CS 317 Mid Term Exam

Tuesday Sept 11th, 2007

This is an OPEN BOOK exam. You may refer to any material YOU have brought in.

1. Mark each of the following questions as True or False and justify in 1 sentence. **(2 points each)**

    a. A weak entity set in an ER diagram represents a collection of entities for which we have not discovered a key yet. Later in the design phase when we learn more about the domain, this set will be converted into a strong entity set by adding the appropriate attributes.

SOLUTION: False. Weak entities do not have a key. They depend on other entities.

    b. If `r(A, C)` and `s(A, D)` are any relations with the given attributes, the following RA expressions are equivalent.

i) $r \bowtie s$

ii) $\pi_{A,C,D}(\sigma_{r.A=s.A} (r \times s))$

SOLUTION: True

    c. **{AB->C, D->E, E->C}** is a minimal cover for the set of functional dependencies **{AB->C, D->E, AB->E, E->C}**

SOLUTION: False. There's no way to derive AB->E from the first list of dependencies

    d. 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)
```

SOLUTION: False. A foreign key must reference all of a key – in this case it must reference a,b

Consider the following relation with airline schedule information. (NUM-BUS and NUMCOACH are the numbers of business and coach seats on the aircraft.)

FLIGHT	FROM	TO	DEPART	ARRIVE	AIRCRAFT	NUMBUS	NUMCOACH
57	MAA	BOM	8:08a	9:20a	737	16	115
61	MAA	BOM	12:35p	1:45p	737	16	115
198	MAA	BOM	4:30p	5:30p	757	14	180
58	BOM	MAA	10:15a	11:25a	737	16	115
63	BOM	MAA	2:40p	3:45p	737	16	115
16	MAA	DEL	10:10a	12:55p	767	22	210
21	MAA	DEL	6:50p	9:20p	767	22	210
17	DEL	MAA	2:20p	4:55p	757	14	180
22	DEL	MAA	10:30p	12:55a	757	14	180
104	DEL	BOM	6:50p	10:15p	DC10	16	210
192	MAA	DEL	8:15a	11:25a	767	22	210

Which of the following FDs hold in this instance? **(3 points)**

FLIGHT -> FROM TO *HOLDS: FLIGHT is key for this instance, hence determines everything*

AIRCRAFT -> NUM-BUS *HOLDS: Tuples with the same AIRCRAFT have the same NUM-BUS*

NUM-BUS -> NUM-COACH *FAILS: See flights 63 and 104*

DEPART -> FROM *FAILS: See flights 21 and 104*

FROM TO -> AIRCRAFT *FAILS: See flights 61 and 198*

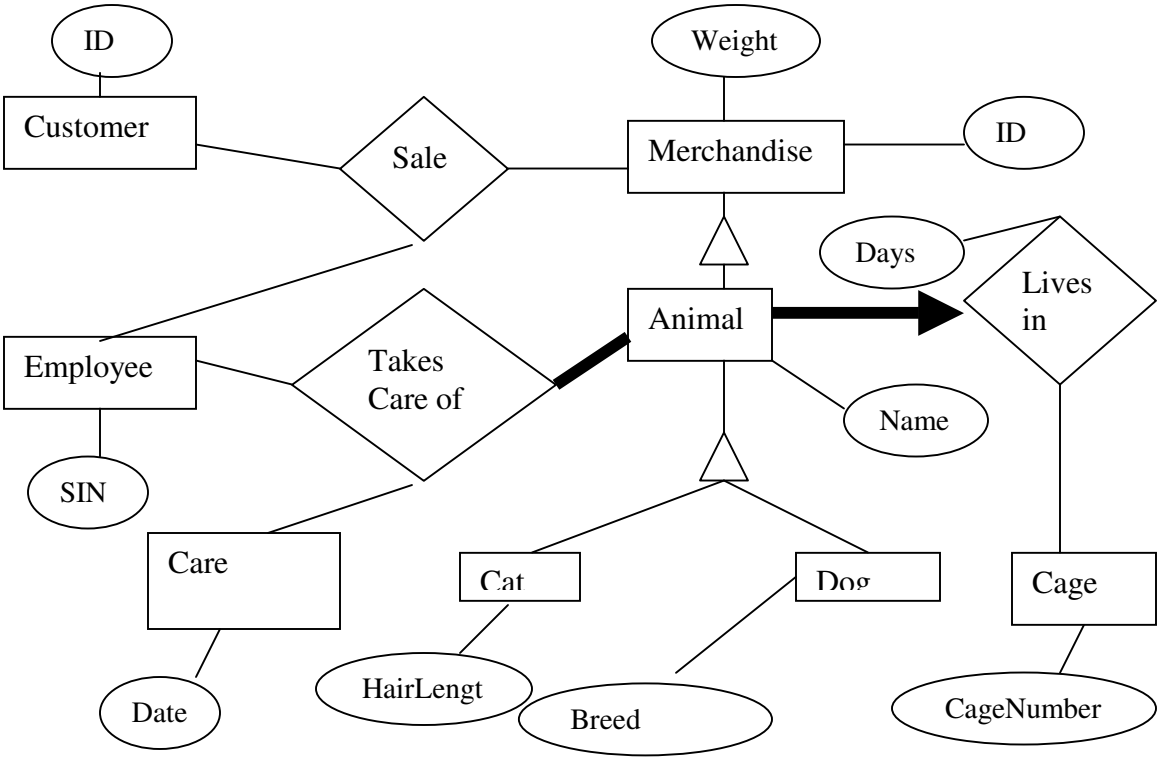
FROM TO DEPART -> ARRIVE *HOLDS: FROM TO DEPART are unique across tuples*

Find another functional dependency that holds for this instance. (It should be non-trivial.) **(3 point)**

Lots of choices: FLIGHT -> DEPART, AIRCRAFT -> NUM-COACH, NUM-BUS NUM-COACH -> AIRCRAFT, DEPART -> NUM-COACH

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
- Employees take care of animals (feed).
- Animals live in cages



Make the following changes/additions to the ER diagram: **(6 Points)**

e. Add in the *constraint* that each animal must be taken care of by some employee

f. 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.

g. Add in the specification that each animal lives in *exactly one* cage

