PROJECT: DESIGN, DEVELOPMENT, AND IMPLEMENTATION OF A RELATIONAL DATABASE

ECE 467/ CSC 423 – Database Design & Management

Airport Information System

Hay, Karysse Dominique
C23722255

Contents

Introduction

The airport information system is an essential database system that helps in organizing information about airplanes stationed and maintained at the airport. In this database design report, we will be outlining a comprehensive database design for an airport information system in the XYZ county. The design will include details on how to store information about airplane models, registration numbers, airplane capacity, weight, and technician expertise. Furthermore, the design will also capture information about traffic controllers, their medical examination dates, and employee union membership numbers. Additionally, the database design will include details on tests that the airport uses to ensure the airworthiness of airplanes and how to store information on testing events. This report will also specify the assumptions that have been made in the design.

Case Study

Airport Information System

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.

Assumptions

These are the following assumptions made throughout the entire design process of the Airport Database.

- Each airplane has a unique registration number.
- Each airplane has a unique model number.
- Each technician has a unique SSN.
- Each traffic controller has a unique identification number which is the social security number.
- Each employee has a unique member number in the employee union.
- Each type of test has a unique FAA test number.
- Each testing event is unique and is attached to its own record number.
- Each airplane is only maintained and stationed at one airport (i.e., this is a single-airport system)
- A technician can perform multiple testing events for the same airplane and test.

Conceptual Model

Relations

- airplane (**registrationNo**, **modelNo)**
- airplaneModel (**modelNo**, capacity, weight)
- Technicians (**emplSSN**, fname, lname, address, phoneNo, salary)
- technicianExpertise (**recordNo**, modelNo, emplSSN)
- trafficController (**emplSSN**, fname, lname, address, phone, recentDate)
- employeeUnion (**emplSSN**, **memberNo**)
- testRecords **(recordNo**, registrationNo, emplSSN, faaNo, date, noHours, maxScore)
- testingInfo (**faaNo**, testName, maxScore)

**in bold are specified primary keys

Simple EER Diagram

Cardinality Constraints

The relationships between the entities:

- airplane: one-to-many relationship with airplaneModel
- Technicians: many-to-many relationship with airplaneModel through technicianExpertise
- trafficController: one-to-one relationship with Technicians
- testRecords: many-to-one relationship with airplane, technicians, and testingInfo
- testingInfo: one-to-many relationship with testRecords
- employeeUnion: one-to-one relationship with Technicians and trafficController

Association between Tables

1. Table 1 - airplane:
   The primary key is a "registrationNo" and the foreign key is "modelNo" which is a reference to the primary key in Table 2 - airplaneModel, which specifies the model of the airplane.

2. Table 2 - airplaneModel:
   The primary key is "modelNo" as it uniquely identifies each airplane model in the table. There are no foreign keys.

3. Table 3 - Technicians:
   The primary key is "emplSSN" as it uniquely identifies each technician in the table. There are no foreign keys.

4. Table 4 - technicianExpertise:
   The primary key is "recordNo" because it uniquely identifies each expertise record in the table. The foreign keys are "modelNo" and "emplSSN" as they reference the primary keys in Table 2 - airplaneModel and Table 3 - Technicians respectively, which specify the airplane model and the technician's social security number associated with each expertise record.

5. Table 5 - trafficController: The primary key is "emplSSN" as it uniquely identifies each traffic controller in the table. There are no foreign keys.

6. Table 6 - employeeUnion: The primary key is "memberNo" because it uniquely identifies each member in the table. The foreign key is "emplSSN" and is a reference to the primary key in Table 3 – Technicians.

7. Table 7 - testingInfo: The primary key is "faaNo" because it uniquely identifies each FAA test in the table. The composite primary key "faaNo" and "testName" together uniquely identify each test in the table.

8. Table 8 - testRecords: The primary key is "recordNo" because it uniquely identifies each test record in the table. The foreign keys are "registrationNo", "emplSSN", and "faaNo" and they reference the primary keys in Table 1 - airplane, Table 3 - Technicians, and Table 7 - testingInfo respectively.

