**Name: Justin Senia**

**EID: E00851822**

**Date: 9/25/2017**

**Class: COSC 471 T 10:00am-11:50am**

**Assignment: Homework\_1**

**CH1**

1.3) List six major steps that you would take in setting up a database for a

particular enterprise.

1. Create a model containing all types of data required and relationships between data.
2. Create a document that lists the integrity constraints for the data.
3. Create a Schema for the model.
4. Create a document describing the Database on a physical level.
5. For frequently recurring tasks, define a view for each and write programs to deal with every individual task occurrence.
6. The final step would be to create the Database, then Initialize it.

1.8) List four significant differences between a file-processing system and a

DBMS.

1. DBMS has very minimal to no Data redundancy & inconsistency, while a file processing systems typically can’t help having a considerable amount of data redundancy & inconsistency due to it’s rather piecemeal design characteristics.
2. DBMS handles atomicity in a fairly efficient/effective way to ensure that there are very few issues, while it is very hard to make sure atomicity is handled in file processing systems which can lead to issues with data inconsistency.
3. DBMS is significantly more secure as the systems are typically constructed so that end users don’t have absolute control over the entire system, one can implement authentication protocols and even have data backups in case of data corruption or other various problems (and all of these precautions are fairly easy to implement over the whole database itself). While file processing systems are not very secure due to most application programs being written by multiple programmers over a long period of time, consistency and interconnectivity of said protocols are very difficult to implement across entire systems, which causes many security loopholes and various issues
4. DBMS gives many ways to access data in an efficient and user friendly way, while in file processing systems accessing data can be (and usually is) both inefficient and difficult to do.

1.9) Explain the concept of physical data independence, and its importance in

database systems.

1. The concept essentially allows for one to change the physical schema without having to rewrite application programs after the fact to accommodate the changes made. This is quite important as it allows for seamless modification of schema when databases need to grow beyond it’s original scope to encompass new fields and required data types.

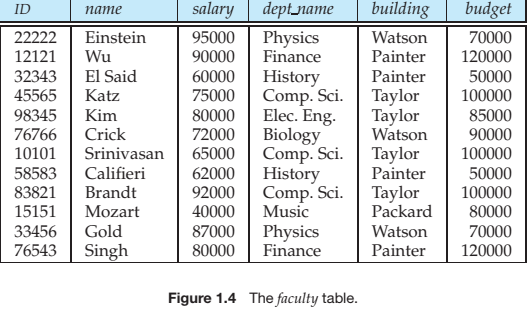
1.11) List at least two reasons why database systems support data manipulation

using a declarative query language such as SQL, instead of just providing

a a library of C or C++ functions to carry out data manipulation.

1. SQL actually prevents those who use the language from misrepresenting and/or improperly accessing data, while C or C++ lack the functional safeguards to prevent these problems which can cause issues for the DB’s relational integrity.
2. SQL is also designed for ease of use by the programmer specifically for Databases, this language was developed over many different versions (each version improving upon and offering more tools than the last), so that it’s streamlined for it’s intended use. Representational independence is a good example of one of the differences between the two options, as SQL won’t let you expose representation, while languages like C or C++ will.

1.12) Explain what problems are caused by the design of the table in Figure 1.4.



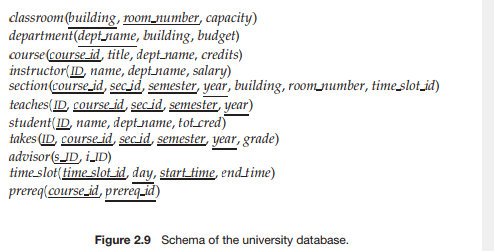
1. The first problem is that there is a significant amount of redundant information due to too many attributes being included in a single table. department names, buildings and budgets have a significant amount of repetition in each of their fields, so it would be better to split the “faculty” table into two tables: “Instructor” and “Department”, so that budget information and buildings associated with departments would only have to be listed once in it’s own table (which is significantly easier to update both integrity and efficiency wise) and foreign keys could be created to supply the information desired in a much more efficient relational form.
2. (as supporting information for point #1) It’s unnecessary to display budget on this particular table, not just because it has many repeated values, but because it’s relatively unrelated to the professors working in the department, it has nothing to do with ID’s and while it might have something to do with salary, it would appear that overall departmental budget has more to do with non-salary based spending potential, as the teacher salaries often exceed departmental funding.
3. We are also unable to represent new data on this field in particular places, if a new department or a new building is added to the school, unless there is a teacher assigned to one of those new places it won’t be represented in the table, which necessitates the splitting of this table into the two tables as previously discussed.
4. I would also argue that there needs to be both a “firstName” attribute as well as a “lastName” attribute on this table instead of the current “name” attribute, as the current “name” attribute is ambiguous as to whether or not it’s referring to the first name, last name or even one’s whole name. Therefore for clarity’s sake it would be very important to break that up into two parts, in order to create a more complete set of information as well as to avoid potential future conflicts as having only a single name identifier will most likely lead to having a large number of duplicate pieces of data, which is confusing, redundant and problematic in many ways.

