Lecture 23:

- Outer Joins
- SQL Set Operations

Announcements:

- PS-4 due
- Project Step 5 due thurs (brief status updated)
- HW-7 out (due next tues)
- PS-5 out (due next thurs)

---

Running Example

**Schema:**

```plaintext
Customer(c_num, name, addr, c_rating, c_amount, c_bal, sp_num)
Salesperson(sp_num, name, address, office)
```

with FK: `customer.sp_num → salesperson.sp_num`

**Example Customer instance**

<table>
<thead>
<tr>
<th>c_num</th>
<th>name</th>
<th>address</th>
<th>c_rating</th>
<th>c_amount</th>
<th>c_balance</th>
<th>sp_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>xxx</td>
<td>5</td>
<td>1000</td>
<td>1000</td>
<td>101</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>yyy</td>
<td>7</td>
<td>5000</td>
<td>4000</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>Chuck</td>
<td>zzz</td>
<td>10</td>
<td>10000</td>
<td>1000</td>
<td>102</td>
</tr>
</tbody>
</table>
Outer Joins

The basic join is called an **inner join**

- The default used with the JOIN keyword
- Can also write an inner join this way:

```
SELECT *
FROM customer c INNER JOIN salesperson s ON (c.sp_num = s.sp_num);
```

- in this join, **customer** is the "left" join relation
- and **salesperson** is the "right" join relation

Note that INNER JOIN answers include all matches, but exclude:

- Customer rows that do **not** have a Salesperson
- Salesperson rows that are **not** assigned any Customers

---

Outer Joins

An **outer join** includes the “non-matches” ... for left and/or right relations

**LEFT OUTER JOIN** includes all matches plus:

- Customers that do **not** have a Salesperson

**RIGHT OUTER JOIN** includes all matches plus:

- Salespeople that are **not** assigned to any customers

**FULL OUTER JOIN** includes all of these

The “missing” attribute values in result are assigned **NULL**

**Note**: can be abbreviated to LEFT JOIN, RIGHT JOIN, FULL JOIN
Inner vs Outer Joins

<table>
<thead>
<tr>
<th>c_num</th>
<th>name</th>
<th>sp_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>Bob</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>Chuck</td>
<td>NULL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sp_num</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>Dave</td>
</tr>
<tr>
<td>65</td>
<td>Eddy</td>
</tr>
<tr>
<td>75</td>
<td>Fionna</td>
</tr>
</tbody>
</table>

customer c INNER JOIN salesperson s ON (c.sp_num = s.sp_num) gives:

1. Alice, 55, 55, Dave
2. Bob, 65, 65, Eddy

customer c LEFT OUTER JOIN salesperson ON (c.sp_num = s.sp_num) gives:

1. Alice, 55, 55, Dave
2. Bob, 65, 65, Eddy
3. Chuck, NULL, NULL, NULL

customer c RIGHT OUTER JOIN salesperson s ON (c.sp_num = s.sp_num) gives:

1. Alice, 55, 55, Dave
2. Bob, 65, 65, Eddy
   NULL, NULL, NULL, 75, Fionna

customer c FULL OUTER JOIN salesperson s ON (c.sp_num = s.sp_num) gives:

1. Alice, 55, 55, Dave
2. Bob, 65, 65, Eddy
3. Chuck, NULL, NULL, NULL
   NULL, NULL, NULL, 75, Fionna
### Bank Account Example Tables

#### Account
<table>
<thead>
<tr>
<th>acct_num</th>
<th>owner</th>
<th>balance</th>
<th>acct_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Alice</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>102</td>
<td>Bob</td>
<td>2000.00</td>
<td>checking</td>
</tr>
<tr>
<td>103</td>
<td>Alice</td>
<td>5000.00</td>
<td>savings</td>
</tr>
<tr>
<td>104</td>
<td>Chuck</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>105</td>
<td>Debbie</td>
<td>10000.00</td>
<td>NULL</td>
</tr>
<tr>
<td>106</td>
<td>Bob</td>
<td>7000.00</td>
<td>savings</td>
</tr>
</tbody>
</table>

