Lecture 9 Notes - Monday 10/24/16

Reading Quiz

Question 1: Ans = C
Question 2: Ans = D
Question 3: Ans = A
Question 4: Ans = E (or possibly D, since we didn't specify whether or not we have a *using* clause)
Question 5: Ans = D

Notes

To find out who was enrolled in CS327E last Spring (*RDBMS = Joins* slide) we want to use an inner join, because we don't care to have information on other students or other classes. We can do the join in either "direction."

Concept Question 1: E. On the "first" join (*Student s **inner join** Enrollment e*), we would get the additional results of all the students without enrollments, because we would keep those rows in *Student* which didn't have corresponding rows in *Enrollment*. However, when we take these results and further join them to the *Class* table, these extra students get filtered out, because we are requiring a corresponding *unique* value, which these extra students don't have (their values for *unique* will be NULL), as they're not in *Enrollment*. So in the end we get the same results using the **left outer join** as with the **inner join**.

Concept Question 2: C. *A* will just gives us all the candidates skilled in either MySQL or Python -- there is no guarantee that they will be skilled in both. *D* does the same thing as *A*. *C* will give us the *candidate_id*s for those candidates who are skilled in both Python and MySQL, which is what we want. Note there will be unused records that are "duplicates" in some sense -- for each candidate having *c1.skill_code = 'MySQL'* and *c2.skill_code = 'Python'*, that candidate will also have a row in the join with *c2.skill_code = 'MySQL'* and *c1.skill_code = 'Python'*, so we just pick one of these combinations to get the *candidate_id*s. *B* technically works too, but note that it doesn't use MySQL to get these results. This is fine, but if you have a large amount of data, this can become inefficient.

Concept Question 3: C. Since we no longer have a foreign key in *account* referencing *cust_id* in *customer*, we are able to add accounts to *account* which do not necessarily have owners (corresponding rows in *customer*). This is what we mean by "orphan accounts." To find the orphan accounts, we need to join *account* with *cust_acct* and find all rows that have accounts but no corresponding customers. *C* accomplishes this using a **left outer join** to retain accounts that lack corresponding *cust_acct* rows, then filters so that we keep only such accounts (those rows in the joined table with *ca.acct_id* being NULL). *A* and *B* will perform **inner join**s and will thus lose the orphan accounts.

(D will not catch the orphan accounts; this **right outer join** will end up the same as an **inner join** since all rows in *cust_acct* necessarily have corresponding rows in *account*. Filtering the joined table in *D* for *ca.acct_id is null* will result in an empty table.)

Note that the query in answer *D* has been changed since the lecture so that it now catches all “orphan customers.”