Agenda

- Readings for today
- Reading Quiz
- Concept Questions
- Homework for next time
Homework for Today

• Chapter 10 from the Learning SQL book
• Exercises at the end of Chapter 10
How many **rows** does the following query return?

SELECT *
FROM customer c LEFT OUTER JOIN account a
ON c.cust_id = a.cust_id;

A. 3  B. 4  C. 5  D. 6
Suppose we execute the query:

```sql
SELECT e.fname, e.lname, d.name
FROM employee e
LEFT OUTER JOIN department d
on e.dept_id = d.dept_id;
```

This is one row from the result set:

```
<table>
<thead>
<tr>
<th>fname</th>
<th>lname</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Gooding</td>
<td>???????????????</td>
</tr>
</tbody>
</table>
```

What is ??????????????????

A. <Blank>  
B. NULL  
C. 0  
D. N/A. The query is syntactically incorrect and results in an error.
Quiz Question 3

mysql> select * from employee;
+-------------------+--------+---------+---------------+
| emp_id | fname   | lname   | superior_emp_id |
|---------+---------+---------+---------------+-------------------+--------+---------+---------------+
| 1       | Michael | Smith   | NULL          |
| 2       | Susan   | Barker  |               1 |
| 3       | Robert  | Tyler   |               1 |
| 4       | Susan   | Hawthorne |               3 |
+-------------------+--------+---------+---------------+

Select the best answer.

A. Query 1 returns more rows than Query 2.
B. Query 2 returns more rows than Query 1.
C. Query 1 and Query 2 both return the same number of rows.
D. Either Query 1 or Query 2 (or both) are syntactically incorrect.

Query 1
SELECT * FROM employee e
INNER JOIN employee emgr
WHERE e.superior_emp_id = emgr.emp_id;

Query 2:
SELECT * FROM employee e
LEFT OUTER JOIN employee emgr
ON e.superior_emp_id = emgr.emp_id;
Quiz Question 4

What happens when you perform a NATURAL JOIN on two tables with no identical column names?

A. It is equivalent to performing an INNER JOIN
B. It is equivalent to performing a LEFT OUTER JOIN
C. It is equivalent to performing a RIGHT OUTER JOIN
D. It is equivalent to performing a Cartesian product or CROSS JOIN
E. None of the above
Quiz Question 5

Consider the following queries on some table Foo with column val:

Q1: SELECT * FROM Foo a INNER JOIN Foo b WHERE a.val = b.val;
Q2: SELECT * FROM Foo a LEFT OUTER JOIN Foo b ON a.val = b.val;
Q3: SELECT * FROM Foo a RIGHT OUTER JOIN Foo b ON a.val = b.val;

Which of the following statements is true?
A. The number of rows from Q1 is always > the number of rows from Q2
B. The number of rows from Q1 is always > the number of rows from Q3
C. The number of rows from Q2 is always > the number of rows from Q3
D. The number of rows from Q3 is always > the number of rows from Q2
E. None of the above
Here is a view of the bank schema from our book. From this diagram, what can you tell about the relationship between a customer, an individual, and a business?

A. A customer is one or more individuals
B. A customer is one or more businesses
C. A customer is either one or more individuals or one or more businesses
D. A customer is either a single business or a single individual
E. None of the above
How can we extend the bank schema to support a **joint** account that is owned by multiple customers?

A. Model **account** and **customer** tables as many-to-many with junction table

B. Combine **customer** and **individual** tables

C. Combine **account** and **customer** tables

D. Model **customer** and **individual** tables as many-to-many with junction table

E. Model **customer** and **business** tables as many-to-many with junction table
New table definitions:

create table account(
    account_id INT(10) primary key AUTO_INCREMENT,
    product_cd VARCHAR(10) NOT NULL,
    cust_id INT(10) NOT NULL,
    open_date DATE NOT NULL,
    close_date DATE DEFAULT NULL,
    ...
)

CREATE TABLE customer(
    cust_id INT(10) primary key AUTO_INCREMENT,
    fed_id VARCHAR(12) NOT NULL,
    cust_type_cd ENUM('I', 'B') NOT NULL,
    address VARCHAR(30),
    ...
)

CREATE TABLE cust_acct(
    acct_id INT(10),
    cust_id INT(10),
    constraint pk_cust_acct primary key (acct_id, cust_id),
    constraint fk_account_id foreign key (acct_id)
        references account (acct_id),
    constraint fk_cust_id foreign key (cust_id)
        references customer (cust_id))
Concept Question 3

Now that we have established a many-to-many relationship between the account and customer entities, we need to watch out for “orphan” accounts, namely accounts which belong to no customers. Which of these queries will find all orphan accounts in the bank database?

A. select a.account_id, ca.acct_id from account a join cust_acct ca on a.account_id = ca.acct_id where ca.acct_id is not null

B. select a.account_id, ca.acct_id from account a join cust_acct ca on a.account_id = ca.acct_id where ca.acct_id is null

C. select a.account_id, ca.acct_id from account a left outer join cust_acct ca on a.account_id = ca.acct_id where ca.acct_id is null

D. select a.account_id, ca.acct_id from account a right outer join cust_acct ca on a.account_id = ca.acct_id where ca.acct_id is null
The Registrar’s Office needs help finding all current classes that have no students enrolled. Which query will compute this answer?

