Today

- SQL Queries (cont)

Assignments

- HW3 out, due Tues
Example Tables

account

<table>
<thead>
<tr>
<th>account 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 number</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 number</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>
</tbody>
</table>

FK's:

- deposit.account \(\rightarrow\) account.number
- check.account \(\rightarrow\) account.number
Querying Multiple Tables ("Joins")

Example

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

- How should this work?
- Which rows from which tables are evaluated in the WHERE clause?

The query uses **correlation names**

- Like a local variable
- Variable "holds" (bound to) one row from corresponding table
- Only needed when two tables have the same attribute names
  - Although often helps when reading queries
- A table name can also be used as a correlation name
  - e.g., account.name or deposit.account
  - When would just table name alone not be enough? ... "self join"
- Note correlation names may not be needed, e.g.:

```sql
SELECT *
FROM account, deposit
```
To answer the query we check every combination of rows

- For each account row, examine each deposit row
- E.g., the first account and deposit rows:
  
  101, J. Smith, 1000.00, checking, 102, 1, 10/22/11, 500.00

- Does this row satisfy the WHERE clause?
  
  – No
  
  – The account number (101) is not the same as the deposit account (102)
  – Also, account balance is not greater than 1000

Query answer:

<table>
<thead>
<tr>
<th>owner</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>W. Wei</td>
<td>2000.00</td>
</tr>
<tr>
<td>W. Wei</td>
<td>2000.00</td>
</tr>
<tr>
<td>H. Martin</td>
<td>10000.00</td>
</tr>
</tbody>
</table>

- Note that the answer contains a duplicate row
This query is an example of a **join**

- In this case, an implicit **equi-join**
- Combining two tables based on values they have in common
- Equi-join since checking equality of account.number and deposit.account
- We’ll talk about other types of joins and join syntax later
Query Example 7

Use the `DISTINCT` keyword to remove duplicates in query results

- E.g., the query
  
  ```sql
  SELECT DISTINCT a.name, a.balance
  FROM accounts a, deposits d
  WHERE a.number = d.account AND
    a.balance > 1000;
  ```

- Returns

<table>
<thead>
<tr>
<th>owner</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>W. Wei</td>
<td>2000.00</td>
</tr>
<tr>
<td>H. Martin</td>
<td>10000.00</td>
</tr>
</tbody>
</table>
Query Example 8

Another example of a join

SELECT a.number, a.owner, d.date, d.amount
FROM accounts a, deposits d
WHERE a.number = d.account AND d.amount > 300

• What is the arity of the answer?
• What is the cardinality of the answer?
Query Example 9

A **self join** joins a table with itself

```
SELECT a1.owner, a1.number, a1.balance, a2.number, a2.balance
FROM   accounts a1, accounts a2
WHERE  a1.owner = a2.owner
```

- Here `a1` and `a2` refer to copies of the same instance
- What is this query trying to do?
- What is the answer?
Query Example 10 & 11

We can rename attributes in query answers

```
SELECT owner, balance AS checking_balance
FROM account
WHERE type = 'checking'
```

- What does this query return?

We can also save the result as a new table

```
CREATE TABLE checking_account
  SELECT owner, balance AS checking_balance
  FROM account
  WHERE type = 'checking'
```

- We can use the new table in subsequent queries
Comparison operators ...

- standard comparison operators (Note: 0 is false, 1 is true):
  - \( x = y \) ... true if \( x \) equal to \( y \)
  - \( x \neq y \) or \( x <> y \) ... true if \( x \) not equal to \( y \)
  - \( x < y \) ... true if \( x \) less than \( y \)
  - \( x > y \) ... true if \( x \) greater than \( y \)
  - \( x \geq y \) ... true if \( x \) greater than or equal to \( y \)
  - \( x \leq y \) ... true if \( x \) less than or equal to \( y \)

- can test these in MySQL:

  ```sql
  SELECT 'a' = 'a';
  +-----------+ +-----------+
  | 'a' = 'a' | | 'a' = 'a' |
  +-----------+ +-----------+
  | 1 | | 0 |
  +-----------+ +-----------+
  
  SELECT ('a' != 'a') = 0;
  +------------------+
  | ('a' != 'a') = 0 |
  +------------------+
  | 1 |
  +------------------+
  
  SELECT ('a' != 'a') IS FALSE;
  +------------------+
  | ('a' != 'a') IS FALSE |
  +------------------+
  | 1 |
  +------------------+
  
  SELECT 'a' != 'a';
  +-----------+ +-----------+
  | 'a' != 'a' | | 'a' != 'a' |
  +-----------+ +-----------+
  | 1 | | 0 |
  +-----------+ +-----------+
  
  SELECT ('a' != 'a') = 0;
  +------------------+
  | ('a' != 'a') |
  +------------------+
  | 1 |
  +------------------+
  
  SELECT ('a' != 'a') IS FALSE;
  +------------------+
  | ('a' != 'a') IS FALSE |
  +------------------+
  | 1 |
  +------------------+
  ```

- plus additional operators, e.g.:
  - \( x \) BETWEEN \( y \) AND \( z \) ... true if \( y \leq x \leq z \)
  - \( x \) IN \( \{u,v,w,…\} \) ... true if \( x \) is a value in the list
  - etc.
Exercise ...

With a partner ...

- come up with at least three “interesting” queries over our tables
- e.g., involving joins and more complicated WHERE clauses (AND, OR, comparisons)
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