relation. We will copy the Primary key of Technician and AirplaneModel as foreign keys and also make them the primary key of the TechnicianExpertise relation.

(8) **Complex relationship types**

We Create a relation to represent the complex relationship and include the attributes that are part of the relationship. We post a copy of the primary key of the entities that participate in the complex relationship into the new relation to act as foreign keys. Any foreign key that represents a many relationship (1..*, 0..*) will also form the primary key of this new relation. (tests conducted case). We have the relation 'TestConducted' in which the TestType, Airplane and Technician entities take part. We will copy the primary keys of these three entities in the relation and also make them primary key of the TestConducted relation.

(9) Multi-valued attributes

We haven included any multi-valued attribute in our case study.

Airplane (regNo, airworthy) **Primary key** regNo **Foreign Key** modelNo **references** AirplaneModel(modelNo)	**AirplaneModel** (modelNo, capacity, weight) **Primary key** modelNo
Employee (ssn, unionMemNo) **Primary key** ssn **Alternate Key** unionMemNo	**TrafficController** (ssn, examDate, medExamScore) **Primary key** ssn, examDate **Foreign Key** ssn **references** Employee(ssn)
Technician (ssn, name, salary, phone, address) **Primary key** ssn **Alternate key** phone **Foreign Key** ssn **references** Employee(ssn)	**TechnicianExpertise** (ssn, modelNo, yearSinceExpert, comments) **Primary key** ssn, modelNo **Foreign Key** ssn **references** Technician(ssn) **Foriegn Key** modelNo **references** AirplaneModel(modelNo)
TestType (FAANo, name, maxscore) **Primary key** FAANo **Alternate Key** name	**TestConducted** (regNo, testNo, ssn, score, testDate, hours) **Primary key** regNo, testNo, ssn **Foreign Key** regNo **references** Airplane(regNo) **Foreign Key** testNo **references** TestType(FAANo) **Foreign Key** ssn **references** Technician(ssn)

Validation of logical model using normalization to BCNF.

UNF to 1NF

Each intersection of rows and columns in the EER diagram contains only one value. Thus the database is 1 NF pass.

1NF to 2NF

In every relation, every non-primary-key attribute is fully functionally dependent on the candidate key and there are no partial dependencies. Thus the database is 2NF pass.

2NF to 3NF

In every relation, no non-primary-key attribute is transitively dependent on its primary key i.e. all non-key attributes are fully dependent on the primary key. Thus the database is 3NF pass.

3NF to BCNF

In BCNF, every determinant is a candidate key. For a dependency to stay in a relation, it must be a candidate key. In our case study, every determinant is a candidate key, thus the table is also in Boyce-Codd Normal Form.

Our EER diagram in Figure 3 remains same after the normalization process.

Definition of integrity constraints including

» Required data
» Attribute domain constraints
» Multiplicity
» Entity integrity
» Referential integrity
» General constraints

Airplane (regNo, airworthy) **Primary key** regNo NOT NULL **Other attribute:** airworthy NOT NULL CHECK (VALUE IN ('Y','N')) **Foreign Key** modelNo **references** AirplaneModel(modelNo) ON DELETE CASCADE
AirplaneModel (modelNo, capacity, weight) **Primary key** modelNo NOT NULL
Employee (ssn, unionMemNo) **Primary key** ssn NOT NULL **Alternate Key** unionMemNo NOT NULL
Technician (ssn, name, salary, phone, address) **Primary key** ssn NOT NULL **Alternate key** phone CHECK (phone >0) **Other Attributes** salary CHECK (salary >0) **Foreign Key** ssn **references** Employee(ssn) ON DELETE CASCADE
TestType (FAANo, name, maxscore) **Primary key** FAANo NOT NULL **Alternate Key** name NOT NULL DEFAULT 'NA'
TrafficController (ssn, examDate, medExamScore) **Primary key** examDate, ssn NOT NULL **Foreign Key** ssn **references** Employee(ssn) ON DELETE CASCADE
TechnicianExpertise (ssn, modelNo, yearSinceExpert, comments) **Primary key** ssn, modelNo NOT NULL **Foreign Key** ssn **references** Technician(ssn) ON DELETE SET NULL **Foriegn Key** modelNo **references** AirplaneModel(modelNo) ON DELETE CASCADE
TestConducted (regNo, testNo, ssn, score, testDate, hours) **Primary key** regNo, testNo, ssn NOT NULL **Other Attributes** score CHECK (score < maxScore) **Foreign Key** regNo **references** Airplane(regNo) ON DELETE CASCADE **Foreign Key** FAANo **references** TestType(FAANo) ON DELETE CASCADE **Foreign Key** ssn **references** Technician(ssn) ON DELETE SET NULL

