INFO 210: Database Management Systems

Midterm
solutions

Descriptions

- In this provided solutions, characters in red color mean the key words/phrases/sentences to gain points.

- Other characters not in red color are my detailed explanations to help your better understanding --- and among which, the **bold** or <u>underlined</u> characters are used to emphasis important knowledge or understanding for the problem questions --- but which are not required for points.
  - Similar explanations with these many details when you were answering the questions in the exams or homeworks are definitely encouraged, but not a must to get points for your correct answers.

- If you get points off for any questions, you should study the concepts and knowledge again, by referring to the chapters of textbook, solutions provided of homework assignments, and the corresponding topics and examples of the lecture materials (for each solution of questions, I have briefly reminded you some references related to the involved concepts and knowledge)

Problem 1 (25 points)

- Consider a **Schema** of relation R:

  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

(1a) (15 points) Given a create table statement below for relation R.

```
create table R (
        A number,
        B varchar(16) unique,
        C char(12),
        D number,
        E char(1) not null,
        unique (B, D)
);
```

Is this a correct statement to make the above instance of R valid? List **two** evidences that support your conclusion. (**what are** correct or incorrect, and **why**).

Tips: Think about concepts and usages of primary key, unique, not null etc.

No, the statement is incorrect, as it will make the instance invalid --- the condition that has given to us in the problem is that the instance is a valid instance. In another word, any conflict between the statement and the facts shown in the instance will make the statement incorrect. (main concepts and knowledge involved for Q1a can be referred to Lecture 2)

The reasons making the statement incorrect:

1. the keyword "unique" conflicts with the instance.
2. the keyword "not null" conflicts with the instance.
3. Every relation MUST have exactly ONE primary key!

Problem 1 (25 points)

- Consider a **Schema** of relation R:
  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

(1a) (15 points) Given a create table statement below for relation R.

```
create table R (
        A number,
        B varchar(16) unique,
        C char(12),
        D number,
        E char(1) not null,
       unique (B, D)
);
```

Is this a correct statement to make the above instance of R valid? List **two** evidences that support your conclusion. (**what are** correct or incorrect, and **why**).

Tips: Think about concepts and usages of primary key, unique, not null etc.

1. the keyword "unique" conflicts with the instance.

The clause in the statement writes " B varchar(16) unique, " where "varchar" indicates the data type for the attribute B (which corresponds with fact that the values in B is actually character, and also corresponds with the Schema "B: string "), and the keyword "unique" is requiring "values in the column B must be unique for different tuples; in another word, no two tuples can agree on values for the attribute B". However, you could see values in column B of the instance is not unique, for example, tuple 2 and 3 all possess a same charac value of "Bob" in B.

2. the keyword "not null" conflicts with the instance.

The clause writes "E char(1) not null," where "char" indicates the data type for the attribute E (which corresponds with fact that the values in E is represented by one character type of data (Y/N), and also corresponds with the Schema "E: string"), and the keyword "not null" is requiring "values in the column E must NOT be empty for different tuples; in another word, each tuple MUST have a value". However, you could see some values in column E is empty(null), for example, tuple 2, and 5 possess a same value of "18" in E.

3. Every relation MUST have exactly ONE primary key! (This rule is re-emphasized many times in Lect 2 page 21, 22, etc.). However for this statement, you could not find the keyword "primary key" on any attribute. To choose which one as THE primary key (A, or C, or other options like a combination are all qualified to be chosen as THE primary key) depends on the case.

Problem 1 (25 points)

- Consider a **Schema** of relation R:
  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

(1a) (15 points) Given a create table statement below for relation R.

```
create table R (
        A number,
        B varchar(16) unique,
        C char(12),
        D number,
        E char(1) not null,
        unique (B, D)
);
```

Is this a correct statement to make the above instance of R valid? List **two** evidences that support your conclusion. (**what are** correct or incorrect, and **why**).

Tips: Think about concepts and usages of primary key, unique, not null etc.

1. the keyword "`unique`" conflicts with the instance.

The clause in the statement writes "`B varchar(16) unique,`" where "`varchar`" indicates the data type for the attribute B (which corresponds with fact that the values in B is actually character, and also corresponds with the Schema "`B: string `"), and the keyword "`unique`" is requiring "values in the column B must be unique for different tuples; in another word, no two tuples can agree on values for the attribute B". However, you could see values in column B of the instance is not unique, for example, tuple 2 and 3 all possess a same charac value of "Bob" in B.