**CH2**

2.6) Consider the following expressions, which use the result of a relational

algebra operation as the input to another operation. For each expression,

explain in words what the expression does.



*a. Sigma.pngyear≥2009(takes) Join.pngstudent*

It selects all rows from the Table “takes” that have a year value greater than or equal to 2009, and then uses a natural join to affix any additional student information (attributes found in the “student table”) to the result of the first select operation by utilizing identical relationships shared by both the “student” and “takes” tables (the “ID” attribute was used in this case) and then creates a new table out of the combination of information. The resulting table will contain: ID, name, dept\_name, tot\_cred, course\_id, sec\_id, semester, year, grade

*b. Sigma.pngyear≥2009(takes Join.pngstudent)*

This gives the same result as the question above but it changes the order of operations so that the natural join happens before the select operation.

*c. thingy.pngID,name,course\_id (student Join.pngtakes)*

The “student” Table is naturally joined to the “takes” table, which as described above, combines multiple tables based on shared values to create a larger table that encompasses both sets of relationally defined tables. Then a projection is made on the newly combined table, to create another final table that consists of every data pairing (for each individual student) which includes: ID, name, course\_id .

2.9) (Copy the Banking database schema from fig. 2.15 and underline your choice of primary key for each relation and italicize the foreign keys in the referencing relation only.)

branch (branch\_name, branch\_city, assets)

customer (customer\_name, customer\_street, customer\_city)

loan (loan\_number, *branch\_name*, amount)

borrower (*customer\_name*, *loan\_number*)

account (account\_number, *branch\_name*, balance)

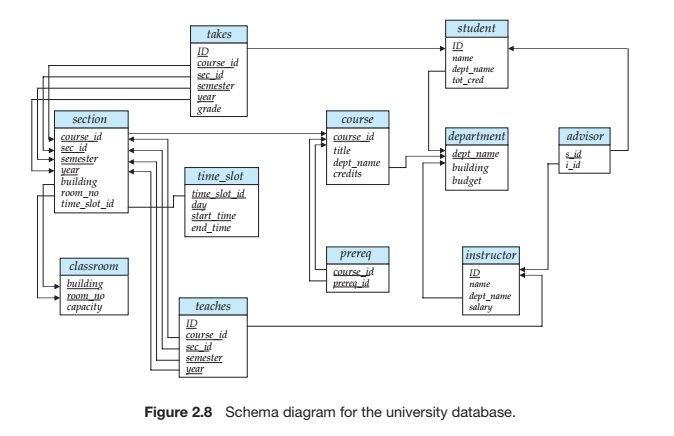
depositor (*customer\_name*, *account\_number*)

2.10) Consider the advisor relation shown in Figure 2.8, with s id as the primary

key of advisor. Suppose a student can have more than one advisor. Then,

would s id still be a primary key of the advisor relation? If not, what should

the primary key of advisor be?



1. It would no longer work as a primary key to the Advisor Table because all of it’s values need to be unique for it to function as a primary key, so instead you could make the combination of both “s\_id” and “i\_id” become the primary key for the advisor table.

