INFO 210: Database Management Systems

Homework 1 solutions

- This assignment covers the following topics
  - Set operation
  - The relational model

Problem 1: Sets

Let us denote by ***P*** the set of all **people**, by ***S*** the set of people who are **single**, by ***C*** the set of people who have **children**, by ***W*** the set of **women** and by ***M*** the set of **men**. For each question below, write down an expression that represents the set being described. Write exactly **one expression** for each question.

P --- set of all people,
S --- set of people who are single,
C --- set of people who have children,
W --- set of women, and
M --- set of men

The usual set operations:
union ∪
intersection ∩
set difference \

Problem 1: Sets

P --- set of all people,
S --- set of people who are **single**,
C --- set of people who **have children**,
W --- set of **women**, and
M --- set of **men**

The usual set operations:

union ∪
intersection ∩
set difference \

Write exactly one expression for each question.

- a) Single men who do not have children.
  - (S ∩ M) - C
- b) Single women who do not have children.
  - (S ∩ W) – C
- c) The number of women who have children.
  - | W ∩ C - S| | (W – S) ∩ C |
- d) Women who are either married with children, or single and do not have children.
  - ((W – S) ∩ C) ∪ ((W ∩ S) – C) ((W ∩ C) –S) U ((S ∩ W) –C)
- e) All possible married heterosexual couples, i.e., all possible pairings of married men and married women.
  - (M – S) X (W – S)

Problem 2 : Keys

- Consider an entity set *Person*, with attributes social security number (ssn), name, nickname, address, and date of birth (dob). Assume that the following conditions hold:
  - (1) no two persons have the same ssn;
  - (2) no two persons have the same combination of name, address, and dob.
  - Further, assume that all persons have an ssn, a name and a dob, but that some persons don't have a nickname nor an address.

- **(a)** List **all** candidate keys and **all** superkeys for this entity set.

- **(b)** Write a create table statement that defines a relation appropriate for this entity set.

Problem 2 : Keys

- Consider an entity set *Person*, with attributes social security number (**ssn**), **name**, **nickname**, **address**, and date of birth (**dob**).
- **(a)** List **all** candidate keys and **all** superkeys for this entity set.

There are 2 candidate keys: (**ssn**) and (**name, address, dob**).

First, we list all sets of attributes that include the first candidate key, (ssn), as a subset.

(ssn, name), (ssn, nickname), (ssn, address), (ssn, dob),

(ssn, name, nickname), (ssn, name, address), (ssn, name, dob),
(ssn, nickname, address), (ssn, nickname, dob),(ssn, address, dob),

(ssn, name, nickname, address), (ssn, name, nickname, dob),

(ssn, name, address, dob), (ssn, nickname, address, dob),

(ssn, name, nickname, address, dob)

There are exactly 4 + 6 + 4 + 1 = 15 such superkeys.

Problem 2 : Keys

- Consider an entity set *Person*, with attributes social security number (**ssn**), **name**, **nickname**, **address**, and date of birth (**dob**).
- **(a)** List **all** candidate keys and **all** superkeys for this entity set.

There are 2 candidate keys: (ssn) and (name, address, dob).

Next, we list all sets of attributes that include the second candidate key, (name, address, dob), as a subset. There are exactly 2+1=3 such superkeys.

(ssn, name, address, dob), (name, nickname, address, dob),

(ssn, name, nickname, address, dob)

Note that 2 superkeys (in bold) include both candidate keys as subsets.

Thus, there are a total of 15 + 3– 2 =16 superkeys for this relation.

Problem 2 : Keys

- Consider an entity set *Person*, with attributes social security number (ssn), name, nickname, address, and date of birth (dob). Assume that the following conditions hold:
  - (1) no two persons have the same ssn;
  - (2) no two persons have the same combination of name, address, and dob.
  - Further, assume that **all** persons **have** an ssn, a name and a dob, but that **some** persons **don't have** a nickname nor an address.

- **(b)** Write a create table statement that defines a relation appropriate for this entity set.

(Lengths of the varchar in this solution could be any the proper values)

```
create table Person (
        ssn char (11) primary key,
        name varchar(64),
        nickname varchar(32),
        address varchar(128),
        dob date,
        unique (name, address, dob)
);
```

Problem 2 : Keys

- Consider an entity set *Person*, with attributes social security number (ssn), name, nickname, address, and date of birth (dob). Assume that the following conditions hold:
  - (1) no two persons have the same ssn;
  - (2) no two persons have the same combination of name, address, and dob.
  - Further, assume that **all** persons **have** an ssn, a name and a dob, but that **some** persons **don't have** a nickname nor an address.

- **(b)** Write a create table statement that defines a relation appropriate for this entity set.

```
create table Person (
  ssn char (11) unique,
  name varchar(64),
  nickname varchar(32),
  address varchar(128),
  dob date,
  primary key (name, address, dob)
);
```

try to designate ssn as unique, and (name, address, dob) as a primary key.
But primary key requires "unique" and "not null" constraint **By Default**!

Problem 3: Schemas and instances

Consider an instance of the relation *Foo*. Below, we ask you to write three create table statements. Each create table statement must define a primary key.

Foo (A, B, C, D)

A	B	C	D
1	Ann	23	3
2	Bob	23	4
3	Joe	20	3
4	Bob	20	4

(a) Write two different create table statements for which the instance of *Foo* is legal. Note that taking the first statement and simply reordering columns does not give a different create table statement.

(b) Write a create table statement that would make the instance of *Foo* above illegal.