2. the keyword "`not null`" conflicts with the instance.

The clause writes "`E char(1) not null,`" where "`char`" indicates the data type for the attribute E (which corresponds with fact that the values in E is represented by one character type of data (Y/N), and also corresponds with the Schema "`E: string`"), and the keyword "`not null`" is requiring "values in the column E must NOT be empty for different tuples; in another word, each tuple MUST have a value". However, you could see some values in column E is empty(null), for example, tuple 2, and 5 possess a same value of "18" in E.

3. Every relation MUST have exactly ONE primary key! (This rule is re-emphasized many times in Lect 2 page 21, 22, etc.). However for this statement, you could not find the keyword "`primary key`" on any attribute. To choose which one as THE primary key (A, or C, or other options like a combination are all qualified to be chosen as THE primary key) depends on the case.

Problem 1 (25 points)

- Consider a **Schema** of relation R:
  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

(1a) (15 points) Given a create table statement below for relation R.

```
create table R (
        A number,
        B varchar(16) unique,
        C char(12),
        D number,
        E char(1) not null,
       unique (B, D)
);
```

Is this a correct statement to make the above instance of R valid? List **two** evidences that support your conclusion. (**what are** correct or incorrect, and **why**).

Tips: Think about concepts and usages of primary key, unique, not null etc.

Other comments:

I: in the statement, the clause "`unique (B, D)`" is correct. The reason is that although "`D number unique`," conflicts with the fact that D has same values that are not unique, the combination of (B, D) is unique. For example, tuple 1, 2, and 5 are all 18, but the combination values of (B, D) for tuple 1, 2, and 5 are (Ann, 18), (Bob, 18), and (Jim, 18), which are all distinguishable (unique) to tell they represent different tuples. So, the combination of "`(B, D)`" can be made "`unique`".

II: The statement writes "`C char(12),`" which is not wrong. As the Schema is "`C:string`", the value in column/attribute C is stored as ONE string in the relation (e.g. "111-11-1111") with the data type as "`char`", in another word, every number and hyphen is all treated as a character. "111-11-1111" for example has 11 character, but which doesn't make "`char(12)`" wrong. *Recall 2.3.2 section in textbook about Data Types, The type `char(n)` and `varchar(n)` all denotes a string of up to n characters, the difference is `char` is for fixed-length strings and strings with length shorter than n will be padded to make n characters, while `varchar` is for variable-length strings and stores the actual length of string.* The string length of 11 in C is not more than 12, and the extra spear character will be padded with space automatically. so "`char(12)`" is not wrong.

III: the clauses that do not have any keyword on (primary key, not null, unique, …) means they are not limited (constrained) to any condition, in another words, as long as their data type are writing correctly or making sense (when not explicitly described in Schema or instance or problem descriptions or business rules), they are good with any values, either unique or not, either null or not, etc. …

Problem 1 (25 points)

- Consider a **Schema** of relation R:

  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

(1b) (10 points) Consider the same Schema and relation R, write **a** create table statements for which would make the relation R **valid**.

two possible options for correct solutions

```
create table R (
   A      number primary key,
   B      varchar(16),
   C      char(12),
   D      number,
   E      char(1),
   unique (B, D)
);
```

OR

```
create table R (
   A      number,
   B      varchar(16),
   C      char(12)primary key,
   D      number not null,
   E      char(1),
   unique (B, E)
);
```

Problem 1 (25 points)

- Consider a **Schema** of relation R:

  R (A:integer, B:string, C:string, D:integer, E:string)

Below also shows you a **valid instance** of the relation R:

A	B	C	D	E
1	Ann	111-11-1111	18	Y
2	Bob	111-11-1112	18	
3	Bob	111-11-1113	22	N
4	Kate	111-11-1114	20	Y
5	Jim	111-11-1115	18	

```
create table R (
   A      number primary key,
   B      varchar(16),
   C      char(12),
   D      number,
   E      char(1),
   unique (B, D)
);
```

OR

```
create table R (
   A      number,
   B      varchar(16),
   C      char(12)primary key,
   D      number not null,
   E      char(1),
   unique (B, E)
);
```

Some comments:

I: **THE** (**on and only one**) primary key in a relation/table, can be composed by **ONE** of the attributes from that relation/table, **OR** by several attributes (as **a combination** of attributes) from that relation/table.

