Name	CWID

Final Exam

December 4th, 2013
8:00-10:00

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 database schema and example instance for a flight information system:

property

pId	price	owner	sqrFeet	managedBy	location
1	100,000	Alice	560	Property Pete	Lake View
2	3,400,000	Bob	2,000	Hyde Park Prop	Hyde Park
3	1,200,000	Bob	1,200	Property Pete	Hyde Park
4	5,000,000	Martha	800	Fancy Rentals	Evanston

management

mgmName	location	yearlyProfit
Property Pete	Lincoln Park	34,000,000
Hyde Park Prop	Downtown	3,000,000
Fancy Rental	Lake View	25,000,000

maintenance

cmpName	empName	salary	location
SuperPlumbing	George	10,000	Lake View
SuperPlumbing	Dave	30,000	Lake View
South Chicago Carpeting	Keith	15,000	South Chicago

repairs

cmpName	empName	pId	date	type
SuperPlumbing	George	1	2013-12-12	sink
SuperPlumbing	George	1	2013-12-13	toilet
South Chicago Carpeting	Keith	4	2012-01-01	painting

Hints:

- Attributes with grey background form the primary key of a relation (.e.g, *pId* for relation *property*)
- The attribute *managedBy* of relation *property* is a foreign key to relation *management*.
- The attribute *pId* of relation *repairs* is a foreign key to relation *property*.
- The attributes *cmpName* and *empName* of relation *repairs* form a foreign key to relation *maintenance*.
- All foreign keys have been created with the **CASCADE** option.

Part 1.1 Relational Algebra (Total: 20 Points)

Question 1.1.1 (4 Points)

Write a **relational algebra** expression that returns the pId, owner, and location of all properties that are larger than 600 square feet (*sqrFeet*).

Solution

$$\pi_{pId,owner,location}(\sigma_{sqrFeet>600}(property))$$

Question 1.1.2 (4 Points)

Write a **relational algebra** expression that returns the names of maintenance personal (*empName*) that did repair a property in Hyde Park.

Solution

$$\pi_{empName}(\sigma_{location='HydePark'}(property) \bowtie repairs)$$

Question 1.1.3 (5 Points)

Write a **relational algebra** expression that returns the number of repairs for property managed by property management company 'Property Pete' (*cmpName*) per location. E.g., this should return $(2, HydePark)$ if there is a property managed by 'Property Pete' in Hyde Park that has been repaired twice.

Solution

$$\begin{aligned} peteRepair &\leftarrow repairs \bowtie \sigma_{managedBy='PropertyPete'}(property) \\ q &\leftarrow_{location} \mathcal{G}_{count(*)}(peteRepair) \end{aligned}$$

Question 1.1.4 (7 Points)

Write a **relational algebra** expression that returns the name of maintenance companies (*cmpName*) that only did repairs in the area they are located in. E.g., in the example instance 'SuperPlumbing' is located in 'Lake View' and, thus, should be returned if all repairs executed by 'SuperPlumbing' are for properties in 'Lake View'.

Solution

$$\begin{aligned} j &\leftarrow \rho_p(property) \bowtie \rho_r(repairs) \bowtie_{r.cmpName=m.cmpName} \rho_m(maintenance)) \\ nonLocalRepairs &\leftarrow \pi_{cmpName}(\sigma_{p.location \neq m.location}(j)) \\ q &\leftarrow \pi_{cmpName}(maintenance) - nonLocalRepairs \end{aligned}$$

Alternatively, count the number of non-local repairs per company and return only companies where the non-local repairs count is 0. Or compare the number of local repairs with the number of repairs.

Part 1.2 SQL - DDL (Total: 11 Points)

Question 1.2.1 (7 Points)

Write an **SQL statement** that creates a new relation *tenant* that records information about tentants that living at property. Tenants are uniquely identified by their SSN. For each tenant we want to record the property the tenant is living at (*pId*), the yearly income of the tenant, whether he.she is the leasee or not, and a bank account number. We require that we have to have the bank account and yearly income information for each leasee, but not necessarily for all tentants living at a property.

Solution

```
CREATE TABLE tenant (
    ssn CHAR(10) PRIMARY KEY,
    pId INT,
    yearlyIncome INT,
    isLeasee INT,
    accountNumber VARCHAR(20),
    FOREIGN KEY (pId) REFERENCES property,
    CHECK(isLeasee = 0 OR (yearlyIncome IS NOT NULL AND accountNumber IS NOT NULL)),
    CHECK(isLeasee IN (0,1))
);
```

Question 1.2.2 (4 Points)

Write an **SQL statement** that creates a new view *HydeParkProperties* that contains all property information (*pId*, *price*, ...) for properties in 'Hyde Park'.