3. Translate the logical data model for the ORACLE Enterprise DBMS

Development of SQL code to create the entire database schema and reflect its constraints

```
SQL> CREATE TABLE AirplaneModel (
  2  modelNo varchar(6) NOT NULL,
  3  capacity int,
  4  weight int,
  5  PRIMARY KEY (modelNo)
  6  );
```

```
SQL> CREATE TABLE Airplane (
  2  regNo int NOT NULL,
  3  modelNo varchar(6) NOT NULL,
  4  airworthy CHAR NOT NULL CHECK (airworthy IN ('Y','N')),
  5  PRIMARY KEY (regNo),
  6  FOREIGN KEY (modelNo) REFERENCES AirplaneModel(modelNo) ON DELETE CASCADE
  7  );
```

```
SQL> CREATE TABLE Employee (
  2  ssn int NOT NULL,
  3  unionMemNo int NOT NULL,
  4  PRIMARY KEY (ssn)
  5  );
```

```
SQL> CREATE TABLE TechnicianExpertise (
  2  ssn int NOT NULL,
  3  modelNo varchar(6) NOT NULL,
  4  yearSinceExpert number(4) NOT NULL CHECK(yearSinceExpert > 1980),
  5  comments varchar(100),
  6  PRIMARY KEY (ssn, modelNo),
  7  FOREIGN KEY (ssn) REFERENCES Technician(ssn) ON DELETE SET NULL,
  8  FOREIGN KEY (modelNo) REFERENCES AirplaneModel(modelNo) ON DELETE SET NULL
  9  );
```

```
SQL> CREATE TABLE TestType (
  2  FAANo int NOT NULL,
  3  name varchar(25) DEFAULT 'NA',
  4  maxScore int,
  5  PRIMARY KEY (FAANo)
  6  );
```

```
SQL> CREATE TABLE TrafficController (
  2  ssn int NOT NULL,
  3  examDate date NOT NULL,
  4  medExamScore int NOT NULL,
  5  PRIMARY KEY(ssn,examDate),
  6  FOREIGN KEY (ssn) REFERENCES Employee(ssn) ON DELETE CASCADE
  7  );
```

```
SQL> CREATE TABLE Technician (
  2  ssn int NOT NULL,
  3  name varchar(30) NOT NULL,
  4  salary number(10) NOT NULL CHECK(salary >0),
  5  address varchar(50),
  6  phone number(10),
  7  PRIMARY KEY (ssn),
  8  FOREIGN KEY (ssn) REFERENCES Employee(ssn) ON DELETE CASCADE
  9  );
```

```
SQL> CREATE TABLE TestConducted (
  2  regNo int NOT NULL,
  3  testNo int NOT NULL,
  4  ssn int NOT NULL,
  5  testDate date,
  6  hours int,
  7  score int NOT NULL,
  8  PRIMARY KEY (regNo, testNo, ssn),
  9  FOREIGN KEY (regNo) REFERENCES Airplane(regNo) ON DELETE CASCADE,
 10  FOREIGN KEY (testNo) REFERENCES TestType(FAANo) ON DELETE CASCADE,
 11  FOREIGN KEY (ssn) REFERENCES Technician(ssn) ON DELETE SET NULL
 12  );
```

4. Interact with the database and enable the transaction requirements.

We create several sample tuples for each relation in our database.

AirplaneModel
INSERT INTO AirplaneModel VALUES('DC10',134,2000);
INSERT INTO AirplaneModel VALUES('B737',129,1900);
INSERT INTO AirplaneModel VALUES('A380',438,4000);
INSERT INTO AirplaneModel VALUES('B777',302,2550);
INSERT INTO AirplaneModel VALUES('A340',195,3600);
INSERT INTO AirplaneModel VALUES('DC9',100,6900);