II: primary key **=** "unique" and also "not null", by default! so when you use **THE** primary key on attributes, you **should not write** "unique" or "not null" in addition.

III: Either A or C or (B,D) is qualified for being chosen as **THE** primary key, but remember primary key need to be **minimal**, in a another word, you should not be able to find any subset of a primary key that still qualified for being a primary key.

For example, if you choose the combination of (A,C) to be THE primary key, then you will find the A as one subset of (A,C), or C as a subset, is still qualified for being a primary key, hence, (A,C) should not be chosen as the primary key.
(Similarly: (A,B), (B,C),(A,C),… do not satisfy the minimal principle)

Problem 2 (25pts)

- Consider a **Schema** of a relation Dancers and the according **business rules** below:

  Dancers (name:string, dob:date, stage_name:string, company:string)
- No two dancers have the same combination of name and date of birth (dob).
- A dancer's name, a dob and a stage's name have to be specified for each dancer.
-
- *Tips: things "have to be specified" means they could not be empty in a relation; not empty doesn't necessarily mean to have different values; values of primary key can not be empty and must be different.*

(2a) (10 points) List all possible choices that could be set as the primary key? (could be zero or one or more).
(2b) (15 points) Write a create table statement that implements this relation.

2a) (name, dob)

According the business rule, **the only one solution** is choosing the combination of name and date of birth as THE primary key: (name, dob), since "No two dancers have the same combination of …"

2b)

```
create table Dancers (
    dancer_name     varchar(64),
    dob             date,
    stage_name      varchar(64) not null,
    company         varchar(64),
    primary key (name, dob)
);
```

Problem 2 (25pts)

- Consider a **Schema** of a relation Dancers and the according **business rules** below:

  Dancers (name:string, dob:date, stage_name:string, company:string)
- No two dancers have the same combination of name and date of birth (dob).
- A dancer's name, a dob and a stage's name have to be specified for each dancer.
-
- *Tips: things "have to be specified" means they could not be empty in a relation; not empty doesn't necessarily mean to have different values; values of primary key can not be empty and must be different.*

(2a) (10 points) List all possible choices that could be set as the primary key? (could be zero or one or more).
(2b) (15 points) Write a create table statement that implements this relation.

2b)

```
create table Dancers (
    dancer_name     varchar(64),
    dob             date,
    stage_name      varchar(64) not null,
    company         varchar(64),
    primary key (name, dob)
);
```

1. "`primary key (name, dob),`" because "**No two dancers have the same** combination of name and date of birth (dob)." so these two attributes as a combination can be chosen as THE primary key
2. "`stage_name varchar(64) not null,`" because "... a stage's name **have to be specified** for each dancer." so it must have a value and can not allow empty
3. "A dancer's name, a dob ... have to be specified for each dancer." so both "`name`" and "`dob`" should be "`not null`", but since primary key is "unique AND not null" **by default**, so **no need to write** "`not null`" on "`dancer_name`" or "`dob`" explicitly in addition, **otherwise will be redundant**.

Problem 3 (30 points)

- Consider two relation instances below, with the given **Schemas**. In each question below, write *a **relational algebra expression**,* and also ***show its results*** when the expression is executed with the given instances.
- Performers (name, genre)
- Nominations (name, category, year, is_winner)
- *Tips: There are many useful operators for **relational algebra**, include but not limited to for example: σ, π, X, ⋈, and ρ. It is always very helpful to break a long query into smaller pieces to express with the relational algebra, and then build up them all in appropriate order to get your ultimate query.*

Performers

name	genre
Coldplay	Rock
Amy Winehouse	R&B
Arizona Shakes	Rock
Frank Ocean	R&B
The Black Keys	Rock

Grammy_Nominations

name	category	year	is_winner
Arizona Shakes	New Artist	2013	no
Arizona Shakes	Rock Performance	2013	no
Frank Ocean	Album of the Year	2013	no
Frank Ocean	Rap/Sung Collaboration	2013	yes
Coldplay	Rock Performance	2013	no
Coldplay	Rock Album	2009	yes
Coldplay	Album of the Year	2009	no
Coldplay	Rock Performance	2012	no
The Black Keys	Rock Performance	2013	yes

Notice：

1. Each question requires **not only** the relational algebra expression, **but also** the **results** (**outputs**) generated by your relational algebra expression
2. It is easier to analyze long query by **breaking** it **into smaller pieces**, and use relational algebra to model each piece, later after which, reassembling them all together will achieve the whole final query.

