

Name: _____

NetID: _____

MIDTERM EXAM, March 9, 2006

CS 411 Database Systems

Department of Computer Science

University of Illinois at Urbana-Champaign

Exam Rules:

- 1) Close book and notes, **75 minutes**, scratch papers are allowed.
- 2) Please write down your name and NetID number NOW.
- 3) Please wait until being told to start reading and working on the exam.
- 4) No question can be asked during the exam (due to departmental regulations, to be fair to off-campus students). If you think a problem is ambiguous, write down your assumptions, argue that they are reasonable, then work on the problem using those assumptions.
- 5) No electronic devices are permitted.

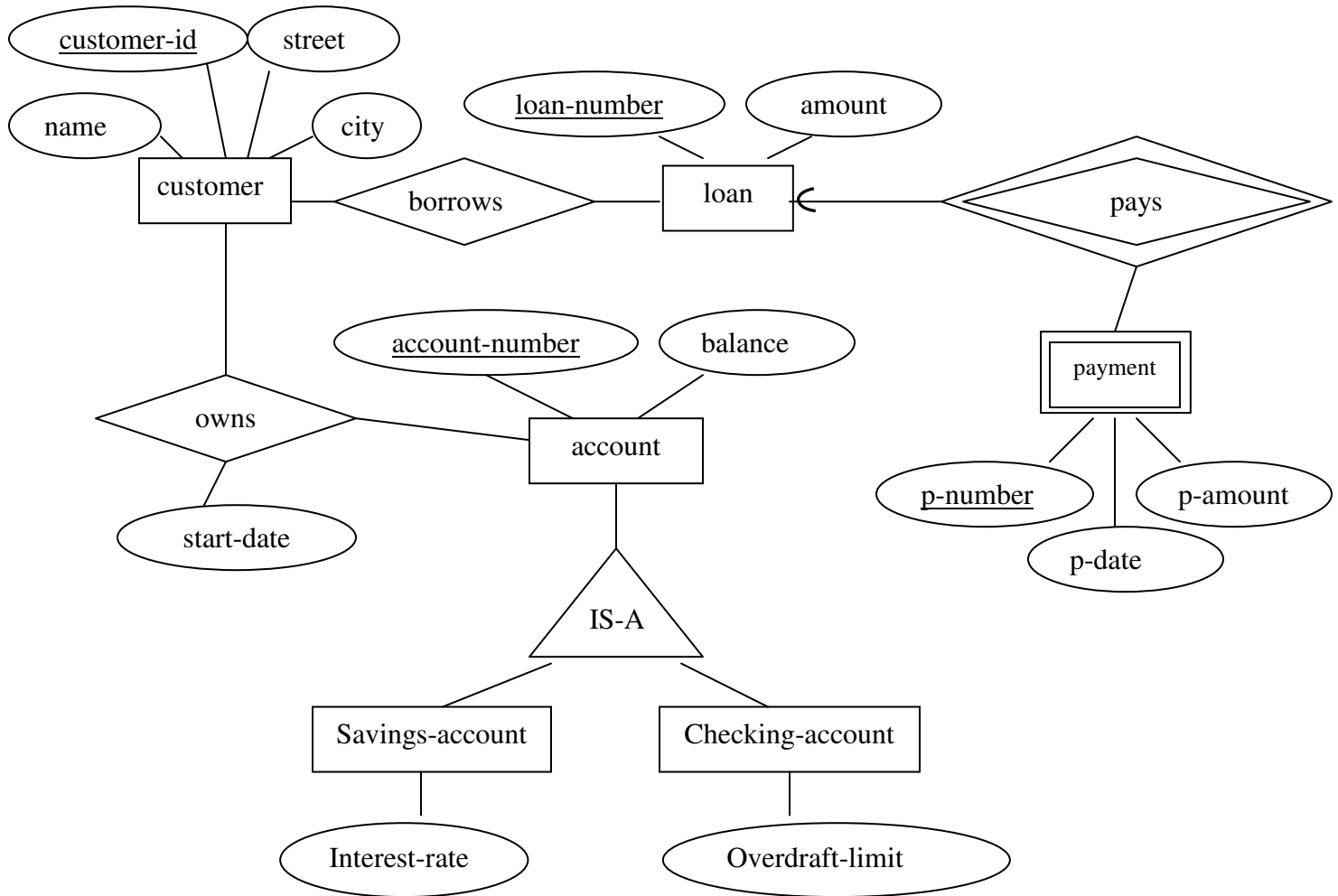
Scores:

Problem 1: out of 15 points
Problem 2: out of 15
Problem 3: out of 20
Problem 4: out of 30
Problem 5: out of 20

Total: out of 100 points

Problem 1. (15 points) ER and translation to relation model

Translate the following ER diagram into a relational schema. For each relation in your schema, **specify the key** of that relation. In translating a subclass hierarchy, **use the ER style translation**.



customer(customer-id, name, street, city)
loan(loan-number, amount)
payment(p-number,loan-number, p-date, p-amount)
borrows(customer-id, loan-number)
account(account-number, balance)
owns(customer-id,account-number, start-date)
Savings-account(account-number, Interest-rate)
Checking-account(account-number, Overdraft-limit)

Grading Scheme:

- 4pts for the “payment” relation
- 3pts for the “Savings-account” relation
- 3pts for the “Checking-account” relation
- 1pt for each other relation

Problem 2. (15 points) Schema Refinement

Consider the relation $R(A,B,C,D,E)$ with the following functional dependencies:

$$A \rightarrow B, A \rightarrow C, BC \rightarrow A, D \rightarrow E.$$

Is R in BCNF or 3NF or neither? If R is not in BCNF, decompose R into a collection of BCNF relations. Show each step of the decomposition process.

[NOTE] $FDs = \{A \rightarrow B, A \rightarrow C, BC \rightarrow A, D \rightarrow E\} = \{A \rightarrow BC, BC \rightarrow A, D \rightarrow E\}$

It is neither in BCNF nor in 3NF.

The keys are AD, BCD .

All the FDs violate the BCNF condition as none of their LHS are superkeys.

There are two possible decompositions: $\{ABC, DE, AD\}$ or $\{BCA, BCD, DE\}$

Decomposition 1

Decomposing on $A \rightarrow BC$:

$$R_1 = (ABC), FD_1 = \{A \rightarrow BC, BC \rightarrow A\}:$$

This is in BCNF as A and BC are keys and LHS of both FDs are super keys.

$$R_2 = (ADE), FD_2 = \{D \rightarrow E\}:$$

This is not in BCNF because the key is AD where as LHS of the FDs is not a superkey.

Decomposing R_2 on $D \rightarrow E$:

$$R_{21} = (DE), FD_{21} = \{D \rightarrow E\}$$

This is in BCNF as the key is D and LHS of the FD a superkey.

$$R_{22} = (AD), FD_{22} = \{\}$$

This is in BCNF as there are no FDs

Therefore the decomposed schema is:

$$R_1 = (ABC), FD_1 = \{A \rightarrow BC, BC \rightarrow A\}, R_{21} = (DE), FD_{21} = \{D \rightarrow E\}, R_{22} = (AD), FD_{22} = \{\}$$

Decomposition 2

Decomposing on $BC \rightarrow A$:

$$R_1 = (BCA), FD_1 = \{A \rightarrow BC \text{ or } BC \rightarrow A\}$$

This is in BCNF as A and BC are the keys and hence LHS of all FDs are superkeys.

$$R_2 = (BCDE), FD_2 = \{D \rightarrow E\}$$

This is not in BCNF as the key is BCD and hence LHS of FD $D \rightarrow E$ is not a superkey.

Decomposing R_2 on $D \rightarrow E$:

$$R_{21} = (DE), FD_{21} = \{D \rightarrow E\}$$

This is in BCNF as D is the key and the LHS of the only FD.

$$R_{22} = (BCD), FD_{22} = \{\}$$

This is in BCNF because there is no FD.

Therefore the decomposed schema is:

$$R_1 = (BCA), FD_1 = \{A \rightarrow BC, BC \rightarrow A\}, R_{21} = (DE), FD_{21} = \{D \rightarrow E\}, R_{22} = (BCD), FD_{22} = \{\}$$

Grading Scheme:

- 5 pts for the first part and 10 pts for the decomposition.

Problem 3. (20 points) Relational Algebra

This problem has 2 parts.