2.13) Consider the bank database of Figure 2.15. Give an expression in the rela-

tional algebra for each of the following queries:

a. Find all loan numbers with a loan value greater than $10,000.

thingy.pngloan\_number(Sigma.pngamount > 10000 (loan))

b. Find the names of all depositors who have an account with a value

greater than $6,000

thingy.pngcustomer\_name(Sigma.pngbalance > 6000 (depositor Join.pngaccount))

c. Find the names of all depositors who have an account with a value

greater than $6,000 at the “Uptown” branch.

thingy.pngcustomer\_name(Sigma.pngbalance > 6000 AND branch\_name = ‘Uptown’ (depositor Join.pngaccount))

2.16) Differentiate between the following:

1. Superkey vs. candidate key

A superkey is an attribute that has unique values for every one of it’s rows, it’s used to uniquely identify tuples (or set of related information), only one superkey may be chosen per relation. A candidate key is any column (or sometimes multiple columns) that qualify to be a superkey but may or may not be chosen as the table’s single superkey.

1. Primary key vs. foreign key

A Primary Key is either a single column or collection of multiple columns that are used to uniquely identify a row in a table. A Foreign Key is either a single field or multiple fields that are used to match values on a secondary table’s Primary Key.

1. Schema vs. instance

An Instance refers to the data stored in a database at any one given point in time, while “schema” refers to the actual design of a database on physical, logical and view levels.

1. Procedural vs. non-procedural query languages

A non procedural query language usually only involves defining input and output, while procedural languages also involve the intermediary steps in between defining input and output.

1. Selection vs. projection operations

Selection operations involve retrieving rows (i.e. tuples) from a database, while projection operations retrieve columns (i.e. attributes)

**CH3**

person (driver\_id, name, address)

car (license, model, year)

accident (report\_number, date, location)

owns (driver\_id, license)

participated (report\_number, license, driver\_id, damage\_amount)

Fig 3.18

3.25) 1. Write SQL DDL statements corresponding to the schema in Fig. 3.18 (Insurance database) in the textbook. Make any reasonable assumptions about the data types and be sure to declare primary and foreign keys.

CREATE TABLE person (

driver\_id INTEGER,

name VARCHAR(20) NOT NULL,

address VARCHAR(50),

PRIMARY KEY (driver\_id)

);

CREATE TABLE car (

license VARCHAR(10),

model VARCHAR(20),

year INTEGER,

PRIMARY KEY (license)

);

CREATE TABLE accident (

report\_number INTEGER,

date VARCHAR(10),

location VARCHAR(50),

PRIMARY KEY (report\_number)

);

CREATE TABLE owns (

Driver\_id VARCHAR(20),

License VARCHAR(20)

PRIMARY KEY (driver\_id, license),

FOREIGN KEY (driver\_id) REFERENCES person,

FOREIGN KEY (license) REFERENCES car

);

CREATE TABLE participated(

Report\_number INTEGER

License VARCHAR(20)

Driver\_id VARCHAR(20)

Damage\_amount INTEGER,

PRIMARY KEY (report\_number, license)

FOREIGN KEY (license) REFERENCES car,

FOREIGN KEY (report\_number) REFERENCES accident,

FOREIGN KEY (driver\_id) REFERENCES person

);

2. Write SQL DDL/DML statements to do the following:

* 1. Alter any one table to include a new, meaningful attribute.

ALTER TABLE car ADD make VARCHAR(20);

* 1. Insert one tuple each into each of the tables. Make sure the data follows all integrity constraints.

INSERT INTO person (

driver\_id, name, address

)

VALUES (546346, ‘Jimmy’, ‘313 parkway dr’);

INSERT INTO car (

license, model, year

)

VALUES (‘A101010DRE’, ‘focus’, 2009);

INSERT INTO accident (

report\_number, date, location

)

VALUES (05923507, ‘03/21/2014’, ‘311 wildhurst ave’);

INSERT INTO owns (

driver\_id, license

)

VALUES (546346, ‘A101010DRE’);

INSERT INTO participated (

report\_number, license, driver\_id, damage\_amount

)

VALUES (05923507, ‘A101010DRE’, 546346, 66 )

* 1. Delete one tuple from any table (do not delete all tuples).

DELETE FROM person

WHERE name = ‘Jimmy’;

* 1. Update any one tuple from any table.

UPDATE person

SET name = ‘Farnsworth’

WHERE driver\_id = “234234”;