A. `select c.ClassID, c.Course
from enrollment e left outer join classes c
on e.ClassID = c.ClassID
where c.ClassID is null and c.StartDate = '2016-01-19'`

B. `select c.ClassID, c.Course
from enrollment e right outer join classes c
on e.ClassID = c.ClassID
where e.ClassID is null and c.StartDate = '2016-01-19'`

C. `select c.ClassID, c.Course
from enrollment e full outer join classes c
on e.ClassID = c.ClassID
where c.StartDate = '2016-01-19'`

D. `select c.ClassID, c.Course
from enrollment e join classes c
on e.ClassID = c.ClassID
where e.ClassID is null and c.StartDate = '2016-01-19'`

E. None of the above
Concept Question 5

Consider the Member and Locker tables in the Rec Center’s database. Suppose we want to see a list of all the members and their assigned locker, including those who have not been assigned to a locker. In the same report, we also want to see a list of all the lockers, including those that have not been assigned to a member. What SQL query will compute this answer?

A. `select m.member_id, l.locker_number`  
   `from Member m left outer join Locker l`  
   `on m.locker_number = l.locker_number`

B. `select m.member_id, l.locker_number`  
   `from Member m right outer join Locker l`  
   `on m.locker_number = l.locker_number`

C. `select m.member_id, l.locker_number`  
   `from Member m full outer join Locker l`  
   `on m.locker_number = l.locker_number`

D. `select m.member_id, l.locker_number`  
   `from Member m inner join Locker l`  
   `on m.locker_number = l.locker_number`
Concept Question 6

The landlord of an apartment complex would like to know who has paid their rent this month. He wants to see a report of all apartment units, tenants, and rent payments, including units with no tenants and tenants who have not paid rent. The time period for the report should be 02/01/16 – 02/08/16.

A. select u.unit_nbr, t.tenant_fname, t.tenant_lname, rp.payment_date
from Units u left outer join Tenants t
on u.unit_nbr = t.unit_nbr
left outer join RentPayments rp
on (t.tenant_id = rp.tenant_id
and u.unit_nbr = rp.unit_nbr)
where rp.payment_date
between '2016-02-01' and '2016-02-08'
or rp.payment_date is null

B. select u.unit_nbr, t.tenant_fname, t.tenant_lname, rp.payment_date
from RentPayments rp
left outer join Tenants t on
(t.tenant_id = rp.tenant_id
left outer join Units u
on (rp.unit_nbr = u.unit_nbr
and u.unit_nbr = t.unit_nbr)
where rp.payment_date
between '2016-02-01' and '2016-02-08'
or rp.payment_date is null

C. None of the above

Table definitions:

create table Units(
    unit_nbr integer primary key,
    unit_size double,
    floor integer,
    is_furnished enum('Y', 'N') default 'N',
    rental_price double);

create table Tenants(
    tenant_id integer primary key,
    tenant_fname varchar(30) not null,
    tenant_lname varchar(30) not null,
    move_in_date date,
    move_out_date date,
    vacated_date date,
    unit_nbr integer not null,
    foreign key (unit_nbr) references Units(unit_nbr));

create table RentPayments(
    payment_id integer primary key,
    payment_date date,
    payment_amount double,
    tenant_id integer not null,
    unit_nbr integer not null,
    foreign key (tenant_id) references Tenants(tenant_id),
    foreign key(unit_nbr) references Units(unit_nbr));
We have a table `Credits` that represents students and the courses they have taken in college. We would like to see how far each student has gone in his/her degree program. However, a student cannot receive credit for a course until he/she has met the prerequisites for that course. Assume that we have only 3 courses, `cs101e`, `cs102e`, and `cs103e`. Also, assume that `cs101e` has no pre-requisites, `cs102e`’s prerequisite is `cs101e` and `cs103e`’s prerequisite is `cs102e`. Which SQL join operators produces the desired output?

**Table definition:**

```sql
create table Credits(
    student_id CHAR(8),
    course_name CHAR(6),
    primary key(student_id, course_name));
```

**Sample input:**

<table>
<thead>
<tr>
<th>student_id</th>
<th>course_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>'adam1'</td>
<td>'cs101e'</td>
</tr>
<tr>
<td>'adam1'</td>
<td>'cs102e'</td>
</tr>
<tr>
<td>'lee5'</td>
<td>'cs101e'</td>
</tr>
<tr>
<td>'wsmith'</td>
<td>'cs102e'</td>
</tr>
<tr>
<td>'wsmith'</td>
<td>'cs103e'</td>
</tr>
</tbody>
</table>

**Desired output:**

<table>
<thead>
<tr>
<th>student_id</th>
<th>course_name1</th>
<th>course_name2</th>
<th>course_name3</th>
</tr>
</thead>
<tbody>
<tr>
<td>adam1</td>
<td>cs101e</td>
<td>cs102e</td>
<td>cs103e</td>
</tr>
<tr>
<td>lee5</td>
<td>cs101e</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
Solution for Concept 7

**SQL Query:**

```sql
select c1.student_id, c1.course_name as course_name1, c2.course_name as course_name2, c3.course_name as course_name3
from Credits c1 left outer join Credits c2
on (c1.student_id = c2.student_id
    and c1.course_name <> c2.course_name)
left outer join Credits c3
on (c2.student_id = c3.student_id
    and c2.course_name <> c3.course_name)
where c1.course_name = 'cs101e'
and (c2.course_name = 'cs102e' or c2.course_name is null)
and (c3.course_name = 'cs103e' or c2.course_name is null)
```

**Desired output:**

```
<table>
<thead>
<tr>
<th>student_id</th>
<th>course_name1</th>
<th>course_name2</th>
<th>course_name3</th>
</tr>
</thead>
<tbody>
<tr>
<td>adam1</td>
<td>cs101e</td>
<td>cs102e</td>
<td>cs103e</td>
</tr>
<tr>
<td>lee5</td>
<td>cs101e</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```
Homework for Next Time

• Read chapter 8 from the Learning SQL book
• Exercises at the end of chapter 8