

CS411 Database Systems  
*Fall 2008*

University of Illinois at Urbana-Champaign

Midterm Examination  
October 14, 2008  
Time Limit: 75 minutes

- Print your name and NetID below. In addition, print your NetID in the upper left corner of every page.

**Name:** \_\_\_\_\_ **NetID:** \_\_\_\_\_

- Closed notes; closed book; no sheet of formulas permitted.
- Please write your answers directly on the exam sheet. The space we left for your answers is often more than what you actually need. Please use the back side of the exam as scratch paper.
- In case you find a question ambiguous, please write down your assumption and answer the question accordingly.
- You may use temporary relations, if you like, for any of the queries below. If you use the same temporary relation for a second exam question, you must redefine the relation in your answer to the second question. In other words, your answer to each question should be self-contained.

Problem	1	2	3	4	5	Total
Points	10	20	15	20	35	100
Score						
Grader						

**Turn over the page when instructed to do so. Good Luck!**

**I. [True/False questions, 10 points]** For each of the following statements, indicate whether it is *TRUE* or *FALSE* by circling your choice. You will get *1 point* for each correct answer, *-1 point* for each incorrect answer, and *0 point* for each answer left blank.

1. **True False**  
Any candidate key of a relation is a super key for that relation.
2. **True False**  
All attributes of a relation together form a super key for the relation.
3. **True False**  
A relation decomposition is **not** always lossless.
4. *True* **False**  
One weakness of triggers is that they can only be activated for insertion and update operations.
5. *True* **False**  
The null value approach in translating subclass relationship in the ER model always saves space
6. *True* **False**  
The default policy for the foreign-key constraint sets the value of a foreign-key to null when the tuple of its referenced attribute gets deleted.
7. *True* **False**  
Join is one of the five basic operation in relational algebra.
8. *True* **False**  
In a relation  $R(X, Y, Z, T)$ , if  $XY \rightarrow YZ$  then  $X \rightarrow Z$ .
9. *True* **False**  
The primary key of a relation may contain a NULL value as a value for their components.
10. *True* **False**  
The expression “(0 != NULL)” in the WHERE clause of a SQL query is evaluated to be true.

**2. ER Diagram and Relational Schema (20 points; 10 points for (a) and (b))**

Your task is to design a database for a banking system, which maintains information about customers and their accounts.

(a) Draw an ER diagram to model the application with the following assumptions:

- Each customer has a name, a permanent address, and a social security number.
- Each customer can have multiple phone numbers, and the same phone number may be shared by multiple customers.
- A customer can own multiple accounts, but each account is owned by a single customer.
- Each account has an account number, a type (such as saving, checking, etc), and a balance.
- The bank issues an account statement for each account and mails it to its account owner every month. As time goes on, there will be multiple statements of the same account.
- Each statement has an issued date and a statement ID. All the statements of the same account have different statement IDs, but two different accounts could have statements with the same statement ID. For example, it is possible that account A has a statement with ID '123', while account B has another statement with the same ID '123'.

If you need additional assumptions to complete your diagram, please state your assumptions here as well.

Solution:

The solution is one of the many possible solutions. See Figure 1.

Points breakdown:

- 2 points for reasonable entity sets
- 2 points for reasonable keys
- 2 points for correct relations
- 2 points for correct notations
- 2 points for recognizing statements is a weak entity (depending on your assumptions, phone may be a weak entity as well)

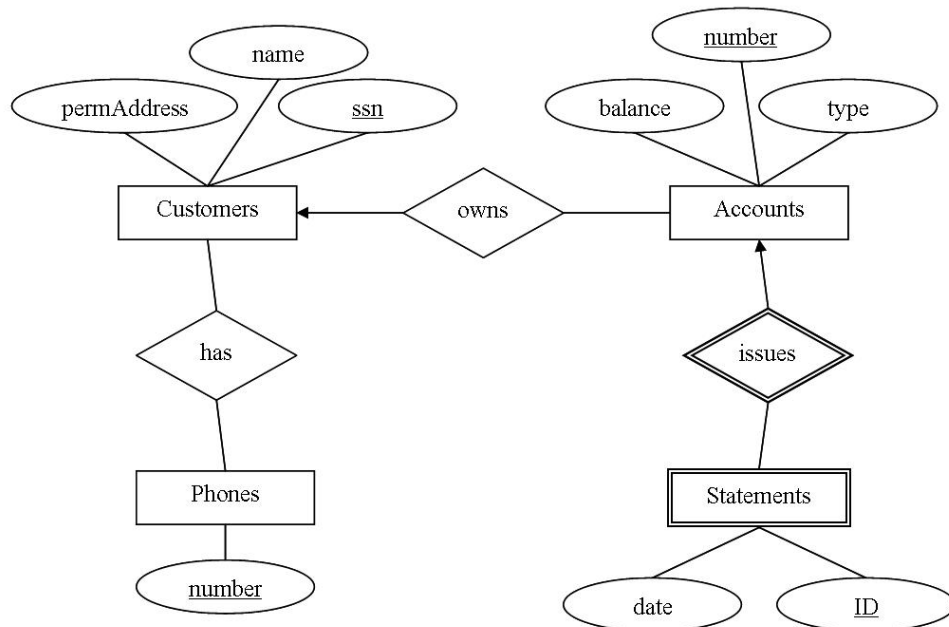


Figure 1: ER Diagram

- (b) Translate your ER diagram into a set of relations. Select approaches that yield the fewest number of relations; merge relations where appropriate.

Solution:

Customers(ssn, name, permAddress)

Accounts(number, type, balance, ssn)

Has(ssn, number)

Statements(number, ID, date)

Points breakdown:

- 4 points for translating entity sets and relations correctly
- 4 points for merging various relations correctly and appropriately (ie. merging of many-to-many relations is bad)
- 2 points for correct key identification
- part b) is graded based on part a) answer. No points are taken off for propagation errors from part a).