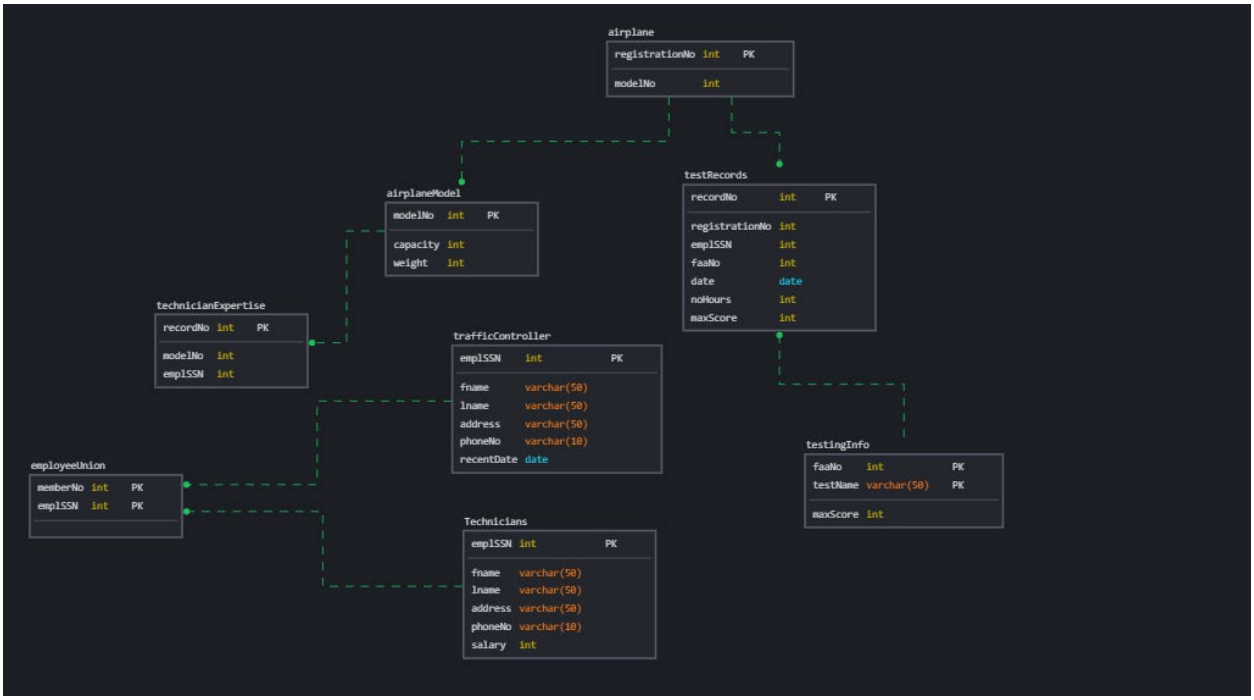
Specialization and Generalization

A potential area where these types of relationships could be applied are:

- The current employee union table includes both technicians and traffic controllers. There are two different type of employees which have different attributes and relationships with other entities. A specialization/generalization hierarchy could be created to separate these two types of employees into their own tables, with common attributes and relationships defined in the parent employee entity.

Logical Data Model

EER – Enhanced Entity-Relationship Diagram of Airport Database (Improved)



Database Tables Information

Table 1 - airplane

Field	Type	Null	Key
registrationNo	int	not null	primary key
modelNo	varchar(50)	not null	foreign key

Table 2 – airplaneModel

Field	Type	Null	Key
modelNo	varchar(50)	not null	primary key
capacity	int	not null	n/a
wieight	int	not null	n/a

Table 3 – Technicians

Field	Type	Null	Key
emplSSN	int	not null	primary key
fname	varchar(50)	not null	n/a
lname	varchar(50)	not null	n/a
address	varchar(50)	not null	n/a
phoneNo	varchar(10)	not null	n/a
salary	int	not null	n/a

Table 4 – technicianExpertise

Field	Type	Null	Key
recordNo	int	not null	primary key
modelNo	varchar(50)	not null	foreign key
emplsSSN	int	not null	foreign key

Table 5 – trafficController

Field	Type	Null	Key
emplSSN	int	not null	primary key
fname	varchar(50)	not null	n/a
lname	varchar(50)	not null	n/a
address	varchar(50)	not null	n/a
phoneNo	varchar(10)	not null	n/a
recentDate	date		n/a

Table 6 – employeeUnion

Field	Type	Null	Key
emplSSN	int	not null	primary key
memberNo	int	not null	primary key

Table 7 – testingInfo

Field	Type	Null	Key
faaNo	int	not null	primary key
testName	varchar(50)	not null	primary key
maxScore	int	not null	n/a

Table 8 – testRecords

Field	Type	Null	Key
recordNo	int	not null	primary key
registrationNo	int	not null	foreign key
emplSSN	int	not null	foreign key
faaNo	int	not null	foreign key
date	date	not null	n/a
noHours	int	not null	n/a
maxScore	int	not null	n/a

Validation using BCNF

What is it?

BCNF stands for Boyce-Codd Normal Form, which is a higher level of normalization for relational database tables. It is designed to remove redundancy and dependency issues that may arise in the design the tables.

Validating:

- The airplane table has a primary key of registrationNo and a foreign key of modelNo which references the airplaneModel table.
- The airplaneModel table has a primary key of modelNo and no dependencies on any other attributes.
- The Technicians table has a primary key of emplSSN and no dependencies on any other attributes.
- The technicianExpertise table has a primary key of recordNo with foreign keys of emplSSN, and modelNo.
- The TrafficController table has a primary key of emplSSN and no dependencies on any other attributes.
- The employeeUnion table has a composite primary key of emplSSN and memberNo. All attributes in the table are dependent on the composite primary key.
- The testingInfo table has a composite primary key of faaNo and testName. All attributes in the table are dependent on the composite primary key.
- The testRecords table has a primary key of recordNo and foreign keys of registrationNo, emplSSN, and faaNo. All attributes in the table are dependent on the primary key.

All tables satisfy BCNF requirements due to the extensive design process. No further normalization is necessary.

Discussion and Analysis

Table 1: Airplane

This table was straight forward. It needed to give each airplane a model number and a registration number. Since both of these numbers are unique, they can both be considered the primary keys his became the two primary keys as they can both uniquely identify the information in this table.

Table 2: airplaneModel

For this table, each model of airplane had its own capacity and weight. This had to be recorded which resulted in this table having three columns: modelNo, capacity and weight. Now, the capacity and weight values can be repeated as some airplanes will have the same values. This meant that only modelNo was the primary key and could be linked to Table 1. Therefore, making it a foreign key in Table 1 as well.

