Instructor: Zinovi Tauber 06/07/2006
TA: Maxim Roy Summer Semester

CMPT-354 Midterm

Total: 100 points – 25% of final grade.
Time: 50 minutes – 12:30-13:20
Difficulty: fill in the blanks[1] – Medium but long[2]

1. List at least three scenarios where converting from E/R diagrams to SQL would require you to have primary keys as foreign keys.

   Answer:
   - i. Converting a weak entity set. The weak entity set needs the primary key of its supporting entity.
   - ii. Converting a many-to-many relationship set. The relationship set needs the keys of all connecting sets to have a primary key.
   - iii. Converting an ISA hierarchy using E/R style conversion. The leaf relations need the primary key of the root as their key.

2. You want to design a bird-cataloguing database. You will store the bird family name, genus, wing length, beak type, average size, URL to pictures, migration routes and associated food. Each bird family has a genus, and a genus can contain many bird families. While genus names are unique, bird family names are unique only within a specific genus. The pictures of a birds are stored on a website by a webmaster, you are not responsible for the correctness of the URL given to you to store in the database. The migration route is an ordered set of countries and cities through which birds fly. The associated food is a list in no particular order of the types of food that a bird can eat.

   a. Draw an appropriate E/R-Diagram for the database specified.

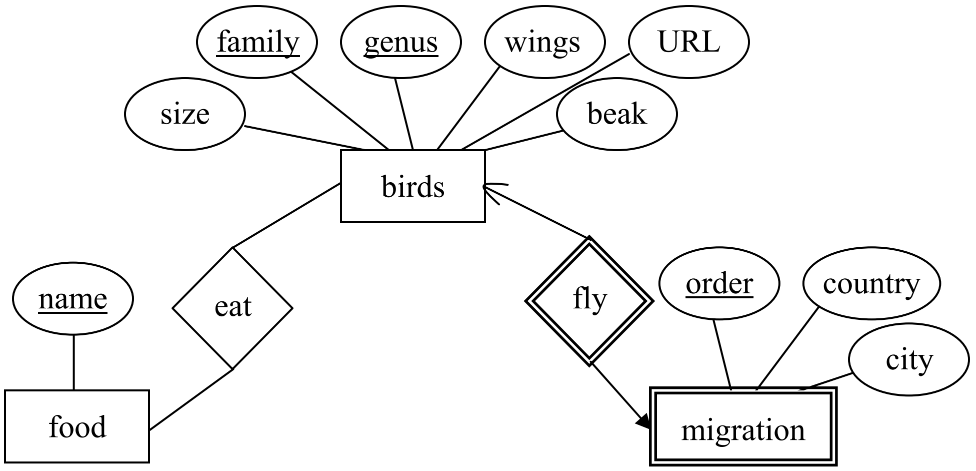
   b. How would your E/R-Diagram change if each genus has also associated descriptions? The storage has to minimize space.

   c. Suppose you wanted to store your guesses about the evolution of birds. Each bird can be evolved from one other bird, or we may not find any other birds we can suppose an evolution from. Draw this new information in an E/R-Diagram, drawing only the sections of the diagram from a. that change (ignore b.). Did you use a referential integrity in the drawing of the new information? Please justify, why or why not?

[1] In case you are drawing a blank on this warm up question, there's one already drawn. More seriously (though still not seriously at all), here are some options: easy/medium/hard. Additional options are: you're-wasting-my-time easy and insanely difficult. Please no obscenity or threats though – they will go to your TA who had nothing to do with your exam. Additionally, you may realize it cannot be a warm up question at all since you must answer it after you attempted the exam. Now, you are ready for the midterm.
[2] Instructor's response: well if it were easy and short it would be unfair as most people would get a perfect score. On the other hand if it were short and difficult, it would again be too long.