Airplane
INSERT INTO Airplane VALUES(12151,'DC10','Y');
INSERT INTO Airplane VALUES(12152,'B737','Y');
INSERT INTO Airplane VALUES(12153,'B737','Y');
INSERT INTO Airplane VALUES(12154,'A380','Y');
INSERT INTO Airplane VALUES(12155,'A380','Y');
INSERT INTO Airplane VALUES(12156,'B777','Y');
INSERT INTO Airplane VALUES(12157,'B777','Y');

Employee
INSERT INTO Employee VALUES(551,1131);
INSERT INTO Employee VALUES(62,1132);
INSERT INTO Employee VALUES(743,1133);
INSERT INTO Employee VALUES(884,1134);
INSERT INTO Employee VALUES(95,1135);
INSERT INTO Employee VALUES(116,1136);
INSERT INTO Employee VALUES(617,1137);
INSERT INTO Employee VALUES(897,1138);
INSERT INTO Employee VALUES(1121,1139);

Technician
INSERT INTO Technician VALUES(551,'John Oliver',5000,'6512 Santona St. Apt 1',3052212521);
INSERT INTO Technician VALUES(62,'Marc Anthony',7000,'815 Gables St. Apt 2',3054512516);
INSERT INTO Technician VALUES(743,'Ann hathaway',9000,'11 McArthur St. Apt 5',3059856321);
INSERT INTO Technician VALUES(884,'John Smith',8000,'87 Coral St. Apt 20',3055484312);
INSERT INTO Technician VALUES(95,'Santa Barbara',7500,'12 Senza St. Apt 14',3058745621);

TrafficController
INSERT INTO TrafficController VALUES(116,'01-May-2016',95);
INSERT INTO TrafficController VALUES(617,'12-Feb-2016',91);
INSERT INTO TrafficController VALUES(897,'12-Jan-2015',95);
INSERT INTO TrafficController VALUES(897,'15-Jan-2015',99);
INSERT INTO TrafficController VALUES(1121,'18-Apr-2014',86);
INSERT INTO TrafficController VALUES(1121,'21-Apr-2014',88);

TestType
INSERT INTO TestType VALUES(119,'Flight deck test',500);
INSERT INTO TestType VALUES(120,'Flight path test',450);
INSERT INTO TestType VALUES(121,'Strobe light test',320);
INSERT INTO TestType VALUES(122,'Stability test',650);
INSERT INTO TestType VALUES(123,'Landing test',500);
INSERT INTO TestType VALUES(124,'Takeoff test',400);

TechnicianExpertise

```
INSERT INTO TechnicianExpertise VALUES(551,'DC10',1991,'Aced the test');
INSERT INTO TechnicianExpertise VALUES(62,'B737',2001,'Best in this class');
INSERT INTO TechnicianExpertise VALUES(62,'B777',2004,'Poor performance');
INSERT INTO TechnicianExpertise VALUES(62,'DC10',1995,'finished early');
INSERT INTO TechnicianExpertise VALUES(743,'A380',2000,'best illustrations');
INSERT INTO TechnicianExpertise VALUES(743,'B777',2001,'good foundations');
INSERT INTO TechnicianExpertise VALUES(743,'DC9',2003,'passed in third attempt');
INSERT INTO TechnicianExpertise VALUES(884,'A340',2004,'great work');
INSERT INTO TechnicianExpertise VALUES(95,'DC10',2005,'flying colors');
```

TestConducted

```
INSERT INTO TestConducted VALUES (12151,119,551,'12-May-2015',12,150);
INSERT INTO TestConducted VALUES (12152,120,62,'02-Jan-2014',12,300);
INSERT INTO TestConducted VALUES (12153,120,62,'21-Feb-2016',12,350);
INSERT INTO TestConducted VALUES (12154,121,743,'05-May-2013',12,180);
INSERT INTO TestConducted VALUES (12155,122,743,'16-Jul-2015',12,190);
INSERT INTO TestConducted VALUES (12156,122,743,'25-Apr-2015',12,100);
INSERT INTO TestConducted VALUES (12156,121,743,'25-Sep-2015',15,110);
INSERT INTO TestConducted VALUES (12156,120,743,'21-Oct-2015',18,112);
INSERT INTO TestConducted VALUES (12156,119,743,'01-Nov-2015',2,85);
INSERT INTO TestConducted VALUES (12157,122,62,'18-Jun-2015',12,140);
INSERT INTO TestConducted VALUES (12157,120,62,'01-Jun-2014',15,190);
INSERT INTO TestConducted VALUES (12157,119,62,'11-Jan-2012',20,195);
INSERT INTO TestConducted VALUES (12157,121,62,'15-Feb-2013',21,165);
```