Table 3: Technicians

This table had to record the standard information for each technician where they were all identified uniquely by their social security number. Since no two technicians can have the same social security number, emplSSN became the primary key for this table. All other information like first name, last name, address etc. was recorded for that specific emplSSN.

Table 4: technicianExpertise

This table was to record what model of airplanes was each technician an expertise in. For this, one technician could have been the expertise in more than one model. Therefore, this meant that both emplSSN and modelNo would be repeated numerous times or could have been. To take care of this, I created another column called recordNo which would serve as the primary key for this table. This meant that each record of expertise was attached to a recordNo that was unique to it.

Table 5: trafficController

This table was a lot like Technicians. It required a unique identifier which was also called emplSSN which we will see was done purposely for Table 6- Employee Union. Since technicians and traffic controllers are both employees and would both be appearing in Table 6, it was just easier to create one parameter called emplSSN which would serve as the employee social security number. This table had a lot of the same information such as first name, last name, address etc.

Table 6: employeeUnion

This table records each employee at the airport. For this report and the instructions, the only employees included were the technicians and the traffic controllers. This table had two columns: emplSSN and memberNo. This meant that each employee was assigned a unique member number. Therefore, both columns could have been considered primary keys as they were both unique as no two employees can have the same social security number nor the same member number.

Table 7: testingInfo

Based on the instructions, this table was required to record each possible test that the airport can offer, and each test was uniquely identified by a FAA number. Therefore, this resulted in a table consisting of

each test that can be identified by their own FAA number. Each test had a name and a record of its maximum score. The Test Name was also unique to this table and can also be considered as a primary key.

Table 8- testingInfo

This table held the record of the tests done over the years. It recorded which test, which technician did the test, what plane the test was done for, how long the test took, what date the test took place on and the score it achieved on the test. Since a technician's social security number and the FAA number could have easily been repeated in the database, a new column called recordNo was added to uniquely identify each time a test was done. A similar thing was done for Table 4.

Results

MySQL Database Tables

airplane

registrationNo	modelNo
100012	AA-10
100013	AB-12
100014	AC-14
100015	AD-16
100016	AE-17
100017	AG-22
100018	AG-23
100019	AH-24
100020	AI-27
100021	AJ-26
100022	AK-30
100023	AL-31
100024	AM-32
100025	AM-33
100026	AN-34
100027	AO-37
100028	AP-38
100029	AQ-38
100030	AR-40
100031	AS-40
100032	AT-41
100033	AU-42
100034	AX-43
100035	AY-44
100036	AZ-45
100037	BA-01
100038	BB-02
100039	BC-03
100040	BD-04
100041	BE-05
100042	BF-06
100043	BG-07
100044	BH-08
100045	BI-09
100046	BJ-10
100047	BK-11
100048	BL-12
100049	BM-13
100050	BN-14
100051	BO-15
100052	BP-16
100053	BQ-17
100054	BR-18
100055	BS-19
100056	BT-20
100057	BU-21
100058	BV-22
100059	BW-23
100060	BX-24

aiplaneModel

modelNo	capacity	weight
AA-10	100	337100
AB-12	100	337120
AC-14	100	337100
AD-16	115	340110
AE-17	85	310500
AG-22	85	300500
AG-23	100	337120
AH-24	115	345500
AI-27	65	250140
AJ-26	120	401450
AK-30	85	300190
AL-31	95	320500
AM-32	100	339450
AM-33	100	340370
AN-34	95	300120
AO-37	85	285340
AP-38	115	290100
AQ-38	120	320340
AR-40	125	350200
AS-40	125	355100
AT-41	65	220100
AU-42	100	343490
AX-43	115	344340
AY-44	120	320900
AZ-45	125	348500
BA-01	120	220000
BB-02	120	200000
BC-03	120	300000
BD-04	110	345000
BE-05	100	100400
BF-06	65	95000
BG-07	100	100785
BH-08	100	100090
BI-09	300	500000
BJ-10	210	400000
BK-11	200	430000
BL-12	210	430000
BM-13	230	430000
BN-14	215	320000
BO-15	180	200100
BP-16	160	190600
BQ-17	150	140020
BR-18	150	102030
BS-19	140	100040
BT-20	130	200020
BU-21	120	103050
BV-22	120	100490
BW-23	100	100370
BX-24	200	200450
NULL	NULL	NULL

Technicians

emplSSN	fname	lname	address	phoneNo	salary
123456789	Aaron	And...	23 First St	8765432146	4567890
234567890	Ben	Brown	24 First St	7654336789	3456780
345678901	Casey	Clark	25 First St	2345678876	456780
456789012	Dan	Davis	32 Second St	3075478886	9876500
567890123	Elias	Evans	33 Second St	8670008706	2345670
678901234	Fin	Foster	34 Third St	1288749404	456780
789012345	Gerard	Garcia	35 Third St	4780303704	765090
890123456	Henny	Hill	37 Main St	8768983900	765430
901234567	Ines	Irwin	38 Red Ave	1838447893	754000
123456780	Jake	Jones	39 Pink Ave	2345678909	3456780
234567891	Ken	King	40 Blue Ave	6333567899	5678900
345678912	Lana	Lopez	41 Blue Ave	8765433456	398760
456789123	Mona	Miller	42 Green St	3054678999	987600
567891234	Ned	Nunez	43 Green ...	4567890980	456780
678912345	Olive	Olson	44 Brown ...	2872647894	456780
789123456	Patrik	Phipps	45 Black Blvd	4938485795	8765430
891234567	Quinn	Quiroz	46 New St	3994949000	765400
912345678	Ryan	Rivera	47 New St	8484902234	987654
987654321	Sam	Smith	48 Old St	2394009440	987654
876543219	Tyler	Torres	49 Old St	8765678303	8765400
765432198	Uriah	Urban	50 Fourth St	3847393042	567890
654321987	Veronica	Vargas	51 Fourth St	2345670841	567890
543219876	Wyatt	Wilson	52 Fifth St	3400966000	987650
432198765	Xile	Xu	53 Fifth St	6543458658	9876580
321987654	Yoland	Yang	54 Sixth St	9875545766	3456700
219876543	Zed	Zhang	55 Sevent...	7543797554	345670