#### Deposit
<table>
<thead>
<tr>
<th>acct_num</th>
<th>transaction_id</th>
<th>deposit_date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>1</td>
<td>10/22/18</td>
<td>500.00</td>
</tr>
<tr>
<td>102</td>
<td>2</td>
<td>10/29/18</td>
<td>200.00</td>
</tr>
<tr>
<td>104</td>
<td>3</td>
<td>10/29/18</td>
<td>1000.00</td>
</tr>
<tr>
<td>105</td>
<td>4</td>
<td>11/2/18</td>
<td>10000.00</td>
</tr>
</tbody>
</table>

#### Withdraw
<table>
<thead>
<tr>
<th>acct_num</th>
<th>check_num</th>
<th>check_date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>924</td>
<td>10/23/18</td>
<td>125.00</td>
</tr>
<tr>
<td>101</td>
<td>925</td>
<td>10/24/18</td>
<td>23.98</td>
</tr>
<tr>
<td>102</td>
<td>128</td>
<td>10/29/18</td>
<td>200.00</td>
</tr>
<tr>
<td>102</td>
<td>131</td>
<td>10/30/18</td>
<td>800.00</td>
</tr>
</tbody>
</table>

---

### SQL Set Operations

#### Set-based operators
- \( \cup \) (union)
- \( \cap \) (intersect)
- \( \setminus \) or \( - \) (difference)

**Check In:** what is returned for \( S_1 = \{1, 3, 5, 7\} \) and \( S_2 = \{1, 2, 3, 4\} \)
- \( S_1 \cap S_2 = \{1, 3\} \)
- \( S_1 \cup S_2 = \{1, 2, 3, 4, 5, 7\} \)
- \( S_1 - S_2 = \{5, 7\} \)
- \( S_2 - S_1 = \{2, 4\} \)

**SQL set operators can only be used with “union compatible” relations**
- relations have the same arity (number of attributes)
- and corresponding attributes have compatible data types (domains)
SQL UNION

Examples:

(SELECT acct_num, owner FROM account WHERE balance >= 10000)
UNION
(SELECT acct_num, owner FROM account WHERE balance <= 500);

Check In: What does this return? Can it be (re)written without UNION?

(SELECT acct_num, amount FROM deposit WHERE amount >= 500)
UNION
(SELECT acct_num, amount FROM withdraw WHERE amount >= 500);

Check In: What does this return? Can it be (re)written without UNION?

SQL INTERSECT

Example:

(SELECT owner FROM account WHERE acct_type = 'checking')
INTERSECT
(SELECT owner FROM account WHERE acct_type = 'savings');

Check In: What does this return? Can it be (re)written without INTERSECT?

SELECT DISTINCT a1.owner
FROM account a1 JOIN account a2 USING (owner)
WHERE a1.acct_type = 'checking' AND a2.acct_type = 'savings';

Note: Can also use an EXISTS subquery, etc.
SQL EXCEPT (set difference)

Example:

(SELECT owner FROM account WHERE acct_type = 'checking')
EXCEPT
(SELECT owner FROM account WHERE acct_type = 'savings')

Check In: What does this return?

Check In: Is this an equivalent query?

SELECT DISTINCT a1.owner
FROM account a1, account a2
WHERE a1.acct_type = 'checking' AND a2.acct_type = 'savings' AND
    a1.owner != a2.owner;

Need to use a subquery (e.g., NOT EXISTS)

Duplicate elimination vs ALL in SQL

By default, set ops remove duplicates ...

MySQL eliminates duplicates then computes set operation

• Which is different than multiset set op followed by duplicate elimination
• E.g., \{1, 2, 2, 2, 3\} − \{2, 3\} = \{1, 2, 2\} but \{1\} in MySQL

To return all duplicate answers use ALL

\{1, 1, 2\} UNION ALL \{1, 2, 3\} = \{1, 1, 1, 2, 2, 3\} ... in SQL

• Different than multiset-union, which returns the max of two relations