Transaction using embedded SQL: Query & Results

1. Insert a new technician into the database.

```
INSERT INTO Employee VALUES(1015,2511);
INSERT INTO Technician VALUES (1015,'John Smith',10000,'6512 Santona St.',3056598654);
```

2. Delete an existing airplane from the database.

```
INSERT INTO Airplane VALUES(125116,'Y');
DELETE FROM Airplane WHERE regNo=125116;
```

3. Update the expertise of an existing technician.

```
UPDATE TechnicianExpertise
SET modelNo='B737'
WHERE ssn=743 AND modelNo='B777';
```

4. List the details of the technician whose salary is greater than the average of the salary of all technicians.

```
SELECT ssn,name,salary,address,phone
FROM Technician
WHERE salary >
        (SELECT AVG(salary)
         FROM Technician);
```

5. List all the model numbers that a given technician has the expertise, along with their capacity and weight information.

```
SELECT t.modelNo, m.capacity, m.weight
FROM TechnicianExpertise t, AirplaneModel m
WHERE t.ssn=62 AND t.modelNo=m.modelNo;
```

6. List the total number of technicians who are experts in each model.

```
SELECT modelNo, COUNT(ssn) AS Experts
FROM TechnicianExpertise
GROUP BY modelNo;
```

7. List the details (test number, test name, maximum score, etc.) of the FAA tests for a given airplane, sorted by the maximum scores.

```
SELECT FAANo,name,maxScore
FROM TestType
WHERE FAANo IN
                (SELECT testNo
                FROM TestConducted
                WHERE regNo = 12157)
ORDER BY maxScore;
```

8. List the most recent annual medical examination and his/her union membership number for each traffic controller.

```
SELECT e.unionMemNo AS union, MAX(c.examDate) AS Recent
FROM TrafficController c, Employee e
WHERE e.unionMemNo =
                        (SELECT unionMemNo
                        FROM Employee
                        WHERE ssn = c.ssn)
GROUP BY e.unionMemNo;
```

9. List the total number of tests done by each technician for a given airplane.

```
SELECT ssn, COUNT(testNo) AS Tests
FROM TestConducted
WHERE regNo=12157
GROUP BY ssn;
```

10. List the name of the technician, the registration number of the airplane, and the FAA number of those tests done between September 2015 and December 2015, sorted by the FAA numbers.

```
SELECT t.name AS name, c.regNo AS regNo, c.testNo AS FAANo
FROM TestConducted c, Technician t
WHERE (c.testDate BETWEEN '01-Sep-2015' AND '31-Dec-2015')
        AND c.ssn=t.ssn;
```

Embedded SQL Results

```
 Recently added new Technician

ssn         Name                 salary    address       phone
-----   -------------          ----------  ------        ---------
551      John Oliver              5000 6512 Santona St. Apt 1   3052212521
62       Marc Anthony             7000 815 Gables St. Apt 2     3054512516
743      Ann hathaway             9000 11 McArthur St. Apt 5    3059856321
884      John Smith               8000 87 Coral St. Apt 20      3055484312
95       Santa Barbara            7500 12 Senza St. Apt 14      3058745621


 Result after deleting existing Airplane

 regNo         Model No
--------   ---------------
12151    DC10
12152    B737
12153    B737
12154    A380
12155    A380
12156    B777
12157    B777


 Details of Tech. Expertise after update

 ssn      Model No         Comments
------  -------------   -----------------
551      DC10                    Aced the test
62       B737                    Best in this class
62       B777                    Poor performance
62       DC10                    finished early
743      A380                    best illustrations
743      B737                    good foundations
743      DC9                     passed in third attempt
884      A340                    great work
95       DC10                    flying colors
```

```
 Details of technicians whose salary is greater than the average

 ssn           name         salary          address            phone
------   ---------------   ---------   ----------------------   ------
743      Ann hathaway             9000 11 McArthur St. Apt 5    3059856321
884      John Smith               8000 87 Coral St. Apt 20      3055484312
95       Santa Barbara            7500 12 Senza St. Apt 14      3058745621


Model No. Capacity and Weight of Technician 62

 Model No.             Capacity    Weight
-----------           ----------  ----------
DC10                     134     2000
B737                     129     1900
B777                     302     2550
```

```
Total No. of Tech. Expert in each model

 Model No.     Experts
-----------   ----------
DC9                     1
A340                    1
DC10                    3
A380                    1
B777                    1
B737                    2


Test details of Airplane with regNo 12157

 FAA No.       Name            maxScore
---------   --------------  --------------
121      Strobe light test        320
120      Flight path test         450
119      Flight deck test         500
122      Stability test           650


Most Recent Annual Exam and union no.

 Union No.     Recent Exam
-----------   -------------
1137             12-FEB-16
1136             01-MAY-16
1138             15-JAN-15
1139             21-APR-14


Total test done by each Tech for Airplane 12157

 ssn       Tests
-------  ----------
62               4


Details of tests between Sep2015 and Dec2015

 Name                  Reg No.      FAA No
-------              ----------  ----------
Ann hathaway             12156   121
Ann hathaway             12156   120
Ann hathaway             12156   119

GOOD-BYE!!
```