Problem 3 (30 points)

Performers

name	genre
Coldplay	Rock
Amy Winehouse	R&B
Arizona Shakes	Rock
Frank Ocean	R&B
The Black Keys	Rock

Grammy_Nominations

name	category	year	is_winner
Arizona Shakes	New Artist	2013	no
Arizona Shakes	Rock Performance	2013	no
Frank Ocean	Album of the Year	2013	no
Frank Ocean	Rap/Sung Collaboration	2013	yes
Coldplay	Rock Performance	2013	no
Coldplay	Rock Album	2009	yes
Coldplay	Album of the Year	2009	no
Coldplay	Rock Performance	2012	no
The Black Keys	Rock Performance	2013	yes

(3a) (10 points) Select all the tuples in the table Performer that has genre as Rock, and show results of your expression.

Recall, selection σ is for selecting tuples, $\sigma_C(R)$ is to select tuples in table R that satisfy the condition C.

Then break down the query in Question 3a in pieces,

1. condition C is the "…that has genre as Rock", in another word, "genre" is "Rock", i.e. C is : genre = ' Rock';
2. The table R is the Performer;
3. The ultimate requirement is "selection";

finally assemble the above pieces all together will get you:

3a) Expression: $\sigma_{\text{genre} = \text{'Rock'}}(\text{Performer})$

Results:

name	genre
Coldplay	Rock
Arizona Shakes	Rock
The Black Keys	Rock

Problem 3 (30 points)

(3b) (20 points) List all the **genres** of 2009 Grammy winner, and **show results** of your expression.

Tips: genre and category are in different table.

Performers

name	genre
Coldplay	Rock
Amy Winehouse	R&B
Arizona Shakes	Rock
Frank Ocean	R&B
The Black Keys	Rock

Grammy_Nominations

name	category	year	is_winner
Arizona Shakes	New Artist	2013	no
Arizona Shakes	Rock Performance	2013	no
Frank Ocean	Album of the Year	2013	no
Frank Ocean	Rap/Sung Collaboration	2013	yes
Coldplay	Rock Performance	2013	no
Coldplay	Rock Album	2009	yes
Coldplay	Album of the Year	2009	no
Coldplay	Rock Performance	2012	no
The Black Keys	Rock Performance	2013	yes

Selection σ is for selecting tuples, $\sigma_C(R)$ is to select tuples in table R that satisfy the condition C.

Projection π is for listing columns, $\pi_{A1, A2...}(T)$ is to list columns in table T that have attribute names "A_1, A_2...".

Natural Join $\bowtie$ is to form a new table. $R \bowtie S$ is to form a new table composed by set of combinations of tuples from R and S that are equal on their common attribute.

Here in Question 3b in the query,

1. condition C is the "...2009 Grammy winner", in another word, C is : "is_winner" in "year 2009", i.e. C is : year=2009 AND is_winner='yes';
2. The table R is the Grammy_Nominations;
3. "genres" is the attribute in another table S which is Performers
4. in order to use information from both table, we need to form a new table T, by combining tuples from R and S that has common attribute (only "name" is the common attribute in the two table here.)
5. With the new table T, which contains information of genres on the names who "win 2009 Grammy", we will be able to list genres attribute/column.

So assemble the above all together will get you:

Results:

genre
Rock

3b) Expression:

$$\pi_{\text{genre}}(\sigma_{\text{year=2009 AND is\_winner='yes'}}(\text{Grammy\_Nomination}) \bowtie \text{Performers})$$

Problem 3 (30 points)

(3b) (20 points) List all the **genres** of 2009 Grammy winner, and **show results** of your expression.

Tips: genre and category are in different table.

Performers

name	genre
Coldplay	Rock
Amy Winehouse	R&B
Arizona Shakes	Rock
Frank Ocean	R&B
The Black Keys	Rock

Grammy_Nominations

name	category	year	is_winner
Arizona Shakes	New Artist	2013	no
Arizona Shakes	Rock Performance	2013	no
Frank Ocean	Album of the Year	2013	no
Frank Ocean	Rap/Sung Collaboration	2013	yes
Coldplay	Rock Performance	2013	no
Coldplay	Rock Album	2009	yes
Coldplay	Album of the Year	2009	no
Coldplay	Rock Performance	2012	no
The Black Keys	Rock Performance	2013	yes