techniciainsExpertise

recordNo	emplSSN	modelNo
1	345678912	AA-10
2	123456789	AA-10
3	234567891	AA-10
4	789123456	AB-12
5	123456789	AB-12
6	345678912	AB-12
7	789123456	AA-10
8	567891234	BH-08
9	678912345	BX-24
10	789123456	BP-16
11	123456789	BU-21
12	234567891	AO-37
13	567891234	AT-41
14	987654321	BJ-10
15	876543219	AM-13
16	765432198	AT-41
17	123456780	AI-27
18	123456780	AM-32
19	654321987	AA-10
20	987654321	BO-15
21	891234567	BD-04
22	432198765	AA-10
23	123456780	AP-38
24	123456780	AQ-38
25	123456780	BD-04
26	345678912	AD-16
27	345678912	AG-22
28	123456780	BG-07
29	432198765	AM-32
30	890123456	BS-19
31	987654321	BS-19
32	234567890	BV-22
33	876543219	BS-19
34	901234567	AK-30
35	901234567	BT-20
36	345678912	AE-17
37	123456780	AJ-26
38	567890123	BH-09
39	345678912	BL-12
40	123456780	AY-44
41	234567890	BM-13
42	890123456	BR-18
43	891234567	BI-09
44	912345678	AD-16
45	765432198	BN-14
46	901234567	BL-12
47	456789123	AS-40
48	456789123	BS-19
49	912345678	AH-24
50	234567890	BQ-17
51	765432198	BR-18
52	456789012	AS-40
53	890123456	BW-23
54	912345678	AJ-26
55	890123456	BG-08
56	345678123	BX-24
57	219876543	BV-22
58	901234567	AA-10
59	765432198	BE-05
60	345678901	BX-24
61	345678912	AG-23
62	345678901	BI-09
63	123456780	BE-05
64	901234567	AL-31
65	912345678	BX-24
66	234567891	AN-34
67	765432198	AM-33
68	123456789	AM-32
69	432198765	AK-30
70	123456789	AZ-45
71	123456789	AX-43
72	123456789	BC-03
73	123456789	BB-02
74	123456789	BA-01

trafficController

emplSSN	fname	lname	address	phoneNo	recentDate
111111111	Anya	Alls	90 First St	4567890870	2022-01-03
222222222	Betty	Bates	91 Second St	3456798760	2023-03-01
333333333	Cody	Cabello	92 Third St	3467876500	2023-02-14
444444444	Dannie	Dalton	93 Fourth St	9765445679	2023-01-12
555555555	Ella	East	94 Fifth St	7653456789	2022-04-14
666666666	Faith	Flinstone	95 Sixth St	2345653454	2022-04-17
777777777	Gabby	Gurbert	96 Seventh St	5433543454	2023-01-21
888888888	Hannah	Hitch	97 Eighth St	3656576543	2022-09-11
999999999	Iana	Ivey	98 Nineth St	8765450009	2022-12-09
NULL	NULL	NULL	NULL	NULL	NULL

employeeUnion

emplSSN	memberNo		
123456780	1		
123456789	2		
219876543	3		
234567891	4		
321987654	5		
345678912	6	901234567	21
432198765	7	912345678	22
456789012	8	987654321	23
456789123	9	111111111	24
543219876	10	222222222	25
567891234	11	333333333	26
654321987	12	444444444	27
678901234	13	555555555	28
678912345	14	666666666	29
765432198	15	777777777	30
789012345	16	888888888	31
789123456	17	999999999	32
876543219	18	234567890	33
890123456	19	567890123	34
891234567	20	345678901	35

testingInfo

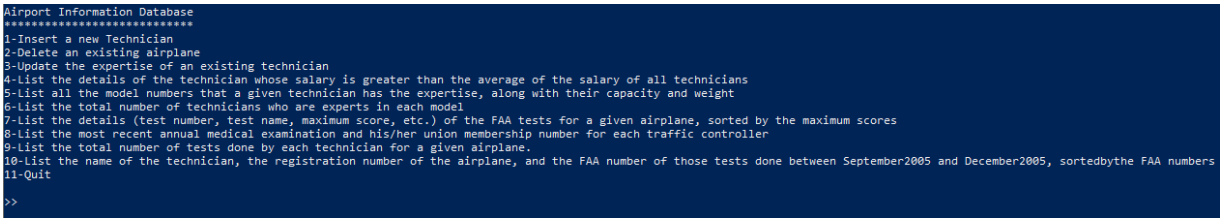
	testName	maxScore	faaNo
▶	Test 1	100	101
	Test 2	150	102
	Test 3	400	103
	Test 4	500	104
	Test 5	650	105
	Test 6	120	106
	Test 7	500	107
	Test 8	650	108
	Test 9	850	109
	Test 10	200	110
	Test 11	250	111
	Test 12	240	112
	Test 13	240	113
	Test 14	250	114
	Test 15	350	115
	Test 16	1050	116
	Test 17	850	117
	Test 18	750	118
	Test 19	950	119
	Test 20	2100	120
*	NULL	NULL	NULL

