

The University of British Columbia
Computer Science 304

Midterm Examination
October 31, 2005

Time: 50 minutes

Total marks: 50

Instructor: Rachel Pottinger

Name **ANSWER KEY** Student No _____
(PRINT) (Last) (First)

Signature _____

This examination has 6 pages.

Check that you have a complete paper.

This is a closed book exam. No books or other material may be used.

Answer all the questions on this paper.

Give very **short but precise** answers.

State any assumptions you make

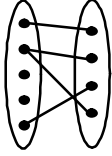
Work fast and do the easy questions first. Leave some time to review your exam at the end.

The marks for each question are given in {}. Use this to manage your time. Do not spend on a question more minutes than the marks assigned to it.

Good Luck

MARKS	
1.	
2.	
3.	
4.	
Total	

1. {8 marks, 1 mark per question} Circle only **one** answer per question – no points will be taken off for incorrect answers (i.e., you might as well guess):

<p>a. SQL is a procedural language <i>False. SQL is declarative – you specify what you want, not how you want it. The query optimizer will come up with a declarative query execution plan</i></p>	<p>True <u>False</u></p>
<p>b. We use ER diagrams to logically model concepts <i>False. ER diagrams are used for conceptual modeling, not logical modeling. This is on the reminder of where we are in the introduction slide for most units.</i></p>	<p>True <u>False</u></p>
<p>c.  This diagram is an example of a relationship with a many to one cardinality ratio <i>False. This is one to many. See ER diagram slides. Note that this was initially graded incorrectly, but this is the correct answer.</i></p>	<p>True <u>False</u></p>
<p>d. Every query that can be expressed in relational algebra can be expressed as a safe query in domain relational calculus; the converse is also true <i>True. The catch is that the queries must be safe, otherwise domain relational calculus is more expressive. See slides on DRC.</i></p>	<p><u>True</u> False</p>
<p>e. $AB \rightarrow C, D \rightarrow E, E \rightarrow C$ is a minimal cover for the set of functional dependencies $AB \rightarrow C, D \rightarrow E, AB \rightarrow E, E \rightarrow C$. <i>False. There's no way to derive $AB \rightarrow E$ from the first list of dependencies</i></p>	<p>True <u>False</u></p>
<p>f. An update anomaly is when it is not possible to store certain information unless some other, unrelated, information is stored as well. <i>False. An update anomaly is if one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly update. What is described is an insertion anomaly (see textbook page 607)</i></p>	<p>True <u>False</u></p>
<p>g. Given the table $R(a,b,c)$ (i.e., a and b form the primary key of R), the following is a valid table definition. CREATE TABLE S (a INTEGER, d INTEGER, e INTEGER, PRIMARY KEY (d), FOREIGN KEY (a) references R) <i>False. A foreign key must reference all of a key – in this case it must reference a,b</i></p>	<p>True <u>False</u></p>
<p>h. In ER diagrams, aggregation may be used instead of a ternary relation if we need to impose additional constraints <i>True. See slides or page 45 in the book</i></p>	<p><u>True</u> False</p>

2. {14 marks} Consider the schema $R = (A, B, C, D, E, F)$ together with the functional dependencies:

$A \rightarrow B$
 $DE \rightarrow F$
 $B \rightarrow C$

- a. {10 marks} Is $R(A, B, C, D, E, F)$ in BCNF? If so, say why. If not, decompose this relation into BCNF using the algorithm we covered in class and in the book; circle all relations in your final decomposition.

We begin by finding all of the (candidate) keys. The easiest way to do this is to calculate the closures. When in doubt, look at all closures. But we can tell that the only interesting closures are those that appear on the left hand side of some rule, so we'll start there:

$A^+ = ABC$
 $B^+ = BC$
 $DE^+ = DEF$

Thus we can see that A must be part of any key (since it's the only way to derive A), and that DE and E must similarly be part of any key. Since $ADE^+ = ABCDEF$, ADE is a key, and in fact, the only key of R .

For a relation S to be in BCNF, each non-trivial functional dependency of the form $X \rightarrow y$ (where X is a set of attributes, and y is a single attribute), it must be the case that X is a superkey of S .

Therefore, since ADE is the only key, all of the given functional dependencies violate BCNF, and we must decompose R .