SOLUTION: 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.

Transform the ER diagram, **including the modifications you made** into a relational schema using the methods discussed in class/the book. State any assumptions that you make – but your assumptions cannot contradict the facts given. Output: Your resulting schema in the form `Relation(attribute1, …, attributeN)`, where you underline each relation's primary key. **(5 points)**

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)
Care(Date)
Sale(cid, esin, mid)
TakesCareOf(sin,animalid, date)

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. **(3 points)**

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(animalid)
references Animal
Foreign Key(date) references Feeding

Answer the following question about your relational schema: are there any constraints discussed so far that cannot be modeled without using *assertions*? If so, which constraint(s)? If not, why not? **(2 points)**
SOLUTION: We cannot model that every animal has to be taken care of by some employee without assertions.

Consider a relational database about hotels, customers (guests) and their bookings that is maintained by an online hotel booking company. The database consisting of the following tables(where the primary keys are underlined):

Hotel (hId, hName, hAddress, hCity)
Guest(gId, gName, gAddress, gCity)
Room(hid, roomNo, type, price)
Booking(gId, hId, roomNo, fromDate, year, noOfDays)

Where, `hId` and `gId` are identifiers for the hotels and the guests, and the Booking relation indicated that a guest booked a hotel room for a specified number of days (noOfDays) starting from `fromDate` of a given year. For instance, a tuple **< g12345, h5555, 220, Jan05, 2007, 8>** in Booking indicates that guest `g12345` booked room 220 of the h5555 hotel for 8 days starting on Jan 5, 2007.

a. Write a **relational algebra** expression that returns the ids of the hotels located in Mumbai which were not booked at all in the year 2005.

$\Pi_{hId}\ (\rho_{hCity=\text{"mumbai"}}(\text{Hotel})) - \Pi_{hId}\ (\rho_{\ year=2005}(\text{Booking}))$

b. Write a **relational algebra** expression that returns the ids of the guests who have booked at least one room of type "suite" in every hotel located in Mumbai.

$\Pi_{\ gId,\ hId}\ (\rho_{\ type=\text{"suite"}}\ (\text{Booking NJOIN Room})) /$

$\Pi\ hId\ (\rho_{\ hCity=\text{"mumbai"}}\ (\text{Hotel})\ )$

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, email, agentID)
Buyer(id, name, home_phone, email, agentID)
Agent(id, name, mobile_phone, email)
Sold(house_id, buyer_id, saledate, 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.

Answer the following questions in SQL:

a. 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

b. 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
     )
```