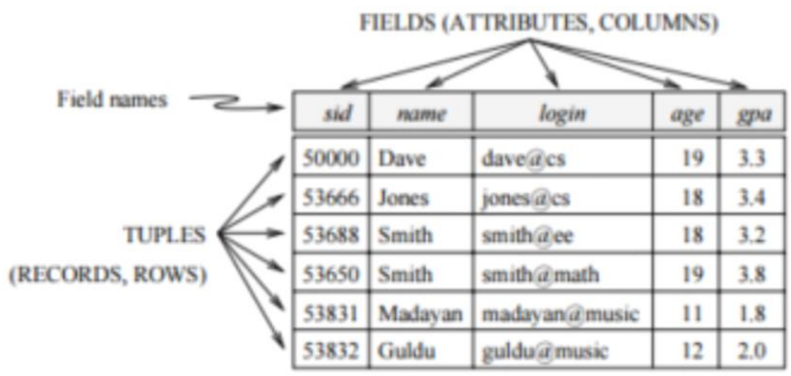
Vlad Predovic
CS 340
Due 10/27/2015

Assignment 3

[My website! --> http://web.engr.oregonstate.edu/~predoviv/CS340/hw2.php](http://web.engr.oregonstate.edu/~predoviv/CS340/hw2.php)

Exercise 3.2 How many distinct tuples are in a relation instance with cardinality 22?
The cardinality of a relation instance is the number of tuples in it. **22**

Exercise 3.5 Consider the instance of the Students relation shown in Figure 3.1.



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.1 An Instance *S*1 of the Students Relation

1. Give an example of an attribute (or set of attributes) that you can deduce is not a candidate key, based on this instance being legal

Name and age. Both of these are repeated and so cannot be candidate keys. All others can if we assume that there will be no more tuples in the table since every value is unique.

2. Is there any example of an attribute (or set of attributes) that you can deduce is a candidate key, based on this instance being legal?

Logically, the ***sid*** would be assumed to be a candidate key as the sole purpose of an id within a table is as an identifier.

Exercise 3.8 Answer each of the following questions briefly. The questions are based on the following relational schema:

Emp (*eid: integer*, ename: string, age: integer, salary: real)

Works (*eid: integer, did: integer*, pcttime: integer)

Dept (*did: integer*, dname: string, budget: real, managerid: integer)

1. *Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?*

   If we consider the 'Works' table I created in problem 2 here we see a relation where did is used as a foreign key for the Dept relation. Logically thinking, an employee cannot work for a department that does not exist so if a Dept tuple were to be deleted, all the relationships built with employees would need to be destroyed as well. Other possible results are disallowing the deletion of a dept tuple with associated employees or associating the employees to another dept automatically.

2. *Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.*

   CREATE TABLE

Emp (eid INTEGER, ename CHAR (15), age INTEGER, salary REAL, PRIMARY KEY (eid))

Works (eid INTEGER, did INTEGER, pcttime INTEGER, PRIMARY KEY (eid, did), FOREIGN KEY (did) REFERENCES Dept, FOREIGN KEY (eid) REFERENCES Emp, ON DELETE CASCADE)

Dept (did INTEGER, dname CHAR (20), budget REAL, managerid INTEGER, PRIMARY KEY (did), FOREIGN KEY (managerid) REFERENCES Emp, ON DELETE SET NULL)

3. *Define the Dept relation in SQL so that every department is guaranteed to have a manager.*

   CREATE TABLE

Dept (did INTEGER, dname CHAR (20), budget REAL, managerid INTEGER NOT NULL, PRIMARY KEY (did), FOREIGN KEY (managerid) REFERENCES Emp, ON DELETE SET NULL)

4. *Write an SQL statement to add John Doe as an employee with eid = 101, age = 32 and salary = 15,000.*

INSERT INTO Emp VALUES (101, 'John Doe', 32 15000); *inserting into all columns, names can be omitted.

5. *Write an SQL statement to give every employee a 10 percent raise.*

Update Emp P SET P.salary = P.salary * 1.1

6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

DELETE FROM Dept T WHERE T.dname = 'Toy'

Since we are deleting a department from the Dept table, all tuples in the Works table with the 'did' foreign key corresponding to the 'toy' department will be deleted since you cannot work in a department that does not exist.

Exercise 3.15 Consider the Notown database from Exercise 2.5. Show the SQL statements for creating relations corresponding to the entity sets and relationship sets in your design. Identify any constraints in the ER diagram that you are unable to capture in the SQL statements and briefly explain why you could not express them.

Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat,
E-flat).
Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
Each song recorded at Notown has a title and an author.
Each musician may play several instruments, and a given instrument may be played by several musicians.
Each album has a number of songs on it, but no song may appear on more than one album.
Each song is performed by one or more musicians, and a musician may perform a number of songs.
Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course

1. CREATE TABLE Musicians (ssn CHAR (9), name CHAR (25), PRIMARY KEY (ssn))

2. CREATE TABLE Lives (ssn CHAR (9), address CHAR (40), Phone_no CHAR (10),
PRIMARY KEY (ssn, address), FOREIGN KEY (ssn, Phone) REFERENCES
HOME_TELEPHONE, FOREIGN KEY (ssn) REFERENCES Musicians

*We don't need a third table because it is a 1 to many
3. CREATE TABLE Place (address CHAR (40))

4. CREATE TABLE Home_Telephone (phone_no CHAR (10), address CHAR (40), PRIMARY
KEY (Phone_no), FOREIGN KEY (address) REFERENCES Place

5. CREATE TABLE Plays (instrID CHAR (10), ssn CHAR (9), PRIMARY KEY (instrID, ssn),
FOREIGN KEY (instrID) REFERENCES Instrument, FOREIGN KEY (ssn) REFERENCES
Musicians

6. CREATE TABLE Instrument (instrID CHAR (10), dname CHAR (20), key CHAR (5), PRIMARY KEY (instrID))

7. CREATE TABLE Song_App (songID CHAR (10), title CHAR (25), author CHAR (25), albumIdentifier Char (10),
PRIMARY KEY (songID), FOREIGN KEY (albumIDENTIFIER) REFERENCES Album

8. CREATE TABLE Album_Produces (albumIdentifier CHAR (10), Format CHAR (5), uid INTEGER, title CHAR
(25), copyrightDt DATE, ssn CHAR (9), PRIMARY KEY (albumIdentifier), FOREIGN KEY
(ssn) REFERENCES Musicians)

9. CREATE TABLE Perform (ssn CHAR (9), songID CHAR (10), PRIMARY KEY (ssn, songID), FOREIGN KEY
(ssn) REFERENCES Musicians, FOREIGN KEY (songID) REFERENCES Songs)

(40 pts) Write SQL statements to create corresponding relations to the ER diagram you have created for you real-world application from Assignment #2. If your translation cannot capture some constraints in the ER diagram, explain why. Write down the list of tables, their attributes, candidate keys and foreign keys, as well as options for insert/update/delete in the same PDF file that also contains (a-d). Note this is not the same as the template. Then, in the same PDF, include all the SQL statements used to create these tables. The tables must be listed in the correct order of creation. Note that, due to foreign key constraints, some tables must be in place before the a table is created

Employees (*EID: Integer*, Name: char (20), password: char (20), PhoneNumber: char (10), adminPrivs: Boolean)

Candidate Keys: EID, PhoneNumber

Work_On (*EID: integer, WID: integer*, HrsWorked: integer)

Candidate Keys: EID, WID

Foreign Keys: EID, WID

Work (*WID: Integer*, DateRequested: Date, CompletionDate: Date, LeadEmployee: Char (20), WorkType: Char (6), CID: Integer)

Candidate Keys: WID

Foreign Keys: CID

Service (*WID: Integer*, ServiceType: Char (10), ServiceLocation: Char (50), ServiceDate: Date)

Candidate Keys: WID

Job (*WID: Integer,* DateStarted: Date, JobType: Char (10), DocID: Integer)

Candidate Keys: WID

Foreign Keys: DocID

Documentation (*DocID: Integer,* DateRecieved: Date, MediaType: Char (20), Description: Char 50))

Candidate Keys: DocID, Description

Clients (*CID: Integer*, ClientName: Char (25), Cpassword: Char (20), Email: Char (15), Address: Char (50), Phone: Char (10))

Candidate Keys: CID, ClientName, Email

1. CREATE TABLE EMPLOYEES (EID INTEGER, Name CHAR(20), Password CHAR(20), PhoneNumber CHAR(10), adminPrivs TINYINT, PRIMARY KEY(EID))

2. CREATE TABLE Clients (CID INTEGER, ClientName CHAR(25), Cpassword CHAR(20), Email CHAR(15), Address CHAR(50), Phone CHAR(10), PRIMARY KEY (CID))

3. CREATE TABLE Work (WID INTEGER, DateRequested DATE, CompletionDate DATE, LeadEmployee CHAR (20), WorkType CHAR(6), CID INTEGER, PRIMARY KEY (WID), FOREIGN KEY (CID) REFERENCES Clients)

4. CREATE TABLE Work_On (EID INTEGER, WID INTEGER, HrsWorked INTEGER, PRIMARY KEY (EID, WID), FOREIGN KEY(EID) REFERENCES Employees, FOREIGN KEY(WID) REFERENCES Work)

5. CREATE TABLE Service (WID INTEGER, ServiceType CHAR(10), ServiceLocation CHAR(50), ServiceDate DATE, PRIMARY KEY (WID), FOREIGN KEY (WID) REFERENCES Work)

6. CREATE TABLE Documentation (DocID INTEGER, DateRecieved DATE, MediaType CHAR(20), Description CHAR(50), PRIMARY KEY (DocID))

7. CREATE TABLE Job (WID INTEGER, JobType CHAR(10), DateStarted DATE, PRIMARY KEY (WID), DocID INTEGER, FOREIGN KEY (WID) REFERENCES Work, FOREIGN KEY DocID REFERENCES Documentation)