Problem 3: Schemas and instances

A	B	C	D
1	Ann	23	3
2	Bob	23	4
3	Joe	20	3
4	Bob	20	4

(a) Write **two different** create table statements for which the instance of *Foo* is **legal**. Note that taking the first statement and simply reordering columns does not give a different create table statement.

```
create table Foo (
  A        number,
  B        char(3),
  C        number,
  D        number,
  primary key (C,D)
);
```

```
create table Foo (
  B        char(3),
  A        number,
  D        number,
  C        number,
primary key (C,D)
);
```

```
create table Foo (
  A        number primary key,
  B        char(3),
  C        number,
  D        number
);
```

Several other solutions are possible for question (a) and (b).

Problem 3: Schemas and instances

A	B	C	D
1	Ann	23	3
2	Bob	23	4
3	Joe	20	3
4	Bob	20	4

(b) Write a create table statement that would make the instance of *Foo* above **illegal**.

```
create table Foo (
        A       number,
        B       char(3) primary key,
        C       number,
        D       number
);
```

Several other solutions are possible for question (a) and (b).

Problem 4: Foreign keys

- Consider relation schemas below, with primary keys underlined.
  - City (name, state, population, elevation)
  - State (name, region)
  - Mayor (name, city, state, party)
  - Governor (name, state, party)
- Suppose that the following business rules hold. Each mayor governs exactly one city. Each governor governs exactly one state.
- **(a)** Write create table statements that encode these relation schemas and business rules with the right foreign key constraints.
- **(b)** In what order would you drop these tables? Give **all** valid sequences.

Problem 4: Foreign keys

- Consider relation schemas below, with primary keys underlined.
  - City (name, state, population, elevation)
  - State (name, region)
  - Mayor (name, city, state, party)
  - Governor (name, state, party)
- Each **mayor governs** exactly one **city**. Each **governor governs** exactly one **state**.
- **(a)** Write create table statements that encode these relation schemas and business rules with the right foreign key constraints.

```
create table City (
        name        varchar(64),
        state       varchar(32),
        population number,
        elevation   number,
        primary key (name, state)
);
```

```
create table State (
        name varchar(32) primary key,
        region varchar(32)
);
```

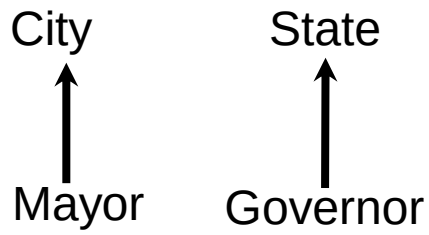
```
create table Mayor (
    name varchar(128),
    city    varchar(64),
    state   varchar(32),
    party varchar(32),
    primary key (name, city, state),
    foreign key (city, state) references City(name, state)
);
```

```
create table Governor (
    name varchar(128),
    state  varchar(32),
    party varchar(32),
    primary key (name, state),
    foreign key (state) references State(name)
);
```

Problem 4: Foreign keys

- Consider relation schemas below, with primary keys underlined.
  - City (<u>name</u>, <u>state</u>, population, elevation)
  - State (<u>name</u>, region)
  - Mayor (<u>name</u>, <u>city</u>, <u>state</u>, party)
  - Governor (<u>name</u>, <u>state</u>, party)
- Each mayor governs exactly one city. Each governor governs exactly one state.

- **(b)** In what order would you drop these tables? Give **all** valid sequences.



Governor, Mayor, City, State

Mayor, Governor, City, State

Mayor, City, Governor, State

Mayor, Governor, State, City

… …

Problem 4: Foreign keys

- Consider relation schemas below, with primary keys underlined.
  - City (name, state, population, elevation)
  - State (name, region)
  - Mayor (name, city, state, party)
  - Governor (name, state, party)
- Each **mayor governs** exactly one **city**. Each **governor governs** exactly one **state**.
- **(a)** Write create table statements that encode these relation schemas and business rules with the right foreign key constraints.

```
create table City (
name          varchar(64),
state         varchar(32),
population number,
elevation    number,
primary key (name, state)
foreign key (state) references State(name)  on delete cascade
);
```

```
create table State (
        name varchar(32) primary key,
        region varchar(32)
);
```

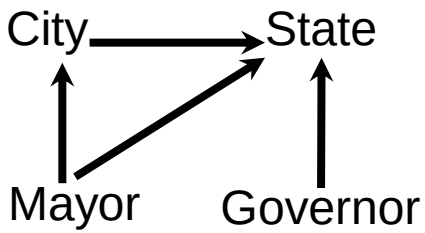
```
create table Mayor (
        name varchar(128),
        city    varchar(64),
        state   varchar(32),
        party varchar(32),
        primary key (name, city, state),
        foreign key (state) references State(name) on delete cascade
        foreign key (city) references City(name) on delete cascade
);
```

```
create table Governor (
        name varchar(128),
        state  varchar(32),
        party varchar(32),
        primary key (name, state),
        foreign key (state) references State(name)
        on delete cascade
);
```

Problem 4: Foreign keys

- Consider relation schemas below, with primary keys underlined.
  - City (<u>name</u>, <u>state</u>, population, elevation)
  - State (<u>name</u>, region)
  - Mayor (<u>name</u>, <u>city</u>, <u>state</u>, party)
  - Governor (<u>name</u>, <u>state</u>, party)
- Each mayor governs exactly one city. Each governor governs exactly one state.

- **(b)** In what order would you drop these tables? Give **all** valid sequences.



Governor, Mayor, City, State

Mayor, Governor, City, State

Mayor, City, Governor, State