3b) $\pi_{\text{genre}}(\sigma_{\text{year=2009 AND is\_winner='yes'}}(\text{Grammy\_Nomination}) \bowtie \text{Performers})$

Results:

genre
Rock

other options:

(Think about how these are generated by different ways in breaking down the query)

$\pi_{\text{genre}}(\ \sigma_{\text{year=2009 AND is\_winner='yes'}}(\ \text{Grammy\_Nomination} \bowtie \text{Performers}\ )\ )$

Problem 4 (20 points)

Consider we know a list of chefs and a list of restaurants. We also set following rules that: A chef is uniquely identified by a social security number (ssn), and is also described by a name and a cuisine in which he/she specializes. No restaurant with same name existed in a same city. (Or in another word, no restaurant can have a same combination of name and city). Each chef works in exactly one of the restaurants. And each restaurant can have at most one chef working at it.

(20 points) **Draw** an ER diagram that encodes the above business rules. You need to ***clearly mark all*** *key constraints ("1") and participation constraints ("═") on the ER diagram if they exist*.

"A chef is uniquely identified by a social security number (ssn), and is also described by a name and a cuisine in which he/she specializes. "

Chef (ssn: string, name:string, cuisine:string)

"No restaurant with same name existed in a same city. (Or in another word, no restaurant can have a same combination of name and city)."

Restaurant(name: string, city: string)

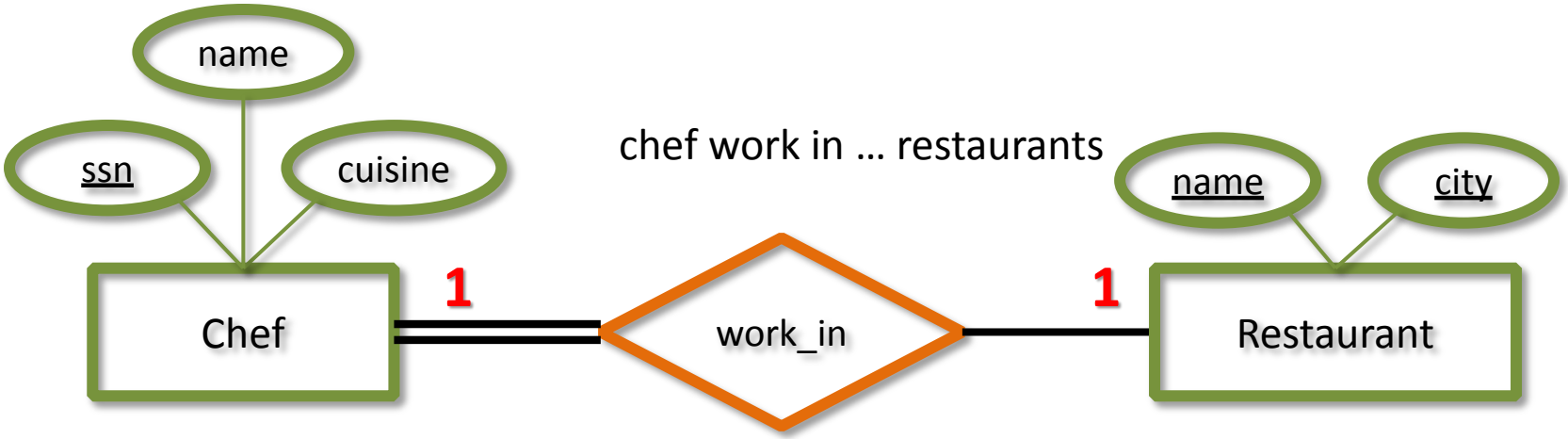
Problem 4 (20 points)

Consider we know a list of chefs and a list of restaurants. We also set following rules that: A chef is uniquely identified by a social security number (ssn), and is also described by a name and a cuisine in which he/she specializes. No restaurant with same name existed in a same city. (Or in another word, no restaurant can have a same combination of name and city). Each chef works in exactly one of the restaurants. And each restaurant can have at most one chef working at it.

(20 points) **Draw** an ER diagram that encodes the above business rules. You need to ***clearly mark all*** *key constraints ("1") and participation constraints ("═") on the ER diagram if they exist*.

Chef (ssn: string, name:string, cuisine:string)

Restaurant(name: string, city: string)



Each chef works in exactly one of the restaurants.

each restaurant can have at most one chef working at it