HW 2 – Database

3.2

Since a relation is formally outlined as a group of tuples, if the cardinality is 22, there should be 22 distinct tuples.

3.4

The primary key is the key chosen by the DBA from the cluster group of candidate keys, all of that unambiguously determine a tuple. A superkey may be a set of properties that includes a key.

3.5

Illustrations of non-candidate keys incorporate such as: {title}, {age}. (Note that {gpa} cannot be pronounced as a non-candidate key as the results this prove that despise the fact that common knowledge demonstrated to us clearly that more than one understudy might have the same review point average.)

It cannot be determined that a key of a correlation given only one illustration of the correlation. The reality that the occasion is "legal" is insignificant. A candidate key, as characterized here, is a key, not something only could be a key.
The illustration appeared is fair one conceivable "snapshot" of the correlation. At other times, the same correlation may have an illustration (or preview) that contains a completely diverse variety of tuples, and we cannot make projections around those illustrations based as it were upon the illustrations that we are given.

3.6

Because foreign key restriction is an explanation of the shape that one or more areas of a correlation, for example R, together advert to a second of correlation, for example S. That's, the values in these areas of a tuple in R are either invalid or extraordinarily distinguish a few tuples in S. As a consequence, these areas of R are defined to be a (candidate or essential) key. As demonstrated, an understudy, extraordinarily distinguished by a **sid**, enlisted in a course must also be matriculated within the educational institution student database (say, in a connection called Understudies). Subsequently, the **sid** of a lawful passage within the Lesson_Enrollment connection must coordinate an existing **sid** within the Understudies connection.

Foreign key restrictions are important because they provide safe alternatives for insuring the integrity of data. Users are alerted/warned when they try to do something that does not seem logical. This can help reduce errors in data-entry or application programs.

Referential integrity is defined as all foreign key restrictions are obligated.

3.7

There is not a purpose for a foreign key restriction (FKC) on the Rooms, Courses, Faculties, or Student connections. These are the foremost fundamental relations and has to be independent. Special requirements must be provided to entering information into these base connections.

In the Enrolled connection, **sid** and **cid** are both required to have FKCs positioned on them. (Real students should be participating in realistic courses.) Moreover, since real teachers must teach realistic subjects, both the **fid** and the **cid** fields in the Teaches connections must have FKCs. Last but not the least, Meets_In must place FKCs on both the **cid** and **rno** fields.

It would probably be intelligent to start a few other restrictions on these DBMS: the length of **fid**, **sid**, and **cid** could be systematized; checksums could be included to these identification numeric subjects; limits could be placed on the size of the numbers entered into the salary fields, credits, and capacity; an specified type should be allocated to the field (limiting a student from getting a grade of G, including other materials); etc.

3.12

1. CREATE TABLE Teaches (ssn CHAR(10), coursed INTEGER, semester CHAR(10), Primary Key (ssn, courseid, semester), Foreign KEY (ssn) References Course) Foreign Key (semester) references Semester)

CREATE TABLE Course (courseId INTEGER, PRIMARY KEY (courseId))

2. CREATE TABLE Teaches (ssn CHAR(10), courseId INTEGER, semester CHAR(10, PRIMARY KEY (ssn, courseID), FOREIGN KEY (ssn) REFRENCES Professor, FOREIGN KEY (courseId) References Course)

3. The participation constraint cannot be captured using only primary and foreign key constraints because we cannot ensure that every entry in Professor has an entry in Teaches

4. CREATE TABLE Professor_teaches (ssn CHAR(10), courseId INTEGER, semester CHAR(10), PRIMARY KEY (ssn), FOREIGN KEY (courseId) REFERENCES Course)

CREATE TABLE Course (courseId INTEGER, PIRMARY KEY (courseId))

5. CREATE TABLE Professor_teaches (ssn CHAR (10), courseId INTEGER, semester CHAR (10), PRIMARY KEY (ssn), FOREIGN KEY (courseId) References Course)

6. CREATE TABLE Teaches (gid INTEGER, courseID INTEGER, semester CHAR(10), PRIMARY KEY (gid, courseId), FOREIGN KEY (gid) REFERENCES Group, FOREIGN KEY (courseId) REFERENCES Course)

CREATE TABLE MemberOf (ssn CHAR(10), gid INTEGER, PRIMARY KEY (ssn, gid), FOREIGN KEY (ssn) REFERENCES Professor, FOREIGN KEY (gid) REFERENCES Group)

CREATE TABLE Course (courseId INTEGER, PRIMARY KEY (courseId))

CREATE TABLE Group (gid INTEGER, PRIMARY KEY (gid))

CREATE TABLE Professor (ssn CHAR(10), PRIMARY KEY (ssn))

3.15

The following SQL statements create the corresponding relations.

