Name	CWID

Final Exam

December 5th, 2017
2:00-4:00

CS425 - Database Organization
Results

Please leave this empty!

1.1	1.2	1.3	1.4	1.5	1.6	Sum

Instructions

- Try to answer all the questions using what you have learned in class. Keep hard questions until the end.
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- The exam is closed book and closed notes!

Consider the following laptop database schema and example instance:

person

name	job	dept	salary
Alice	manager	HR	100,000
Peter	admin	IT	100,000
Bob	case worker	HR	100,000

invitations

name	mId	attendance
Peter	1	yes
Bob	1	maybe
Aice	2	no
Bob	2	maybe
Peter	2	yes

meeting

mId	building	room	date	createdBy
1	Stuart	208	12-07-17	Peter
21	Life Sciences	540	12-08-17	Peter

location

building	room	maxOccupancy	type
Stuart	104	150	classroom
Stuart	208	53	classroom
Life Sciences	540	5	office

Hints:

- Attributes with black background form the primary key of a relation (e.g., *name* for relation *person*)
- The attribute *name* of relation *invitations* is a foreign key to relation *person*
- The attribute *mId* of relation *invitations* is a foreign key to relation *meeting*
- The attributes *building* and *room* of relation *meeting* are a foreign key to relation *location*.
- The attribute *createdBy* of relation *meeting* is a foreign key to relation *person*
- All foreign keys have been created with the **CASCADE** option.

Part 1.1 Relational Algebra (Total: 20 Points)

The queries in this part should be written using the **set semantics** version of relational algebra.

Question 1.1.1 (6 Points)

Write a **relational algebra** expression that returns the `mId` for all meetings that a person named "Peter" got invited to (table `invitations`).

Solution

$$\pi_{mId}(\sigma_{name=Peter}(invitations))$$

Question 1.1.2 (7 Points)

Write a **relational algebra** expression that returns all meetings (`building`, `room`, and `date`) that take place in offices (`location type`) and are overbooked. A meeting is overbooked if the number of people that have been invited to the meeting is larger than the maximal occupancy of the room (`maxOccupancy`).

Solution

$$\pi_{building,room,date}\sigma_{maxOccupancy<numInvites}({}_{mId}\mathcal{G}_{numInvites \leftarrow count(*)}(invitations) \bowtie meeting \bowtie \sigma_{type=office}(location))$$

Question 1.1.3 (7 Points)

Write a **relational algebra** expression that returns the `mId` of meetings where all invited persons have confirmed their attendance (`attendance` is `yes`)

Solution

$$\pi_{mId}({}_{mId}\mathcal{G}_{n \leftarrow count(*)}(invitations) \bowtie {}_{mId}\mathcal{G}_{n \leftarrow count(*)}(\sigma_{attendence=yes}(invitations)))$$

Part 1.2 SQL - DDL (Total: 7 Points)

Question 1.2.1 (7 Points)

Write an **SQL statement** that creates a new relation equipment that records information about equipement available at locations. For each piece of equipement we want to store a model, the category (e.g., audio, video), its price, its weight, and a building and room. The combination of building and room identifies a location from table location. A piece of equipement is identified by a combination of its model and the location where it is at. Ensure that the price and weight of a piece of equipment is a positive number.

Solution

```sql
CREATE TABLE equipment (
    model VARCHAR(100),
    category VARCHAR(50),
    price NUMERIC,
    weight NUMERIC,
    building VARCHAR(200),
    room VARCHAR(50),

    PRIMARY KEY (model, building, room),
    FOREIGN KEY (building, room) REFERENCES location,
    CHECK(price > 0 AND weight > 0)
);
```

Part 1.3 SQL - Queries (Total: 20 Points)

Question 1.3.1 (4 Points)

Write an **SQL query** that returns meetings created by *Peter* for which not all participants have confirmed their attendance.

Solution

```
SELECT DISTINCT mId
FROM meeting
WHERE mId IN (SELECT mId FROM invitations WHERE attendance <> 'yes')
```

Alternatively, use a join.

Question 1.3.2 (3 Points)

Write an **SQL query** that returns the number of meetings for each `date`.

Solution

```
SELECT count(*) AS numMeet, date
FROM meeting
GROUP BY date;
```

Question 1.3.3 (6 Points)

Write an **SQL query** that returns the `building`, `room`, and `date` of the meeting with the most invited persons.