Embedded SQL Code

```
/*
 *  procdemo.pc
 *
 *  This program connects to ORACLE, declares and opens a cursor,
 *  and performs the queries mentioned in the case study
 *  displays the results, then closes the cursor.
 */

#include <stdio.h>
#include <string.h>
#include <sqlca.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>

#define UNAME_LEN      20
#define PWD_LEN        25

typedef char asciiz[PWD_LEN];

EXEC SQL TYPE asciiz IS CHARZ(PWD_LEN) REFERENCE;
asciiz     username;
asciiz     password;

struct emp_info
{
    int ssn;
    asciiz name;
    int salary;
    asciiz address;
    asciiz phone;
};

struct del_plane
{
    int regNo;
    asciiz modelNo;
};

struct upd_exp
{
    int ssn;
    asciiz modelNo;
    asciiz comments;
};

struct avg_sal
{
   int ssn;
   asciiz name;
   int salary;
   asciiz address;
   asciiz phone;
};

struct tech_models
{
  asciiz modelNo;
  int capacity;
  int weight;
};

struct total_experts
{
  asciiz modelNo;
  int Experts;
};
```

```
struct test_12157
{
  int FAANo;
  asciiz name;
  int maxScore;
};

struct recent_med
{
  int unionMemNo;
  asciiz Recent;
};


struct total_tests
{
  int ssn;
  int Tests;
};

struct sep_dec
{
  asciiz name;
  int regNo;
  int FAANo;
};

void sql_error(msg)
    char *msg;
{
    char err_msg[512];
    size_t buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);

/* Call sqlglm() to get the complete text of the
 * error message.
 */
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK RELEASE;
    exit(EXIT_FAILURE);
}

void main()
{
    struct del_plane *del_plane_ptr;
    struct upd_exp *upd_exp_ptr;
    struct emp_info *emp_rec_ptr;
    struct avg_sal *avg_sal_ptr;
    struct tech_models *tech_models_ptr;
    struct total_experts *total_experts_ptr;
    struct test_12157 *test_12157_ptr;
    struct recent_med *recent_med_ptr;
    struct total_tests *total_tests_ptr;
    struct sep_dec *sep_dec_ptr;

/* Allocate memory for emp_info struct. */
    if ((emp_rec_ptr =
        (struct emp_info *) malloc(sizeof(struct emp_info))) == 0)
    {
        fprintf(stderr, "Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    if ((del_plane_ptr =
        (struct del_plane *) malloc(sizeof(struct del_plane))) == 0)
```

```
    {
        fprintf(stderr, "Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    if ((upd_exp_ptr =
        (struct upd_exp *) malloc(sizeof(struct upd_exp))) == 0)
    {
        fprintf(stderr, "Memory allocation error.\n");
        exit(EXIT_FAILURE);
    }

    if ((avg_sal_ptr =
       (struct  avg_sal *) malloc(sizeof(struct avg_sal))) == 0)
    {
        fprintf(stderr, "Memory allocation error2.\n");
        exit(EXIT_FAILURE);
    }

     if((tech_models_ptr =
        (struct tech_models *) malloc(sizeof(struct tech_models))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

     if((total_experts_ptr =
        (struct total_experts *) malloc(sizeof(struct total_experts))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

      if((test_12157_ptr =
        (struct test_12157 *) malloc(sizeof(struct test_12157))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

      if((recent_med_ptr =
        (struct recent_med *) malloc(sizeof(struct recent_med))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

      if((total_tests_ptr =
        (struct total_tests *) malloc(sizeof(struct total_tests))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

      if((sep_dec_ptr =
        (struct sep_dec *) malloc(sizeof(struct sep_dec))) == 0)
     {
        fprintf(stderr, "Memory allocation error3,\n");