Consider the following database schema:

course (course#, dept-name)

enroll (studentID, course#, status)

(a) (10 points) Write a **relational algebra** expression to find the course# of all the courses offered by CS department.

Project_{course#}(Select_{dept-name=="CS"}(Course))

Grading Scheme:

- No partial credits here.

(b) (10 points) Write a **relational algebra** expression to find the studentIDs of all the students who are not enrolled in course# 411.

Project_{studentID}(enroll) – Project_{studentID}(Select_{course# == "411"}(Enroll))

Some students answered:

Project_{studentID}(Select_{course# <> "411"}(Enroll))

This answer is incorrect. Suppose there is a student who does not register for 411, however, the above query would still return the studentID for that student if he/she is registered for any course other than 411.

And a few students answered:

Project_{studentID}(enroll – Select_{course# == "411"}(Enroll))

This was a good attempt at removing above problem, however, still suffers from that problem.

A good way to verify your relational algebra expression or sql query is to try on a small example.

Grading Scheme:

- For the two incorrect answers, we award 5/10 points.

Problem 4. (30 points) SQL

This problem has 3 parts.

Again consider the following database schema:

course (course#, dept-name)

enroll (studentID, course#, status)

(a) (10 points) Write a **SQL** query that finds the studentID of all the students who are enrolled in at least one course offered by CS department and are not enrolled in any courses offered by EE department.

```
{ SELECT studentID
FROM course, enroll
WHERE course.course# = enroll.course# AND dept-name = 'CS' }
EXCEPT
{ SELECT studentID
FROM course, enroll
WHERE course.course# = enroll.course# AND dept-name = 'EE' }
```

(b) (10 points) Write a **SQL** query that lists the course# of all the courses for which more than 50 students are enrolled.

```
SELECT course#
FROM enroll
GROUP BY course#
HAVING count(studentID)>50
```

(c) (10 points) Write a **SQL** query that finds the studentID of all the students who are enrolled in the maximum number of courses. (For example, suppose there are 3 students, each is enrolled in 5 courses, and all other students are enrolled in fewer than 5 courses. Then you are expected to return the studentID of these 3 students.)

```
CREATE VIEW v AS
SELECT student ID, count(course#) as NumOfCourses
FROM enroll
GROUP BY studentID

SELECT studentID
FROM v
WHERE NumOfCourses=max(NumOfCourses)
```

Problem 5. (20 points) Constraints and Assertions

In this problem, you are given the schema for a bank and will be required to **write an assertion**. The relation *Account-owners* is a many-to-many relationship between the relation *Customers* and the relation *Accounts*. The SQL definitions for these relations are as follows:

```
create table Customers
(customer-name    char(20),
 customer-address char(20),
 primary key (customer-name))
```

```
create table Accounts
(account-number   char(10),
 balance         integer,
 primary key (account-number))
```

```
create table Account-owners
(customer-name    char(20),
 account-number   char(10),
 primary key (customer-name, account-number),
 foreign key (customer-name) references Customers(customer-name),
 foreign key (account-number) references Accounts(account-number))
```

Write an assertion such that for every customer at least one of the following conditions holds true:

- He (or she) is owner of at most 5 accounts
- The sum of the balance of various accounts he (or she) owns is greater than \$50,000.

```
CREATE ASSERTION mid-term-assert AS CHECK
(NOT EXISTS (SELECT O.customer-name
             FROM Accounts A, Account-Owners O
             WHERE A.account-number == O.account-number
             GROUP BY O.customer-name
             HAVING COUNT(A.account-number) > 5
                    and SUM(A.balance) < 50000))
```

Alternative answer:

```
CREATE ASSERTION mid-term-assert2 AS CHECK
(NOT EXISTS (SELECT * FROM Customers C
             WHERE (SELECT count(*) FROM Account-Owners O
                    WHERE O.customer-name == C.customer-name) > 5
             AND (SELECT SUM(balance)
                  FROM Accounts A, Account-Owners O2
                  WHERE A.account-number == O2.account-number
                        AND A.customer-name == C.customer-name) < 50000
             ))
```

Grading Scheme:

- checking the two conditions: 8 each
- combining the two conditions: 2 points
- syntax of assertion: "CREATE ASSERTION name CHECK (some Boolean condition)": 2 points