We arbitrarily choose to begin decomposition on $A \rightarrow B$. We decompose into $R_1(AB)$, $R_2(A, C, D, E, F)$

Now we determine if we need to continue decomposing R_1 and R_2 . Because R_1 is a 2 valued relation, we know it is in BCNF, so we turn to R_2 . $DE \rightarrow F$ violates BCNF because in R $DE^+ = DEF$. Therefore, projected onto R_2 , $DE^+ = DEF$, and DE is not a key of R_2 . $DE \rightarrow F$ is therefore a violation of BCNF, and we must decompose again. We get $R_3(D, E, F)$, and $R_4(A, C, D, E)$. Are they in BCNF? R_3 is in BCNF because DE is a key, so it is not a BCNF violation, and there are no other non-trivial functional dependencies on R_3 . For R_4 , however, we know from above that $A^+ = ABC$. Projected onto R_4 , this means that $A^+ = AC$. Which means both that the functional dependency $A \rightarrow C$ holds on R_4 and that A is not a key of R_4 . So we decompose to yield $R_5(A, C)$, $R_6(A, D, E)$. R_5 is in BCNF since it only has 2 attributes, and since we know that the closure of A yields nothing in R_6 other than itself, and similarly for the closure of DE , R_6 is in BCNF.

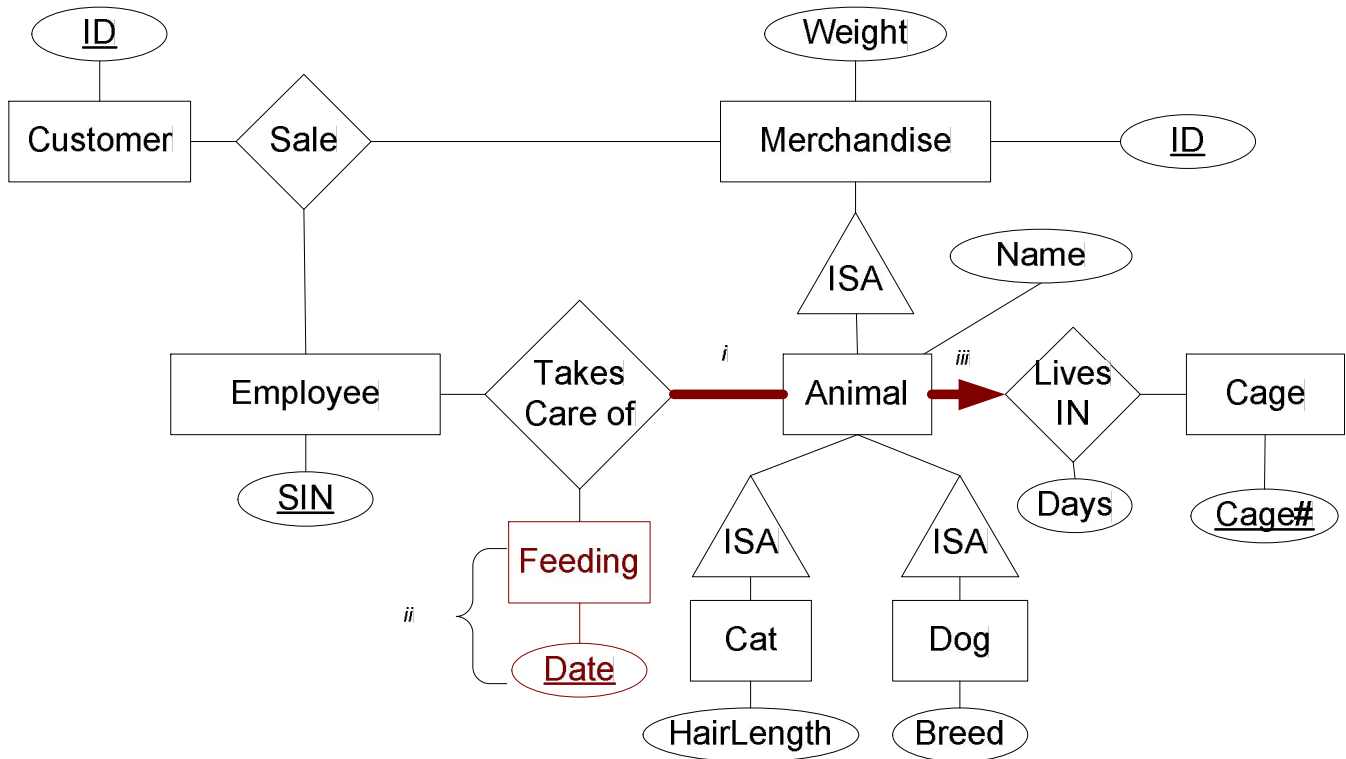
Therefore, the final answer is: $R_1(A, B)$, $R_3(D, E, F)$, $R_5(A, C)$, $R_6(A, D, E)$

- b. {4 marks} Is $R(A, B, C, D, E, F)$ in 3NF? If so, why. If not, list all violations of 3NF in R .

B , C , and F are part of no keys, nor, as illustrated above, are A , DE , or B part of a superkey. Therefore, all three violate 3NF.