testRecords

recordNo	registrationNo	emplSSN	faaNo	date	noHours	maxScore
1	100012	345678912	101	2005-09-12	5	95
2	100013	789123456	102	2005-09-14	5	140
3	100014	234567891	103	2001-01-12	6	400
4	100015	123456789	104	2002-08-08	6	480
5	100016	123456789	105	2005-09-18	7	645
6	100017	123456789	106	2005-10-21	3	120
7	100018	123456789	107	2005-11-06	3	430
8	100024	789123456	108	2005-10-11	3	600
9	100025	678912345	109	2005-11-12	4	845
10	100026	567891234	110	2005-10-20	7	140
11	100027	123456780	111	2005-11-03	4	230
12	100028	123456780	112	2005-11-19	3	230
13	100029	654321987	113	2005-09-11	2	200
14	100030	432198765	114	2005-09-05	7	200
15	100031	432198765	115	2005-10-27	8	310
16	100033	345678912	116	2003-12-01	5	1000
17	100034	345678912	117	2001-04-14	8	800
18	100035	123456780	118	2001-04-09	5	710
19	100036	123456780	119	2001-06-01	3	935
20	100037	123456780	120	2002-09-01	7	2100
21	100038	432198765	101	2002-10-17	3	100
22	100039	890123456	102	2005-11-12	7	105
23	100040	876543219	103	2006-11-09	2	380
24	100041	345678912	104	1999-03-11	7	480
25	100042	123456780	105	2006-02-12	3	605
26	100043	234567890	106	2009-08-29	4	115
27	100044	765432198	107	2003-01-10	3	450
28	100046	456789123	108	2003-03-13	3	645
29	100048	765432198	109	2005-07-24	5	825
30	100049	901234567	110	2001-06-03	6	145
31	100052	123456780	111	2007-03-12	4	215
32	100053	912345678	112	2012-12-11	6	240
33	100054	765432198	113	2005-12-02	4	205
34	100055	123456789	114	2012-10-20	3	250
35	100056	123456789	115	2012-07-19	5	345
36	100057	123456789	116	2005-03-12	9	985
37	100058	123456789	117	2005-11-29	8	825
38	100059	123456789	118	2008-12-22	4	745
39	100060	123456789	119	2001-12-11	2	915
40	100012	234567891	120	2005-11-30	3	2000
41	100012	567891234	102	2005-09-15	4	115
42	100012	123456789	102	2005-10-31	5	145
43	100012	123456789	103	2005-10-12	6	345
44	100012	876543219	104	2005-11-12	2	435
45	100012	567891234	105	2005-09-19	3	645
46	100012	345678912	101	2010-10-21	4	100
47	100012	345678912	102	2011-10-21	2	135
48	100012	345678912	102	2014-09-17	3	140
49	100012	234567891	105	2015-04-12	6	615
50	100012	567891234	107	2003-02-28	4	350

Sample Queries

Query Menu



Query 1

Insert a new technician into the database.

Code

```
INSERT INTO Technicians (emplSSN, fname, lname, address, phoneNo,
salary)

VALUES ('101010101', 'Karysse', 'Hay', '123 Main St', '3059304043',
100000);
```

Result

```
>>1
Record inserted successfully.
SSN          First Name   Last Name    Address         Phone No     Salary
================================================================================
101010101    Karysse      Hay          123 Main St     3059304043   100000.0
123456780    Jake         Jones        39 Pink Ave     2345678909   3456780.0
123456789    Aaron        Anderson     23 First St     8765432146   4567890.0
219876543    Zed          Zhang        55 Seventh St   7543797554   345670.0
234567890    Ben          Brown        24 First St     7654336789   3456780.0
234567891    Ken          King         40 Blue Ave     6333567899   5678900.0
321987654    Yoland       Yang         54 Sixth St     9875545766   3456700.0
345678901    Casey        Clark        25 First St     2345678876   456780.0
345678912    Lana         Lopez        41 Blue Ave     8765433456   398760.0
432198765    Xile         Xu           53 Fifth St     6543458658   9876580.0
456789012    Dan          Davis        32 Second St    3075478886   9876500.0
456789123    Mona         Miller       42 Green St     3054678999   987600.0
543219876    Wyatt        Wilson       52 Fifth St     3400966000   987650.0
567890123    Elias        Evans        33 Second St    8670008706   2345670.0
567891234    Ned          Nunez        43 Green Ave    4567890980   456780.0
654321987    Veronica     Vargas       51 Fourth St    2345670841   567890.0
678901234    Fin          Foster       34 Third St     1288749404   456780.0
678912345    Olive        Olson        44 Brown Blvd   2872647894   456780.0
765432198    Uriah        Urban        50 Fourth St    3847393042   567890.0
789012345    Gerard       Garcia       35 Third St     4780303704   765090.0
789123456    Patrik       Phipps       45 Black Blvd   4938485795   8765430.0
876543219    Tyler        Torres       49 Old St       8765678303   8765400.0
890123456    Henny        Hill         37 Main St      8768983900   765430.0
891234567    Quinn        Quiroz       46 New St       3994949000   765400.0
901234567    Ines         Irwin        38 Red Ave      1838447893   754000.0
912345678    Ryan         Rivera       47 New St       8484902234   987654.0
987654321    Sam          Smith        48 Old St       2394009440   987654.0
```

Query 2

Delete an existing airplane from the database.