        exit(EXIT_FAILURE);
     }

/* Connect to ORACLE. */
    strcpy(username, "musa567");
    strcpy(password, "sado1896");

    EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--");

    EXEC SQL CONNECT :username IDENTIFIED BY :password;
    printf("\nConnected to ORACLE as user: %s\n", username);

/* Declare the cursor. All static SQL explicit cursors
 * contain SELECT commands. 'salespeople' is a SQL identifier,
```

```
 * not a (C) host variable.
 */
    EXEC SQL DECLARE firstquery CURSOR FOR
        SELECT ssn,name,salary,address,phone
            FROM Technician;

/* Open the cursor. */
    EXEC SQL OPEN firstquery;

    EXEC SQL DECLARE secondquery CURSOR FOR

        SELECT regNo,modelNo
        FROM Airplane;

    EXEC SQL OPEN secondquery;

    EXEC SQL DECLARE thirdquery CURSOR FOR

        SELECT ssn,modelNo,comments
        FROM TechnicianExpertise;

    EXEC SQL OPEN thirdquery;

    EXEC SQL DECLARE fourthquery CURSOR FOR

        SELECT ssn,name,salary,address,phone
        FROM Technician
        WHERE salary >
                        (SELECT AVG(salary)
                         FROM Technician);

    EXEC SQL OPEN fourthquery;

    EXEC SQL DECLARE fifthquery CURSOR FOR

        SELECT t.modelNo AS modelNo, m.capacity AS capacity, m.weight AS weight
        FROM TechnicianExpertise t, AirplaneModel m
        WHERE t.ssn=62 AND t.modelNo=m.modelNo;

    EXEC SQL OPEN fifthquery;

   EXEC SQL DECLARE sixthquery CURSOR FOR

        SELECT modelNo, COUNT(ssn) AS Experts
        FROM TechnicianExpertise
        GROUP BY modelNo;

    EXEC SQL OPEN sixthquery;

   EXEC SQL DECLARE seventhquery CURSOR FOR

        SELECT FAANo,name,maxScore
        FROM TestType
        WHERE FAANo IN
                        (SELECT testNo
                        FROM TestConducted
                        WHERE regNo = 12157)
        ORDER BY maxScore;

    EXEC SQL OPEN seventhquery;

    EXEC SQL DECLARE eighthquery CURSOR FOR

       SELECT e.unionMemNo AS unionMemNo, MAX(c.examDate) AS Recent
        FROM TrafficController c, Employee e
        WHERE e.unionMemNo =
                                (SELECT unionMemNo
                                FROM Employee
                                WHERE ssn = c.ssn)
        GROUP BY e.unionMemNo;

    EXEC SQL OPEN eighthquery;
```

```
    EXEC SQL DECLARE ninthquery CURSOR FOR

        SELECT ssn, COUNT(testNo) AS Tests
        FROM TestConducted
        WHERE regNo=12157
        GROUP BY ssn;

    EXEC SQL OPEN ninthquery;

    EXEC SQL DECLARE tenthquery CURSOR FOR

       SELECT t.name AS name, c.regNo AS regNo, c.testNo AS FAANo
        FROM TestConducted c, Technician t
        WHERE (c.testDate BETWEEN '01-Sep-2015' AND '31-Dec-2015')
                 AND c.ssn=t.ssn;

    EXEC SQL OPEN tenthquery;

/* Get ready to print results. */
    printf("\n\n  Recently added new Technician\n\n");
    printf("ssn         Name                salary    address       phone\n");
    printf("-----   -------------         ----------  ------        ---------    \n");

    EXEC SQL WHENEVER NOT FOUND DO break;

    for (;;)
    {
        EXEC SQL FETCH firstquery INTO :emp_rec_ptr;
        printf("%d \t %s %d %s %s\n", emp_rec_ptr->ssn,
                emp_rec_ptr->name, emp_rec_ptr->salary,emp_rec_ptr->address,emp_rec_ptr->phone);
    }

        printf("\n\n Result after deleting existing Airplane \n\n");
        printf(" regNo         Model No    \n");
        printf("--------   --------------- \n");

    for(;;)
        {
         EXEC SQL FETCH secondquery INTO:del_plane_ptr;