3. {14 marks} Look at the following partially-constructed ER diagram about pet stores, which includes the following:

- Merchandise can be of many types, including animals and other types
- Animals can be cats, dogs, or other species
- Merchandise can be sold, which involves the merchandise, an employee, and a customer
- Animals are taken care of by some employee
- Animals live in cages



- a. {6 marks} Make the following changes/additions to the ER diagram:
- Add in the constraint that each animal must be taken care of by some employee
 - Add in the date that an employee takes care of an animal. Note that you should make it possible for an employee to take care of an animal on more than one date
 - Add in the specification that each animal lives in *exactly one* cage

See the above diagram. Note that we could not add an attribute for date to *Takes Care of*, because each relationship must be uniquely determined by its attributes, so we must add an entity for it. See Slide 22 on Chapter 2: Conceptual design.

- b. {6 marks} Transform the ER diagram, **including the modifications you made in part a**, into a relational schema using the methods discussed in class/the book. Note that this ER diagram has been carefully chosen so that there is *one* “right” choice for each transformation. State any assumptions that you make – but your assumptions cannot contradict the facts given. Output:
- i. Your resulting schema in the form Relation(attribute1, ..., attributeN), where you underline each relation’s primary key.

Customer (cid)

Employee(esin)

Merchandise(mid, weight)

AnimalLivesIn(Amid, Name, days, cage#) (combined because it’s many to one)

Cage(cage_num)

Cat(catid, HairLength)

Dog(dogid, Breed)

Feeding(Date)

Sale(cid, esin, mid)

TakesCareOf(sin, animal-id, date)

Note that because each Is-a relation is not total, and each superclass has a relationship on it, we must use the method where we create one table for each of Merchandise, Animal (which is combined with Lives in since it’s a many to one relationship), Dog, and Cat. See slides 15 & 16 in second part of Relational model.

- ii. For any foreign keys that you have identify the table in which they appear, and write how they would appear in SQL DDL – note you only have to declare the foreign key constraint in SQL DDL – *not* the rest of the question.

In Sale: Foreign Key (mid) references Merchandise

Foreign Key (esin) references Employee

Foreign Key (cid) references Customer

In Animal: Foreign Key(id) references Merchandise

In Cat: Foreign Key(catid) references Animal (We took to Merchandise, but should be Animal only)

In Dog: Foreign Key(dogid) references Animal

In TakesCareOf: Foreign Key(sin) references Employee

Foreign Key(animal-id) references Animal

Foreign Key(date) references Feeding

- c. {2 marks} Answer the following question about your relational schema: are there any constraints that cannot be modeled without using assertions? If so, which constraint(s)? If not, why not?

We cannot model that every animal has to be taken care of by some employee without assertions. See slides 10 & 11 in second part of Relational Model

4. {14 marks} Consider a relational database about real estate that is maintained by a real estate agency. The database consists of the following table (where the primary keys are underlined):

House (id, asking_price, address, postal_code, baths, beds, sqft, sellerID)
 Seller(id, name, home_phone, e-mail, agentID)
 Buyer(id, name, home_phone, e-mail, agentID)
 Agent(id, name, mobile_phone, e-mail)
 Sold(house_id, buyer_id, sale_date, selling_price)

Where House gives information about a house for sale, seller gives information about the sellers of a house, Buyer gives information about (prospective) home buyers, Agent gives information about agents (who can act on behalf of either the buyer or the seller), and sold gives information about the sale of a particular home – including the price at which the home actually sold, which may be different from the asking price.

For all questions, note that this is just a right way to answer the queries, there may be others

- a. {7 marks} answer the following questions in relational algebra:
 i. What are the id, addresses, asking_price, and selling_price of all houses that sold for less than the asking price?

$\pi_{house.id, address, asking_price, selling_price} (House \bowtie_{House.id = Sold.id \wedge Sold.selling_price < House.asking_price} Sold)$

- ii. What are names of all of (prospective) buyers who have not bought a house? Each name should appear only once.

$\pi_{name} (Buyer \bowtie_{id=buyer_id} (\pi_{buyer_id} (\rho(id \rightarrow buyer_id) Buyer) - \pi_{buyer_id} Sold))$

Note that because relational algebra treats all values as sets, we don't need to make any special arrangement to ensure that each name appears only once. Note that unless you were careful you missed non-buyers with same name as buyers

- b. {7 marks} Answer the following questions in SQL:

- i. For each postal code in which there were at least three houses sold, find the postal code and the average selling price of houses in that postal code.

Select H.postal_code, avg(selling_price) as AVG_Price

From House H, Sold S

Where H.id = S.house_id

Group by H.postal_code

Having Count(house_id) >= 3

Note: the count must be in the having clause, and the comparison must be in the where

- ii. Find the addresses and asking prices of all houses that have at least 3 bedrooms and two bathrooms that have not sold. Each address, asking price pair should appear only once.

Select DISTINCT H.address, H.asking_prices,

FROM House H,

Where H.beds > 2 and

H.baths > 1 and

NOT EXISTS (

Select house_id

From Sold S

Where S.house_id = H.id

)