1. CREATE TABLE Musicians (ssn CHAR(10), name CHAR(30), PRIMARY KEY (ssn))
2. CREATE TABLE Instruments (instrId CHAR(10), dname CHAR(30), key CHAR(5), PRIMARY KEY (instrId))
3. CREATE TABLE Plays (ssn CHAR(10), instrId INTEGER, PRIMARY KEY (ssn, instrId), FOREIGN KEY (ssn) REFERENCES Musicians, FOREIGN KEY (instrId) REFERENCES Instruments)
4. CREATE TABLE Songs Appears (songId INTEGER, author CHAR(30), title CHAR(30), albumIdentifier INTEGER NOT NULL, PRIMARY KEY (songId), FOREIGN KEY (albumIdentifier) References Album Producer,
5. CREATE TABLE Telephone Home (phone CHAR(11), address CHAR(30), PRIMARY KEY (phone), FOREIGN KEY (address) REFERENCES Place,
6. CREATE TABLE Lives (ssn CHAR(10), phone CHAR(11), address CHAR(30), PRIMARY KEY (ssn, address), FOREIGN KEY (phone, address) References Telephone Home, FOREIGN KEY (ssn) REFERENCES Musicians)
7. CREATE TABLE Place (address CHAR(30))
8. CREATE TABLE Perform (songId INTEGER, ssn CHAR(10), PRIMARY KEY (ssn, songId), FOREIGN KEY (songId) REFERENCES Songs, FOREIGN KEY (ssn) REFERENCES Musicians)
9. CREATE TABLE Album Producer (albumIdentifier INTEGER, ssn CHAR(10), copyrightDate DATE, speed INTEGER, title CHAR(30), PRIMARY KEY (albumIdentifier), FOREIGN KEY (ssn) REFERENCES Musicians)
10. CREATE TABLE Musicians (ssn CHAR(10), name CHAR(30), PRIMARY KEY (ssn))
11. CREATE TABLE Instruments (instrId CHAR(10), dname CHAR(30), key CHAR(5), PRIMARY KEY (instrId))

12. CREATE TABLE Plays (ssn CHAR(10), instrId INTEGER, PRIMARY KEY (ssn, instrId), FOREIGN KEY (ssn) REFERENCES Musicians, FOREIGN KEY (instrId) REFERENCES Instruments)
13. CREATE TABLE Songs Appears (songId INTEGER, author CHAR(30), title CHAR(30), albumIdentifier INTEGER NOT NULL, PRIMARY KEY (songId), FOREIGN KEY (albumIdentifier) References Album Producer,
14. CREATE TABLE Telephone Home (phone CHAR(11), address CHAR(30), PRIMARY KEY (phone), FOREIGN KEY (address) REFERENCES Place,
15. CREATE TABLE Lives (ssn CHAR(10), phone CHAR(11), address CHAR(30), PRIMARY KEY (ssn, address), FOREIGN KEY (phone, address) References Telephone Home, FOREIGN KEY (ssn) REFERENCES Musicians)
16. CREATE TABLE Place (address CHAR(30))
17. CREATE TABLE Perform (songId INTEGER, ssn CHAR(10), PRIMARY KEY (ssn, songId), FOREIGN KEY (songId) REFERENCES Songs, FOREIGN KEY (ssn) REFERENCES Musicians)
18. CREATE TABLE Album Producer (albumIdentifier INTEGER, ssn CHAR(10), copyrightDate DATE, speed INTEGER, title CHAR(30), PRIMARY KEY (albumIdentifier), FOREIGN KEY (ssn) REFERENCES Musicians)

3.16

The following SQL statements create the corresponding relations.

1. CREATE TABLE Expert (

ssn CHAR(11),

model_no INTEGER,

PRIMARY KEY (ssn, model_no),

FOREIGN KEY (ssn) REFERENCES Technician,

FOREIGN KEY (model_no) REFERENCES Models)

The participation constraint cannot be captured in the table.

2. CREATE TABLE Models (

Model_no INTEGER,

capacity INTEGER,

weight INTEGER,

PRIMARY KEY (model_no))

3. CREATE TABLE Employees (

ssn CHAR(11),

union_mem_no INTEGER,

PRIMARY KEY (ssn))

4. CREATE TABLE Technician_emp (

ssn CHAR(11),

name CHAR(20),

address CHAR(20),

phone_no CHAR(14),

PRIMARY KEY (ssn),

FOREIGN KEY (ssn)

REFERENCES Employees

ON DELETE CASCADE)

5. CREATE TABLE Traffic_control_emp (

ssn CHAR(11),

exam_date DATE,

PRIMARY KEY (ssn),

FOREIGN KEY (ssn)

REFERENCES Employees

ON DELETE CASCADE)

6. CREATE TABLE Plane_Type (

Reg_no INTEGER,

Model_no INTEGER,

PRIMARY KEY (reg_no),

FOREIGN KEY (model_no) REFERENCES Models)

7. CREATE TABLE Test_info (

FAA_no INTEGER,

ssn CHAR(11),

reg_no INTEGER,

hours INTEGER,

date DATE,

score INTEGER,

PRIMARY KEY (ssn, reg_no, FAA_no),

FOREIGN KEY (reg_no) REFERENCES Plane_Type,

FOREIGN KEY (FAA_no) REFERENCES Test,

FOREIGN KEY (ssn) REFERENCES Employees)

8. CREATE TABLE Test (

FAA_no INTEGER,

name CHAR(10),

max_score INTEGER,

hours INTEGER,

date DATE,

score INTEGER,

PRIMARY KEY (FAA_no))

9. The restriction that tests on a plane must be performed by a technician who is proficient on working on the model and can be expressed in SQL as shown below:

CREATE TABLE Test_info (

FAA_no INTEGER,

ssn CHAR(11),

reg no INTEGER,

hours INTEGER,

date DATE,

score INTEGER,

PRIMARY KEY (ssn, reg no, FAA no),

FOREIGN KEY (reg_no) REFERENCES Plane_Type,

```
FOREIGN KEY (FAA_no) REFERENCES Test,
FOREIGN KEY (ssn) REFERENCES Technician_emp )
CONSTRAINT MODEL
CHECK ( SELECT * FROM Expert, Type
WHERE Expert.ssn = ssn AND
Expert.model_no = Type.model_no AND
Type.reg_no = reg_no )
```