January 16, 2013 Activities CS 386/586

Experimenting with bags (tables and query answers with duplicate rows):

Write an SQL query (and run it against the sailors database) that does the following:

1. List every row in the boats table.

   Are there any duplicate rows? (You may think that this is a mistake; but there are some situations where a table may very well have duplicate rows. This table allows us to experiment with duplicate rows.)

   This allows you to see that a basic SQL query does NOT eliminate duplicates. It shows you the duplicate rows in the query answer. (There is a way to ask SQL to eliminate duplicates from a query answer using the word distinct; we'll see more about that down below.)

2. List every row from the boats table where the color is not red. Are there any duplicate rows in the query answer? (This shows you what SQL does when you issue a query that is equivalent to a single select operator (from relational algebra on bags).

3. List every combination of one boat and one reservation. Before you run the query, estimate the number of rows that you will have in the query answer. After you run the query, notice the names of all the attributes. Is there an attribute name that appears more than once? (This shows you what SQL does with a simple cross product – even if there is ambiguity in the attribute names.)
   Does this query answer have any duplicate rows? (This shows you what SQL does with the cross product operator when there are bags, i.e., tables with duplicates, as input tables.)

4. Extend the query from item 3 so that the only combinations (of a boat and a reservation) shown in the query answer are where the boat id in the reservation matches the boat id in the boat table. (Do not use a join clause.)

   Try writing this query without using any correlations names (e.g., aliases or short names) for the tables.

   Try writing this query with just one correlation name (for one of the two tables).

   Try writing this query with a correlation name for both of the two tables.

   Notice the attributes that appear in the query answer. Are they any different from the attributes that appeared in your answer to query from item 3?

   Try writing this query where you list the two tables involved in the opposite order. Is there any

difference in the attributes that appear in the query answer? Do the same rows appear in these two variations of the query? Note: SQL makes no promise about the order that the tuples appear in a query answer. As long as the proper rows are returned in the query answer – in any order – then the query answer is deemed to be correct. (There is a way to control the order of the rows in the query answer – using the order by clause; we'll talk about that next week.)

Notice the duplicates in the query answer. Does it make sense – what you see as duplicates (knowing which rows are in the boats table and in the reserves table)?

5. Write an SQL query (without using a join clause in the from clause) that lists the sailor id, the sailor name, the boat id, and the boat name where the sailor has reserved the boat. Explain in English what it means if there are duplicate rows in the query answer.

6.

Experimenting with joins – including self-joins – without using the join clause

7. Run the following query:

```
select b.bid, b.bname, r1.day, r2.day
from  boats b, reserves r1, reserves r2
where b.bid = r1.bid and b.bid = r2.bid and r1.day < r2.day
```

Describe in English what this query computes.

What happens if you leave out the second part of the where clause (i.e., leave out the "r1.day < r2.day")? (You can run the query to see what happens.)

What happens if you leave out the entire WHERE clause?

Contrast the following query with the one above:

```
select b.bid, b.bname, r1.day
from  boats b, reserves r1
where b.bid = r1.bid
```

Describe in English what this query computes.

8. Run the following query (which is a modification of the first query in item 8 – with fewer attributes in the final query answer):

```
select b.bid, b.bname
from  boats b, reserves r1, reserves r2
where b.bid = r1.bid and b.bid = r2.bid and r1.day < r2.day
```

Describe in English what this query computes.

Now, try this:

```
select distinct b.bid, b.name
from boats b, reserves r1, reserves r2
where b.bid = r1.bid and b.bid = r2.bid and r1.day < r2.day
```

Explain in English what this query computes.

9. Write and run an SQL query (modeled after the query above) that finds all boats that have at least three reservations.

Experimenting with the join clause including left, right and full outer joins.

10. Rewrite and run the query from item 4 using a join clause in the from clause. Your query answer should be identical to the answer you got for the query in item 4. (The order of the rows might be different, as explained above.)

11. Rewrite and run the query from item 5 using two join clauses in the from clause. Your query answer should be identical to the answer you got the query in item 5.

12. Rewrite and run the query from item 6 using the join clause in the from clause. Note: you'll have to use two join clauses.

13. Item 6 and item 11 asked you to write a query to join reserves and boats.

    Write and then run a similar query to compute the left outer join of reserves and boats, matching on bid. Explain in English what this query computes.

    Write and run a similar query to compute the right outer join of reserves and boats, matching on bid. Explain in English what this query computes.

    Write and run a similar query (as the one you just wrote) but change the order of the two tables listed in the from clause (but still use a right outer join). What is the difference between this query and the previous one? What is the difference between this query and the first one you wrote for this item (Item 12)?

Write a similar query to compute the full outer join of reserves and boats, matching on bid. Explain what this query computes in English.

Identify a row that appears in all three answers (the left, right, and full outer join) and explain why it appears in all three answers.

Is there a row in the query answer to the full outer join that does NOT appear in any of the other queries? Explain why this is so (or not so).

14. In general, what is the difference between computing the cross product of two tables and computing the full outer join of the same two tables (where you join for equality on some attribute that appears in both tables)? Explain.

    Does a full outer join contain rows that do not appear in a cross product? Does a cross product contain rows that do not appear in a join?

    Make sure you consider the case where the full outer join includes data in the original tables where at least one row from the first (left) table does not match any rows (in the second table) and where there is at least one row from the second (right) table does not match any rows (in the first table). For this experiment, you should insert data into the tables you created last week – to play around with left, right, and full outer join.