Solution

```
CREATE VIEW HydeParkProperties AS
  SELECT * FROM property WHERE location = 'Hyde␣Park';
```

Part 1.3 SQL - Queries (Total: 20 Points)

Question 1.3.1 (4 Points)

Write an **SQL query** that returns all owners whose sink's have been repaired, i.e., that own a property where the sink has been repaired.

Solution

```
SELECT DISTINCT owner
FROM property NATURAL JOIN repairs
WHERE type = 'sink';
```

Solutions without the distinct should be accepted.

Question 1.3.2 (4 Points)

Write an **SQL query** that returns the maximum salary per maintenance company.

Solution

```
SELECT MAX(salary), cmpName
FROM maintenance
GROUP BY cmpName;
```

Question 1.3.3 (5 Points)

Write an **SQL query** that returns average price of properties in Lake View that are between 500 and 800 square feet large.

Solution

```
SELECT AVG(price)
FROM property
WHERE location = 'Lake␣View' AND sqrFeet BETWEEN 500 AND 800;
```

Question 1.3.4 (7 Points)

Write an **SQL query** that for each location returned the name(s) of the management firm(s) which manage(s) the most expensive property at this location. Consider only locations which have at least one property.

Solution

```
SELECT managedBy AS firm, location
FROM
     property
     NATURAL JOIN
     (SELECT max(price) AS price, location
     FROM property
     GROUP BY location) AS maxP
```

or

```
SELECT managedBy AS firm, location
FROM property p1
WHERE price IN (SELECT max(price)
               FROM property p2
               WHERE p1.location = p2.location)
```

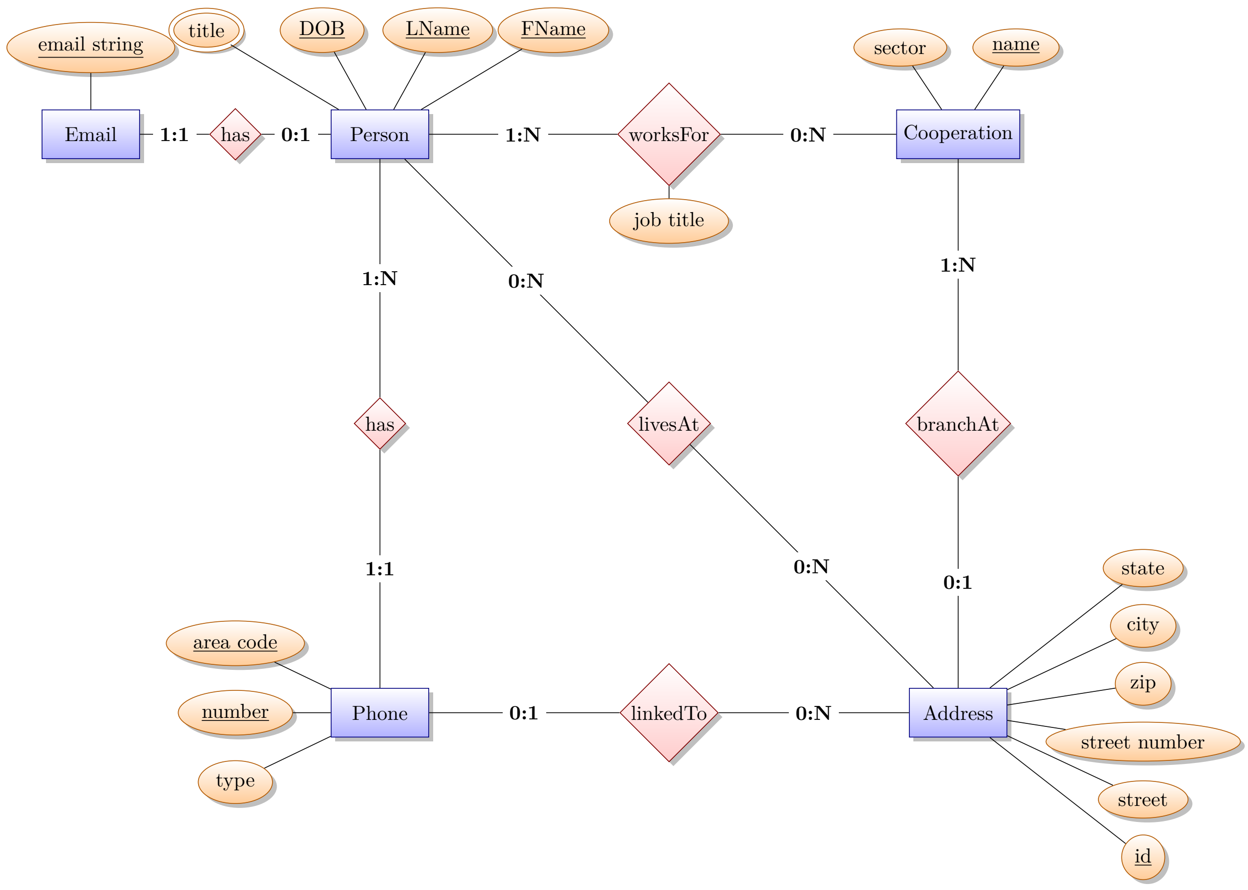
Part 1.4 Database Modelling - ER (Total: 22 Points)

Question 1.4.1 (22 Points)

Create an **ER-model** for a customer database for the marketing department of a cooperation based on the following requirements.