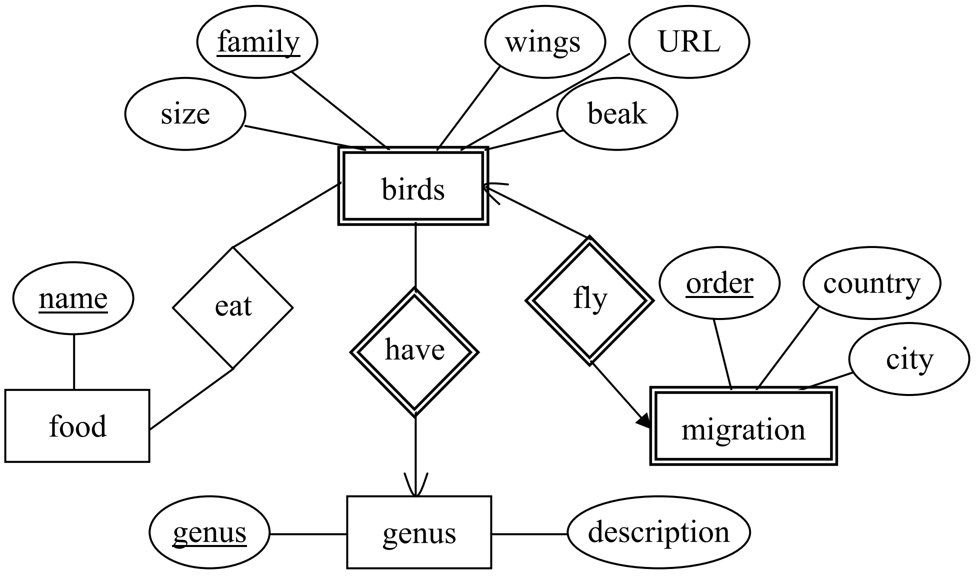
Answer:

a. Here we can have genus as a separate entity set, but there is no compelling reason to except to consider future expansion where genus will have its own properties. However, for simplicity here its better to just have genus as part of the birds entity set.



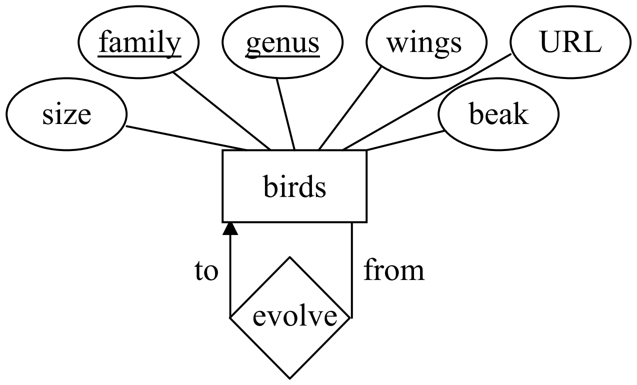
In order to have an ordered migration path, we need to associate order with each location. Unfortunately, no combination of attributes of migration can be a primary key since birds from Canada, Vancouver will always have a migration path part starting with (1, Canada, Vancouver). Although we can avoid redundancy by associating a location and order with any number of birds, this will introduce anomalies, such as deleting the migration path of one bird will delete a part of the path of another. As well, each bird family should have at most one migration path. A different approach would be to put the order attribute on the fly relation. Normally, this is wise as it allows us to store every country, city pair exactly once. However, here a bird can fly over a city twice, once on its way south, and once back up north. The order will then need to be part of the key. This is somewhat ambiguous as drawn on a relationship set since we have not defined it. Other assumption can be used.

b.



If a genus has associated descriptions then the descriptions in the diagram in a. will be duplicated for every bird family. Now we have a good reason to separate genus from family. This is an intuition for applying normalization.

c.



Referential integrity here is not necessary since some bird have no guesses as to which other birds they evolved from. It is best to simply not have these birds in the evolve relationship.

3. You know the following set of Functional Dependencies hold for fountain drinks in your store:
   i. money → drink_size
   ii. drink_color → flavour
   iii. flavour → drink_color
   iv. drink_size flavour → money

a. What are the closures of {money}, {flavor, drink_color} and {money, drink_color}?

b. What are the closures of the FDs in i., in ii. and in iv.?

c. What are the candidate keys for the relation Drinks(money, drink_size, drink_color, flavour)?

d. Decompose the Drinks relation into BCNF. For each relation give the primary key. Are any FDs violated? Give an example how.

e. Decompose the Drinks relation into 3NF. For each relation give the primary key.

f. Find a decomposition that's better than both BCNF and 3NF.

