Today

- Quiz 3
- Joins, Nulls

Assignments

- HW3 due
- HW4 out, due Tues
Example Tables

account

<table>
<thead>
<tr>
<th>number</th>
<th>owner</th>
<th>balance</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>J. Smith</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>102</td>
<td>W. Wei</td>
<td>2000.00</td>
<td>checking</td>
</tr>
<tr>
<td>103</td>
<td>J. Smith</td>
<td>5000.00</td>
<td>savings</td>
</tr>
<tr>
<td>104</td>
<td>M. Jones</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>105</td>
<td>H. Martin</td>
<td>10000.00</td>
<td>checking</td>
</tr>
</tbody>
</table>

deposit

<table>
<thead>
<tr>
<th>account</th>
<th>transaction_id</th>
<th>date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>1</td>
<td>10/22/11</td>
<td>500.00</td>
</tr>
<tr>
<td>102</td>
<td>2</td>
<td>10/29/11</td>
<td>200.00</td>
</tr>
<tr>
<td>104</td>
<td>3</td>
<td>10/29/11</td>
<td>1000.00</td>
</tr>
<tr>
<td>105</td>
<td>4</td>
<td>11/2/11</td>
<td>10000.00</td>
</tr>
</tbody>
</table>

check

<table>
<thead>
<tr>
<th>account</th>
<th>check_number</th>
<th>date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>924</td>
<td>10/23/11</td>
<td>125.00</td>
</tr>
<tr>
<td>101</td>
<td>925</td>
<td>10/24/11</td>
<td>23.98</td>
</tr>
<tr>
<td>102</td>
<td>746</td>
<td>10/25/11</td>
<td>51.73</td>
</tr>
</tbody>
</table>

FK’s:

- deposit.account → account.number
- check.account → account.number
Recap: Where we are / Where we’re going

We covered the basic forms of SELECT, FROM, WHERE

Now we’re going to dive more into each of these and introduce some new clauses
More SQL: JOIN expressions

SQL supports various types of joins

- So far, we’ve looked at “comma” joins (a kind of “inner” join)
- we’ll look at the inner joins now, then outer joins later

Special syntax for inner joins

- These queries are equivalent

    -- comma join syntax
    SELECT a.owner, d.date, d.amount
    FROM account a, deposit d
    WHERE a.number = d.account and a.type = 'checking';

    -- inner join syntax
    SELECT a.owner, d.date, d.amount
    FROM account a INNER JOIN deposit d ON a.number = d.account
    WHERE a.type = 'checking';

    -- default join is an inner join
    SELECT a.owner, d.date, d.amount
    FROM account a JOIN deposit d ON a.number = d.account
    WHERE a.type = 'checking';
JOIN with USING if tables have same attribute names

SELECT d.account, d.amount, c.number
FROM deposit d JOIN check c USING (account);

- Can also include multiple join attributes:

SELECT d.account, d.amount, c.number
FROM deposit d JOIN check c USING (account, date);

- or ...

SELECT d.account, d.amount, c.number
FROM deposit d JOIN check c ON
  d.account=c.account AND d.date=c.date;
Exercise: Rewrite the following queries using JOIN syntax

Query 1:

```sql
SELECT a.name, a.balance
FROM account a, deposit d
WHERE a.number = d.account AND a.balance > 1000;
```

– In this case: `ON a.number = d.account`

Query 2:

```sql
SELECT a1.owner, a1.number, a1.balance, a2.number, a2.balance
FROM account a1, account a2
WHERE a1.owner = a2.owner AND a1.balance > a2.balance
```

– In this case: `USING (owner)`
Multiple Joins

```
SELECT *
FROM deposit d JOIN check c USING (account)
    JOIN account a ON c.account = a.number
```

Natural Joins

```
SELECT *
FROM deposit NATURAL JOIN check
```

- joins on same-named attributes between two tables

Cross Joins

```
SELECT *
FROM deposit CROSS JOIN check
```

- this is the same as:

```
SELECT *
FROM deposit, check
```

We'll look at outer joins later ...
SQL Null values

**NULL is a special value in SQL**

- Indicates the value is **unknown**: either **missing** or **does not exist**
  - The car can be purchased new, but we don’t know the retail price (missing)
  - The car cannot be purchased new, and so doesn’t have a retail price (does not exist)
  ... In general, can stand in for many different cases

**Handling unknown (NULL) values in SQL can be tricky**

- WHERE only selects conditions that are True
  - E.g., What is the result of this query?
    ```sql
    SELECT *
    FROM account
    WHERE NULL
    ```
  - It is always empty!

- Comparisons involving NULL are always Unknown — return NULL
  - E.g., What is the result of this query?
    ```sql
    SELECT *
    FROM account
    WHERE type != NULL
    ```
  - Always returns the empty set (because type != NULL returns NULL)
• SQL provides IS NULL and IS NOT NULL comparators

  – E.g., What is the result of this query?

    ```sql
    SELECT *
    FROM account
    WHERE type IS NOT NULL
    ```

  – All accounts whose type attribute is something other than NULL

• Logical connectives based on “3-valued” logic (“Unknown” implies NULL):

<table>
<thead>
<tr>
<th>X AND Y</th>
<th>True</th>
<th>False</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>Unknown</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>Unknown</td>
<td>Unknown</td>
<td>False</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X OR Y</th>
<th>True</th>
<th>False</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>Unknown</td>
</tr>
<tr>
<td>Unknown</td>
<td>True</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT Y</th>
<th>True</th>
<th>False</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

• You can try these out in SQL (0=False, 1=True):

    ```sql
    SELECT 1 AND NULL;
    +------------+
    | 1 AND NULL |
    +------------+
    | NULL       |
    +------------+
    1 row in set (0.00 sec)
SELECT 0 AND NULL;
+------------+
| 0 and null |
+------------+
| 0          |
+------------+
1 row in set (0.00 sec)

SELECT NULL OR 0;
+-----------+
| NULL or 0  |
+-----------+
| NULL      |
+-----------+
1 row in set (0.00 sec)

SELECT NOT NULL;
+-----------+
| not NULL  |
+-----------+
| NULL      |
+-----------+
1 row in set (0.00 sec)

SELECT (NOT NULL) IS NULL;
+-----------------------+
| (not NULL) is NULL    |
+-----------------------+
| 1                     |
+-----------------------+
1 row in set (0.00 sec)