- Persons are identified by a combination of their first name, last name, and date of birth. A person can have an arbitrary number of titles (E.g., *Prof. Dr. Heinz Wichtig*).
- For cooperations we record the sector (e.g., *IT*) and a unique name.
- Persons work for one or more cooperations. For each relation between a person and a cooperation we record a job title. We allow cooperations in our database to not have any associated employees.
- Addresses consist of a street, street number, zip code, city, state, and a unique Id. A person can be associated with an arbitrary number of addresses and multiple persons can live at the same address. A cooperation is associated with one or more addresses. No two cooperations share the same address. Some addresses in the database may not be associated with any cooperation or person.
- Phone contacts consists of a type (e.g., *home* or *work*) and are uniquely identified by a combination of area code and number (e.g., `312` and `123456`). Each person is associated with at least one phone contact. Each phone belongs to exactly one person.
- Some, but not all phone contacts are associated with an address. We allow multiple phone numbers for the same address.
- An email address consists of a unique address string (e.g., `bglavic@iit.edu`). Each email address has one owner (a person), but not all persons have an email address. We do not record multiple email addresses per person.

Solution



Some alternative cardinalities are ok:

- **Person-Address:** 0:N on the person side

Part 1.5 Normalization and Functional Dependencies (Total: 14 Points)

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

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

Question 1.5.1 (3 Points)

Determine all candidate keys of R.

Solution

$\{A, B\}$ $\{A, C\}$

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:

$$\begin{aligned} F_1 = &A, B \to C \\ &C \to B \\ &A, C \to D, E \\ &A, B, D \to E \end{aligned}$$

1th iteration: 2) Find extraneous attribute:
D is extraneous in $A, B, D \to E$

$$\begin{aligned} F_2 = &A, B \to C \\ &C \to B \\ &A, C \to D, E \\ &A, B \to E \end{aligned}$$

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

$$\begin{aligned} F_3 = &A, B \to C, E \\ &C \to B \\ &A, C \to D, E \end{aligned}$$

2nd iteration: 2) Find extraneous attribute:
Attribute E is extraneous in $A, B \to C, E$.

$$\begin{aligned} F_4 = &A, B \to C \\ &C \to B \\ &A, C \to D, E \end{aligned}$$

3rd iteration: 1) + 2)
No more changes

$$\begin{aligned} F_c = &A, B \to C \\ &C \to B \\ &A, C \to D, E \end{aligned}$$

Question 1.5.3 (3 Points)

In which normal form is relation R (recall that a relation can be in multiple normal forms).

- [✓ 2NF]
- [✓ 3NF]
- ☐ BCNF

Part 1.6 Concurrency Control (Total: 13 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* (*strict*). 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

$$S_1 = r_1(C), w_1(C), r_1(A), w_1(A), r_2(B), r_2(A), w_2(B), c_2, w_1(C), c_1$$

$$S_2 = w_1(A), r_1(B), r_3(B), w_2(A), r_2(B), w_1(C), c_1, w_3(B), c_2, c_3$$

$$S_3 = r_1(A), w_1(A), r_2(A), w_2(A), r_3(A), w_3(A), r_2(B), w_2(B), c_2, r_1(B), w_1(B), c_1, c_3$$

- ☐ S_1 is recoverable
- ☐ S_1 is cascade-less
- ■ S_1 is conflict-serializable
- ■ S_1 is 2PL
- ☐ S_1 is S2PL
- ☐ S_1 is SS2PL

- ■ S_2 is recoverable
- ■ S_2 is cascade-less
- ■ S_2 is conflict-serializable
- ■ S_2 is 2PL
- ☐ S_2 is S2PL
- ☐ S_2 is SS2PL

- ☐ S_3 is recoverable
- ☐ S_3 is cascade-less
- ☐ S_3 is conflict-serializable
- ☐ S_3 is 2PL
- ☐ S_3 is S2PL
- ☐ S_3 is SS2PL

Question 1.6.2 (4 Points)

Check all correct statements below

- ❑ Every recoverable schedule is also cascade-less
- ■ In a recoverable schedule we never need to undo the operations of an already committed transaction
- ❑ Conflict-serializability and view-serializability are the same concept
- ■ Shared locks are compatible with shared locks under 2PL
- ❑ Exclusive locks are compatible with shared locks under 2PL
- ■ Under SS2PL all locks are hold until transaction end
- ■ A schedule that is 2PL (could have been produced using the 2PL locking protocol) is conflict-serializable
- ■ Locking is a type of pessimistic concurrency control