         printf("%d \t %s \n", del_plane_ptr->regNo,del_plane_ptr->modelNo);
        }

        printf("\n\n Details of Tech. Expertise after update \n\n");
        printf(" ssn      Model No         Comments       \n");
        printf("------  -------------   ----------------- \n");

    for(;;)
        {
         EXEC SQL FETCH thirdquery INTO:upd_exp_ptr;
         printf("%d \t %s %s \n", upd_exp_ptr->ssn,upd_exp_ptr->modelNo,upd_exp_ptr->comments);
        }

        printf("\n\n Details of technicians whose salary is greater than the average \n\n");
        printf(" ssn           name         salary          address           phone  \n");
        printf("------   ---------------   ---------   ----------------------   ------ \n");

    for(;;)
        {
         EXEC SQL FETCH fourthquery INTO:avg_sal_ptr;
         printf("%d \t %s %d %s %s \n", avg_sal_ptr->ssn,avg_sal_ptr->name,avg_sal_ptr->salary,
                           avg_sal_ptr->address,avg_sal_ptr->phone);
        }

        printf("\n\nModel No. Capacity and Weight of Technician 62 \n\n");
        printf(" Model No.             Capacity    Weight   \n");
        printf("-----------           ----------  ----------\n");

     for(;;)
        {
         EXEC SQL FETCH fifthquery INTO:tech_models_ptr;
         printf("%s %d \t %d\n",tech_models_ptr->modelNo,tech_models_ptr->capacity,tech_models_ptr->weight);
```

```
        }

        printf("\n\nTotal No. of Tech. Expert in each model \n\n");
        printf(" Model No.     Experts  \n");
        printf("-----------   ----------\n");

     for(;;)
        {
         EXEC SQL FETCH sixthquery INTO:total_experts_ptr;
         printf("%s %d\n",total_experts_ptr->modelNo,total_experts_ptr->Experts);
        }

        printf("\n\nTest details of Airplane with regNo 12157 \n\n");
        printf(" FAA No.       Name            maxScore   \n");
        printf("---------   --------------  --------------\n");

     for(;;)
        {
         EXEC SQL FETCH seventhquery INTO:test_12157_ptr;
         printf("%d \t %s %d\n",test_12157_ptr->FAANo,test_12157_ptr->name,test_12157_ptr->maxScore);
        }

        printf("\n\nMost Recent Annual Exam and union no. \n\n");
        printf(" Union No.     Recent Exam \n");
        printf("-----------   -------------\n");

     for(;;)
        {
         EXEC SQL FETCH eighthquery INTO:recent_med_ptr;
         printf("%d \t\t %s\n",recent_med_ptr->unionMemNo,recent_med_ptr->Recent);
        }


        printf("\n\nTotal test done by each Tech for Airplane 12157 \n\n");
        printf(" ssn       Tests    \n");
        printf("-------  ---------- \n");

     for(;;)
        {
         EXEC SQL FETCH ninthquery INTO:total_tests_ptr;
         printf("%d \t\t %d\n",total_tests_ptr->ssn,total_tests_ptr->Tests);
        }

        printf("\n\nDetails of tests between Sep2015 and Dec2015\n\n");
        printf(" Name                  Reg No.      FAA No   \n");
        printf("-------              ----------  ----------\n");

     for(;;)
        {
         EXEC SQL FETCH tenthquery INTO:sep_dec_ptr;
         printf("%s %d \t %d \n",sep_dec_ptr->name,sep_dec_ptr->regNo,sep_dec_ptr->FAANo);
        }

/* Close the cursor. */
    EXEC SQL CLOSE firstquery;
    EXEC SQL CLOSE fourthquery;
    EXEC SQL CLOSE fifthquery;
    EXEC SQL CLOSE sixthquery;
    EXEC SQL CLOSE seventhquery;
    EXEC SQL CLOSE eighthquery;
    EXEC SQL CLOSE ninthquery;
    EXEC SQL CLOSE tenthquery;

    printf("\nGOOD-BYE!!\n\n");

    EXEC SQL COMMIT WORK RELEASE;
    exit(EXIT_SUCCESS);
}
```