Solution

```
WITH numInvited AS (
  SELECT count(*) AS n, mId
  FROM invitations
)
SELECT mId
FROM numInvited
WHERE n IN (SELECT max(n) FROM numInvited);
```

Question 1.3.4 (7 Points)

Write an **SQL query** that returns the `names` of persons which have not been invited to any meetings yet.

Solution

```
SELECT name
FROM person
WHERE name NOT IN (SELECT name
                   FROM invitations);
```

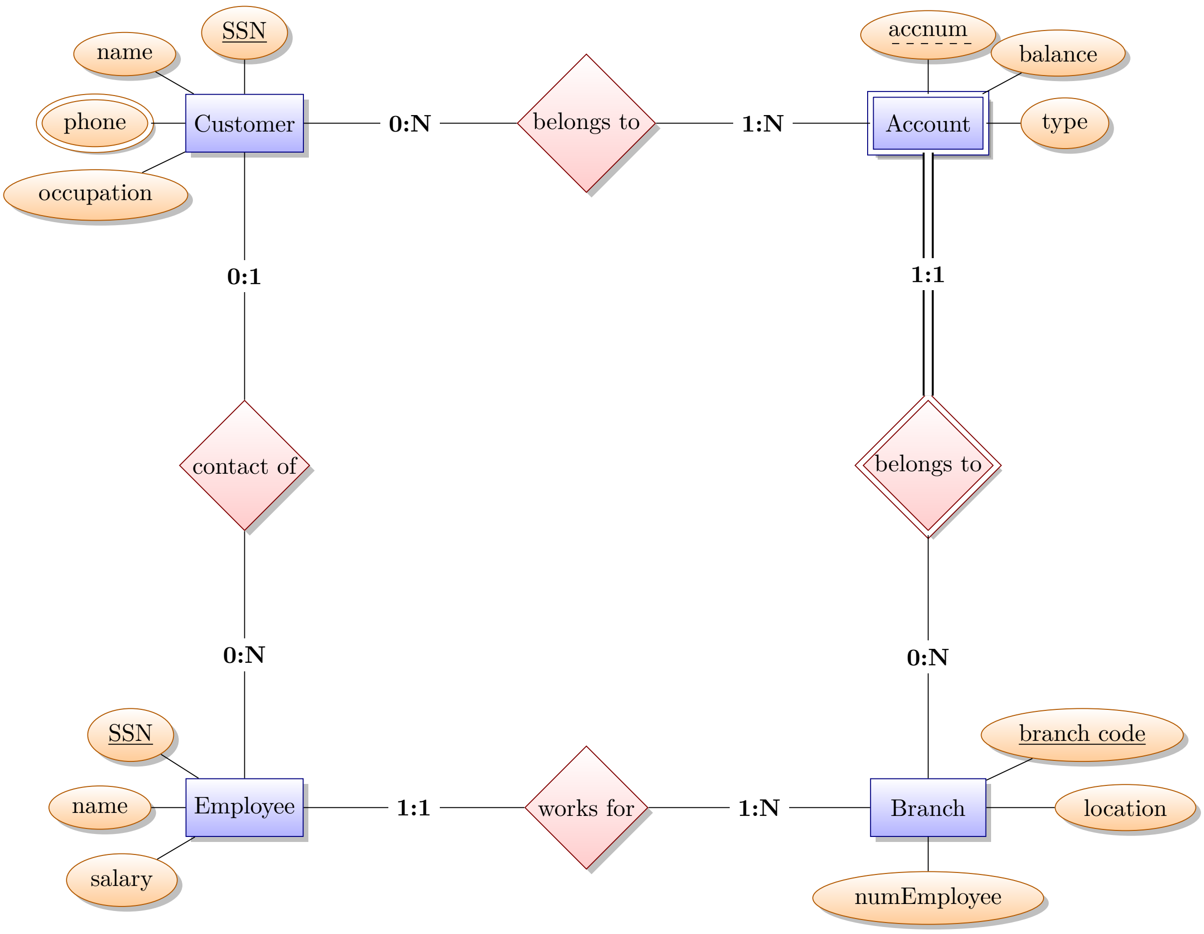
Part 1.4 ER Design (Total: 22 Points)

Question 1.4.1 (22 Points)

Design an **ER-diagram** for a bank that implements the following requirements. The database you design should store information about customers, accounts, branches and employees