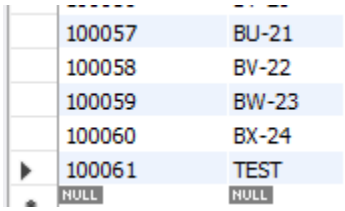
Code

DELETE FROM airplane

WHERE registrationNo = '100061';

Result

In the Airplane Table, I inserted a TEST airplane so that it can be deleted. Here it is in Workbench:



100057	BU-21
100058	BV-22
100059	BW-23
100060	BX-24
100061	TEST
NULL	NULL

Query on square:

```
>>2
Enter a Registration No: 100061
Airplane with Registration No 100061 was deleted successfully.
```

Checking to see if the Airplane still exists:

```
>>2
Enter a Registration No: 100061
Airplane with Registration No 100061 does not exist.
```

Query 3

Update the expertise of an existing technician.

Code

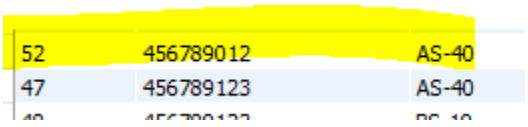
UPDATE technicianExpertise

SET ModelNo = 'AS-40'

WHERE emplSSN = '456789012';

Result

Before query, this is the Model No got Technician 456789012:



52	456789012	AS-40
47	456789123	AS-40

After query, it is now:

```
>>3
Enter the technician's SSN: 456789012
Enter the new Model Number: BS-19
SSN          Model No
========================
456789012    BS-19
```

Query 5

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

Code

```
SELECT *

FROM Technicians

WHERE salary > (SELECT AVG(salary) FROM Technicians);
```

Result

```
>>4
SSN          First Name     Last Name      Address         Phone #        Salary
=============================================================================================================
123456780    Jake           Jones          39 Pink Ave     2345678909     $3,456,780.00
123456789    Aaron          Anderson       23 First St     8765432146     $4,567,890.00
234567890    Ben            Brown          24 First St     7654336789     $3,456,780.00
234567891    Ken            King           40 Blue Ave     6333567899     $5,678,900.00
321987654    Yoland         Yang           54 Sixth St     9875545766     $3,456,700.00
432198765    Xile           Xu             53 Fifth St     6543458658     $9,876,580.00
456789012    Dan            Davis          32 Second St    3075478886     $9,876,500.00
789123456    Patrik         Phipps         45 Black Blvd   4938485795     $8,765,430.00
876543219    Tyler          Torres         49 Old St       8765678303     $8,765,400.00
```

Query 6

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

Code

```
SELECT A.modelNo, A.capacity, A.weight

FROM airplaneModel A

JOIN technicianExpertise T ON T.modelNo = A.modelNo

WHERE T.emplSSN = '123456780';
```

Result

```
>>5
Enter a Technician's SSN: 123456780
Model No        Capacity        Weight
================================================
AI-27           65              250140
AM-32           100             339450
AP-38           115             290100
AQ-38           120             320340
BD-04           110             345000
BG-07           100             100785
AJ-26           120             401450
AY-44           120             320900
BE-05           100             100400
```

Query 7

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

Code

```
SELECT modelNo, COUNT(emplSSN) AS numExperts

FROM technicianExpertise

GROUP BY modelNo;
```

Result

```
>>6
Model No     Number of Experts
==================================
AA-10        7
AB-12        3
BH-08        1
BX-24        4
BP-16        1
BU-21        1
AO-37        1
AT-41        2
BJ-10        1
AM-13        1
AI-27        1
AM-32        3
BO-15        1
BD-04        2
AP-38        1
AQ-38        1
AD-16        2
AG-22        1
BG-07        1
BS-19        4
BV-22        2
AK-30        2
```

```
BT-20        1
AE-17        1
AJ-26        2
BH-09        1
BL-12        2
AY-44        1
BM-13        1
BR-18        2
BI-09        2
BN-14        1
AS-40        2
AH-24        1
BQ-17        1
BW-23        1
BG-08        1
BE-05        2
AG-23        1
AL-31        1
AN-34        1
AM-33        1
AZ-45        1
AX-43        1
BC-03        1
BB-02        1
BA-01        1
```

Query 7

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

Code

```
SELECT T.registrationNo, T.emplSSN, I.testName, T.noHours, T.date, T.maxScore

FROM testRecords T

JOIN testingInfo I ON T.faaNo = I.faaNo

WHERE T.registrationNo = '100012'

ORDER BY T.maxScore ASC;
```

Result

```
>>7
Enter your desrired registration number: 100012
Registration No      SSN              Test Name      Hours      Date            Max Score
================================================================================================
100012               345678912        Test 1         5          12 Sep 2005     95
100012               345678912        Test 1         4          21 Oct 2010     100
100012               567891234        Test 2         4          15 Sep 2005     115
100012               345678912        Test 2         2          21 Oct 2011     135
100012               345678912        Test 2         3          17 Sep 2014     140
100012               123456789        Test 2         5          31 Oct 2005     145
100012               123456789        Test 3         6          12 Oct 2005     345
100012               567891234        Test 7         4          28 Feb 2003     350
100012               876543219        Test 4         2          12 Nov 2005     435
100012               234567891        Test 5         6          12 Apr 2015     615
100012               567891234        Test 5         3          19 Sep 2005     645
100012               234567891        Test 20        3          30 Nov 2005     2000
```