Answer:

a. $\{\text{money}\}^+ = \{\text{money, drink\_size}\}$
   $\{\text{flavor, drink\_color}\}^+ = \{\text{flavour, drink\_color}\}$
   $\{\text{money, drink\_color}\}^+ = \{\text{money, drink\_size, flavour, drink\_color}\}$

b. $\{\text{money} \rightarrow \text{drink\_size}\}^+ = \{\text{money} \rightarrow \text{drink\_size}\}$
   $\{\text{drink\_color} \rightarrow \text{flavour}\}^+ = \{\text{drink\_color} \rightarrow \text{flavour, flavour} \rightarrow \text{drink\_color}\}$
   $\{\text{drink\_size flavour} \rightarrow \text{money}\}^+ = \{\text{money} \rightarrow \text{drink\_size},$
   $\text{drink\_size flavour} \rightarrow \text{money, flavour} \rightarrow \text{drink\_color, drink\_color} \rightarrow \text{flavour}\}$

c. The candidate keys can be confirmed by the closures:
{flavour, drink_size}$^+$ = {flavour, drink_size, drink_color, money}
{money, flavour}$^+$ = {money, drink_size, flavour, drink_color}
{money, drink_color}$^+$ = {money, drink_size, flavour, drink_color}
{drink_size, drink_color}$^+$ = {money, drink_size, flavour, drink_color}

d. BCNF:
money → drink_size: {money}+ = {money, drink_size}
money is not a superkey for Drinks so:

SizeforMoney(money, drink_size)
Drinks(money, flavour, drink_color)

drink_color → flavour (can normalize instead on flavour → drink_color)
drink_color is not a superkey for Drinks (doesn't functionally determine money) so:
{drink_color}+ = {drink_color, flavour}
ColorFlavour(drink_color, flavour)
Drinks(drink_color, money)

FD iv. is violated. Example: Large orange costs $1, it is stored as SizeforMoney($1, Large), ColorFlavour(orange, orange), Drinks(Orange, $1). Drinks(Orange, $2) is also valid, and there could as well be SizeforMoney($2, Large) but now we don't know how much should a large orange drink cost.

e. 3NF:
money → drink_size: money is not a superkey for drinks but drink_size is part of a candidate key.
drink_color → flavour: drink_color is not a superkey, but flavour is part of a candidate key.
flavour → drink_color: flavour is not a superkey, but drink_color is part of a candidate key.
drink_size flavour → money: {drink_size, flavour} is our primary key.
No decomposition necessary. Primary key is any of the candidate keys from c.

f. DrinkPrice(drink_size, flavour, money)
DrinkColor(flavour, drink_colour)
We don't need to duplicate the drink colour for every drink size, when flavour uniquely determines it.

4. Now you are a sporting goods store owner. You have the following relations:
Equipment(name:string, category:string, quantity:int, price:float)
SalesPersons(name:string, address:string, phone:int)
Sold(salesperson:string, item:string, saletime:date)

a. Write SQL code to create tables with the attribute and domain names specified in the relational schema. Don't forget to indicate primary keys and foreign keys.

b. Write a SQL query to find all item names sold on the date ‘2006-06-30’.

c. Write a SQL query to find the item categories sold on ‘2006-06-30’.

d. Write a SQL query to show the name and money from sales that each sales person generated for the month of June (06) in 2006. The column headers should be salesperson_name and total_sales.

Answer:

a.
```
CREATE TABLE Equipment (
        name VARCHAR(20) PRIMARY KEY,
        category VARCHAR(20),
        quantity INT,
        price FLOAT);

CREATE TABLE SalesPersons (
        name VARCHAR(20) PRIMARY KEY,
        address VARCHAR(50),
        phone INT);

CREATE TABLE Sold (
        salesperson VARCHAR(20) REFERENCES SalesPersons(name),
        item VARCHAR(20) REFERENCES Equipment(name),
        saletime Date,
        PRIMARY KEY (item));
```

b.
```
SELECT item
FROM Sold
WHERE saletime = DATE ‘2006-06-30’
```

c.
```
SELECT category
FROM Sold JOIN Equipment ON item=name
WHERE saletime = DATE ‘2006-06-30’
```

d.
```
SELECT salesperson AS salesperson_name, SUM(price) AS total_sales
FROM Sold JOIN Equipment ON item=name
WHERE saletime BETWEEN DATE ‘2006-06-01’ AND DATE ‘2006-06-31’
GROUP BY salesperson
```