**3. Functional Dependencies and Normal Forms (15 points; 5 points for (a), 2 points for (b) and (c), and 6 points for (d))**

Consider a relation  $R(A, B, C, D, E)$  with FD's  $AB \rightarrow C$ ,  $CD \rightarrow E$ ,  $C \rightarrow A$ ,  $C \rightarrow D$ ,  $D \rightarrow B$ .

- (a) Determine all the keys of relation R. Do not list superkeys that are not a key.

Solution:

Keys: AB, AD, C

To get the key AB, we can do the following:

- From  $AB \rightarrow C$  and  $C \rightarrow D$ , we obtained  $AB \rightarrow D$ .
- From  $AB \rightarrow C$  and  $AB \rightarrow D$ , we obtained  $AB \rightarrow CD$ .
- From  $AB \rightarrow CD$  and  $CD \rightarrow E$ , we obtained  $AB \rightarrow E$ .

To get the key C, we can do the following:

- From  $C \rightarrow A$  and  $C \rightarrow D$ , we obtained  $C \rightarrow AD$ .
- From AD, we can obtain the rest of the attributes.

To get the key AD, we can do the following:

- From  $D \rightarrow B$ , we can get  $AD \rightarrow AB$ .
- From AB, we can obtain the rest of the attributes.

Points breakdown:

- 3 points for correct keys.
- 2 points for work shown.

- (b) List all the FD's that violate 3NF, if any.

Solution: No violations.

Points breakdown:

- no points are taken off for propagation errors.

- (c) List all the FD's that violate BCNF.

Solution: Violation:  $D \rightarrow B$

Points breakdown:

- no points are taken off for propagation errors.

- (d) Decompose  $R$  using the BCNF decomposition algorithm. Show your work and summarize your final set of relations.

Solution:

Use the violation  $D \rightarrow B$  to decompose the relation.

$R_1=DB$ ,  $R_2=ACDE$

Points breakdown:

- 4 points for correct decomposition
- 2 points for work shown.
- no points are taken off for propagation errors.

**4. Relational Algebra (20 points; 10 points for (a) and (b))**

Tables 1, 2, and 3 on the next page show an example instance of corporate database for a coalition of retail stores. Those tables maintain information about stores, products, and the inventories of products in different stores. We assume that all the stores sell each product at the same price in Table 2.

Write down queries in relational algebra for the following questions. Do not use cartesian products if you can use joins instead. Please refer to the Store relation, Product relation, and Inventory relation as  $S$ ,  $P$ , and  $I$  respectively in your solutions. We here assume that relational algebra supports the set semantics; that is, each operation in relational algebra eliminates duplicate tuples from the output relation.

- (a) Return the name of the stores that have at least one product in their inventories whose unit price is greater than 2 USD.

$$\pi_{StoreName} \sigma_{UnitPrice > 2} (S \bowtie I \bowtie P)$$

- (b) Return the name of the products with the maximum unit price.

We first find out the products that do not have the maximum price. Let's rename Product relation to  $P_1$  and  $P_2$ :

$$A = \pi_{ProductName} (P_1 \bowtie_{(P_1.ProductName \neq P_2.ProductName) \text{ and } (P_1.UnitPrice < P_2.UnitPrice)} P_2)$$

Then we get the answer by computing the difference between the complete list of the product names and the above relation.

$$\text{Answer} = \pi_{ProductName} P_1 - A.$$

Table 1: Store relation: S

StoreID	StoreName
1	Kmart
2	Walmart
3	Safeway
4	Meijer
5	Schnucks
6	Xmart
7	Ymart

Table 2: Product relation: P

ProductName	UnitPrice(in USD)
Bread	1.6
Cheese-Cheddar	2
Cheese-Feta	4
Cheese-Livarot	9
Cheese-Mozzarella	5
Cucumber	4
Lettuce	1
Onion	2
Potato	2.5
Tomato	2.1
Tuna	0.5
Yam	4

Table 3: Inventory relation: I

ProductName	StoreID
Bread	2
Cheese-Cheddar	4
Cheese-Cheddar	5
Cheese-Feta	7
Cheese-Livarot	4
Cucumber	1
Lettuce	4
Onion	1
Potato	5
Tomato	3
Tuna	5
Yam	3

**5. SQL (35 points; 5 points for (a), 10 points for (b), (c), and (d))**

Consider the relations given in problem 4. Write down the following queries in SQL. You do not have to worry about duplicate tuples in the output relations of your queries.

- (a) Return the names of the stores that have Tomato and Lettuce in their inventories.

```
(Select StoreName
From Store,Inventory
Where Store.StoreID = Inventory.StoreID and Inventory.Product ='Tomato')
INTERSECT
(Select StoreName
From Store,Inventory
Where Store.StoreID = Inventory.StoreID and Inventory.Product ='Lettuce')
```

- (b) Return the names of the products whose unit prices are below the average unit price of all the products.

```
Select ProductName
From Product
Where UnitPrice <
( Select Avg(UnitPrice)
  From Product)
```

- (c) Each store has an *average unit price*. It is the average of the unit prices of the products available at the store. Return the average unit prices of all stores. Your query should produce pairs of store names and their average unit prices.

```
Select StoreName, Avg(UnitPrice)
From Store, Inventory, Product
Where Store.StoreID = Inventory.StoreID and Inventory.ProductName = Product.ProductName
GroupBy StoreName
```

- (d) Return the product(s) that are available in at least three stores.

```
Select ProductName
From Inventory
GroupBy ProductName
Having (Count(distinct StoreID) >= 3)
```