Query 8

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

Code

```
SELECT T.emplSSN, T.recentDate, E.memberNo

FROM trafficController T

LEFT JOIN employeeUnion E ON T.emplSSN = E.emplSSN

ORDER BY E.memberNo ASC;
```

Result

```
>>8
SSN             Recent Date     Member No
================================================
111111111       03 Jan 2022     24
222222222       01 Mar 2023     25
333333333       14 Feb 2023     26
444444444       12 Jan 2023     27
555555555       14 Apr 2022     28
666666666       17 Apr 2022     29
777777777       21 Jan 2023     30
888888888       11 Sep 2022     31
999999999       09 Dec 2022     32
```

Query 8

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

Code

```
SELECT emplSSN, COUNT(*) AS numTests

FROM testRecords

WHERE registrationNo = '100012'

GROUP BY emplSSN

ORDER BY numTests ASC;
```

Result

```
>>9
Enter your desrired registration number: 100012
SSN             Number of Tests
================================
876543219       1
234567891       2
123456789       2
567891234       3
345678912       4
```

Query 10

List the name of the technician, the registration number of the airplane, and the FAA number of those tests done betweenSeptember2005 and December2005, sorted by the FAA numbers.

Code

```
SELECT T.fname, T.lname, R.registrationNo, R.faaNo

FROM Technicians T

INNER JOIN testRecords R ON T.emplSSN = R.emplSSN

WHERE R.date BETWEEN '2005-09-01' AND '2005-12-31'
```

ORDER BY R.faaNo ASC;

Result

```
>>10
First Name    Last Name     Registration No      FAA No
=============================================================
Lana          Lopez         100012               101
Patrik        Phipps        100013               102
Aaron         Anderson      100012               102
Ned           Nunez         100012               102
Henny         Hill          100039               102
Aaron         Anderson      100012               103
Tyler         Torres        100012               104
Ned           Nunez         100012               105
Aaron         Anderson      100016               105
Aaron         Anderson      100017               106
Aaron         Anderson      100018               107
Patrik        Phipps        100024               108
Olive         Olson         100025               109
Ned           Nunez         100026               110
Jake          Jones         100027               111
Jake          Jones         100028               112
Veronica      Vargas        100029               113
Uriah         Urban         100054               113
Xile          Xu            100030               114
Xile          Xu            100031               115
Aaron         Anderson      100058               117
Ken           King          100012               120
```

Conclusion

The conceptual design created provided an overview of the database requirements and helped in identifying the entities, relationships, and attributes involved in this airport database. The logical design of the database was then implemented, which involved the creation of tables and their relationships, as well as the definition of constraints to ensure data integrity. This also included the insertion of many data tuples which allowed for the sample queries to be executed seamlessly.

Sample queries were then executed to test the functionality and efficiency of the database. The queries demonstrated the usefulness of the database in managing airport operations, including managing employee records, airplane registrations, and test records. These sample queries were embedded in the square server and tested again on the square server rather than on MySQL workbench.

Overall, the airport database provides a solid foundation for managing airport operations and ensuring data integrity. Further improvements and optimizations can be made in the future to enhance its functionality and performance.