- **Customer**: Customers are identified by their `SSN`. For each customer we store a `name`, multiple `phone numbers` (one or more), and an `occupation`.
- **Account**: Accounts are identified by an `account number` and the `branch` they belong to. For each account we store a `balance` and the `type` of `account` (e.g., savings).
  - An `account` belongs to one or more `customers`. A `customer` can have any number of `accounts`.
  - An `account` belongs to exactly one `branch`. Obviously, `branches` can have multiple `accounts` (`branches` are not required to have `accounts`).
- **Branch**: A branch is identified by a unique `branch code`. For each branch we want to store a `location` and `number of employees`.
- **Employee**: Employees are identified by their *SSN*. For each employee we store a `name` and `salary`.
  - An `employee` works for exactly one `branch`. `Banches` have one or more `employees`.
  - An `employee` is the contact for zero or more `customers`. Every `customer` has at most one `employee` as a contact.

Solution


SSN
name
phone
occupation
Customer
0:N
belongs to
1:N
accnum
balance
Account
type
0:1
1:1
contact of
belongs to
0:N
0:N
SSN
name
salary
Employee
1:1
works for
1:N
branch code
Branch
location
numEmployee

Part 1.5 Normalization and Functional Dependencies (Total: 22 Points)

Consider the following relation $R(A, B, C, D, E, F)$ and functional dependencies $\mathcal{F}$ that hold over this relation.

$$
\begin{aligned}
\mathcal{F} = & A \rightarrow B, C \\
& C \rightarrow D, A \\
& D \rightarrow E \\
& A, D \rightarrow F
\end{aligned}
$$

Question 1.5.1 (4 Points)

Determine all candidate keys of R.

Solution

$\{A\}$ $\{C\}$

Question 1.5.2 (8 Points)

Compute the canonical cover of F. Show each step of the generation according to the algorithm shown in class.

Solution

1th iteration: 1) Apply union rule to combine right-hand sides:
no changes
1th iteration: 2) Find extraneous attribute:
D is extraneous in $A, D \rightarrow F$

$$
\begin{aligned}
\mathcal{F}_1 = & A \rightarrow B, C \\
& C \rightarrow D, A \\
& D \rightarrow E \\
& A \rightarrow F
\end{aligned}
$$

2nd iteration: 1) Apply union rule to combine right-hand sides:

$$
\begin{aligned}
\mathcal{F}_1 = & A \rightarrow B, C, F \\
& C \rightarrow D, A \\
& D \rightarrow E
\end{aligned}
$$

2nd iteration: 2) Find extraneous attribute:

no changes

Question 1.5.3 (6 Points)

Apply the 3NF decomposition algorithm to relation R.

Solution

1st step: Create one relation for each FD in F_c:

$$R_1(A, B, C, F)$$
$$R_2(A, C, D)$$
$$R_3(D, E)$$

2st step: Remove redundant relations
None are redundant

]

3st step: If no relation contains all attributes of a candidate key, then create a relation with all attributes of a candidate key
R_1 and R_2 contain a candidate key.

Question 1.5.4 (4 Points)

In what normal forms is relation R?

Solution

1NF

in 2NF because all candidate keys have only one attribute

It is not in 3NF, because in $D \to E$ neither is D a superkey, nor is E part of any candidate key.

Part 1.6 Concurrency Control (Total: 9 Points)

Question 1.6.1 (9 Points)

For each of the following schedules determine which properties this schedule has. E.g., a schedule may be *recoverable* and *cascade-less*. Consider the following notation for operations of transactions:

$w_1(A)$	transaction 1 wrote item A
$r_1(A)$	transaction 1 read item A
c_1	transaction 1 commits
a_1	transaction 1 aborts

$$S_1 = r_3(B), w_3(C), r_1(A), w_1(D), w_2(A), r_2(C), w_4(A), c_2, c_4, r_3(D), c_3, c_1$$

$$S_2 = r_3(C), w_1(A), r_2(B), w_3(B), c_3, w_1(C), c_1, r_2(A), c_2$$

$$S_3 = w_3(E), w_4(A), w_1(D), r_1(B), w_1(C), c_1, r_2(D), r_4(D), w_3(B), w_4(C), c_4, c_3, c_2$$

☐ S_1 is recoverable

☐ S_1 is cascade-less

■ S_1 is conflict-serializable

■ S_2 is recoverable

■ S_2 is cascade-less

☐ S_2 is conflict-serializable

■ S_3 is recoverable

■ S_3 is cascade-less

■ S_3 is conflict-serializable