Appendix

```
import mysql.connector

import datetime


# Added the username and password to change out faster

USERNAME='kdh85'

PASSWORD='~Csc4449'


def displayMenu():

    print("Airport Information Database")

    print("**************************")

    print("1-Insert a new Technician")

    print("2-Delete an existing airplane")

    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")

    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 September2005 and
December2005, sortedbythe FAA numbers")

    print("11-Quit")

    print()
```

```
def getChoice():
    choice = eval(input(">>"))
    return choice
def main():
    cnx = mysql.connector.connect(host='square.law.miami.edu',user=USERNAME,
password=PASSWORD, database='{0}_FINAL_PROJECT'.format(USERNAME))
    cursor = cnx.cursor()
    displayMenu()
    choice = getChoice()
    while(choice!=11):
        if (choice==1):
            query = ("INSERT INTO Technicians (emplSSN, fname, lname,
address, phoneNo, salary) " +
                "VALUES ('101010101', 'Karysse', 'Hay', '123 Main St',
'3059304043', '100000')")

            cursor.execute(query)

            print("Record inserted successfully.")

            cursor.execute("SELECT * FROM Technicians")

            print("{:<12} {:<12} {:<12} {:<15} {:<12} {:<10}".format(
                  "SSN", "First Name", "Last Name", "Address", "Phone No",
"Salary"))

            print("="*80)

            for (emplSSN, fname, lname, address, phoneNo, salary) in cursor:
                print("{:<12} {:<12} {:<12} {:<15} {:<12} {:<10}".format(
```

```
                        emplSSN, fname, lname, address, phoneNo, salary))

            print()

            displayMenu()
            choice=getChoice()

        elif (choice==2):
            query = ("DELETE FROM airplane " +
                "WHERE registrationNo = %s")

            regNo = input("Enter a Registration No: ")
            cursor.execute(query, (regNo,))

            if cursor.rowcount > 0:
                print("Airplane with Registration No {} was deleted
successfully.".format(regNo))
            else:
                print("Airplane with Registration No {} does not
exist.".format(regNo))

            print()

            displayMenu()
            choice=getChoice()

        elif (choice==3):
            emplSSN = input("Enter the technician's SSN: ")
            modelNo = input("Enter the new Model Number: ")
            query = "UPDATE technicianExpertise SET modelNo = %s WHERE
emplSSN = %s"
```

```
            cursor.execute(query, (modelNo, emplSSN))

            query = "SELECT emplSSN, modelNo FROM technicianExpertise WHERE
emplSSN = %s"
            cursor.execute(query, (emplSSN,))
            print("{:<12} {:<8}".format("SSN", "Model No"))
            print("="*25)

            for row in cursor.fetchall():
                print("{:<12} {:<8}".format(*row))
            print()

            displayMenu()
            choice=getChoice()

        elif (choice==4):

            result=cursor.execute("SELECT * " +
                "FROM Technicians " +
                "WHERE salary > (SELECT AVG(salary) FROM Technicians)")

            print("{:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format(
            "SSN", "First Name", "Last Name", "Address", "Phone #",
"Salary"))

            # Print a line to separate the headers from the data
            print("="*110)

            # Print each row of data from the database
            for row in cursor.fetchall():
```

```
                print("{:<15} {:<15} {:<15} {:<15} {:<15}
${:<15,.2f}".format(*row))

            print()
            displayMenu()
            choice=getChoice()


        elif (choice==5):
            query = ("SELECT A.modelNo, A.capacity, A.weight " +
                "FROM airplaneModel A " +
                "JOIN technicianExpertise T ON T.modelNo = A.modelNo "+
                "WHERE T.emplSSN = %s")

            ssn = input("Enter a Technician's SSN: ")
            cursor.execute(query, (ssn,))

            print("{:<15} {:<15} {:<15}".format(
                 "Model No", "Capacity", "Weight"))

            print("="*50)

            for row in cursor.fetchall():
                print("{:<15} {:<15} {:<15}".format(*row))

            print()

            displayMenu()
            choice=getChoice()
```

```
        elif (choice==6):
            query = ("SELECT modelNo, COUNT(emplSSN) AS numExperts " +
                "FROM technicianExpertise "+
                "GROUP BY modelNo")

            result = cursor.execute(query)

            print("{:<12} {:<15}".format(
                 "Model No", "Number of Experts"))

            print("="*45)

            for row in cursor.fetchall():
                print("{:<12} {:<15}".format(*row))



            print()
            displayMenu()
            choice=getChoice()



        elif (choice==7):
            query = ("SELECT T.registrationNo, T.emplSSN, I.testName,
T.noHours, T.date, T.maxScore " +
                "FROM testRecords T "+
                "JOIN testingInfo I ON T.faaNo = I.faaNo "+
                "WHERE T.registrationNo = %s " +
                "ORDER BY T.maxScore ASC")

            regNo = input("Enter your desrired registration number: ")
```

```
            cursor.execute(query, (regNo,))

            print("{:<20} {:<15} {:<15} {:<10} {:<15} {:<15}".format(
                 "Registration No", "SSN", " Test Name", "Hours", "Date",
"Max Score"))

            print("="*100)

            for row in cursor.fetchall():
               print("{:<20} {:<15} {:<15} {:<10} {:<15}
{:<15}".format(row[0], row[1], row[2], row[3], row[4].strftime('%d %b %Y'),
row[5]))

            print()

            displayMenu()
            choice=getChoice()



        elif (choice==8):
            query = ("SELECT T.emplSSN, T.recentDate, E.memberNo " +
                "FROM trafficController T "+
                "LEFT JOIN employeeUnion E ON T.emplSSN = E.emplSSN "+
                "ORDER BY E.memberNo ASC")

            cursor.execute(query)

            print("{:<15} {:<15} {:<15}".format(
                 "SSN", "Recent Date", "Member No"))

            print("="*50)
```

```
            for row in cursor.fetchall():
                print("{:<15} {:<15} {:<15}".format(row[0],
row[1].strftime('%d %b %Y'), row[2]))



            print()


            displayMenu()
            choice=getChoice()



        elif (choice==9):
            query = ("SELECT emplSSN, COUNT(*) AS numTests " +
                "FROM testRecords "+
                "WHERE registrationNo = %s "+
                "GROUP BY emplSSN " +
                "ORDER BY numTests ASC")

            regNo = input("Enter your desrired registration number: ")
            cursor.execute(query, (regNo,))

            print("{:<15} {:<15}".format(
                 "SSN", "Number of Tests"))

            print("="*35)

            for row in cursor.fetchall():
                print("{:<15} {:<15}".format(*row))
```

```
        print()

        displayMenu()
        choice=getChoice()


    elif (choice==10):
        query = ("SELECT T.fname, T.lname, R.registrationNo, R.faaNo " +
            "FROM Technicians T "+
            "INNER JOIN testRecords R ON T.emplSSN = R.emplSSN "+
            "WHERE R.date BETWEEN '2005-09-01' AND '2005-12-31'" +
            "ORDER BY R.faaNo ASC")

        cursor.execute(query)

        print("{:<12} {:<12} {:<20} {:<10}".format(
        "First Name", "Last Name", "Registration No", "FAA No"))

        print("="*60)

        for row in cursor.fetchall():
            print("{:<12} {:<12} {:<20} {:<10}".format(*row))

        print()

        displayMenu()
        choice=getChoice()

    elif (choice==11):
        cursor.close()
```

```
            cnx.close()

if __name__ == "__main__":
    main()
```