Lecture 7:
  • SQL Query Basics (cont)

Announcements:
  • HW-1 due
  • PS-2 due Tues, Sept 27
  • Q-2 on Tues, Sept 27

---

Example Tables

<table>
<thead>
<tr>
<th>Loan</th>
<th>acct_id</th>
<th>barcode</th>
<th>checkout_date</th>
<th>due_date</th>
<th>return_date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>101</td>
<td>4242</td>
<td>8/12</td>
<td>8/26</td>
<td>8/24</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>4243</td>
<td>8/12</td>
<td>8/19</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>4242</td>
<td>8/25</td>
<td>9/7</td>
<td>8/29</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>4243</td>
<td>7/10</td>
<td>7/17</td>
<td>7/18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Branch</th>
<th>branch_name</th>
<th>address</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Central</td>
<td>906 Main</td>
<td>444-5300</td>
</tr>
<tr>
<td></td>
<td>South Hill</td>
<td>3324 Perry</td>
<td>444-5301</td>
</tr>
<tr>
<td></td>
<td>Shadle</td>
<td>2111 Wellesley</td>
<td>444-5302</td>
</tr>
<tr>
<td></td>
<td>Hillyard</td>
<td>4110 Cook</td>
<td>444-5303</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Account</th>
<th>acct_id</th>
<th>acct_name</th>
<th>main_branch</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>101</td>
<td>Alice</td>
<td>Central</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>Bob</td>
<td>Central</td>
</tr>
<tr>
<td></td>
<td>103</td>
<td>Alice</td>
<td>Shadle</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>Chuck</td>
<td>South Hill</td>
</tr>
</tbody>
</table>
Warm Up Example

Q: What does this query return?

```sql
SELECT *
FROM account
WHERE main_branch = 'Central' AND
main_branch = 'Shadle'
```

This query is always empty (unsatisfiable) ... but the optimizer catches it!

```sql
mysql> EXPLAIN SELECT * FROM account WHERE main_branch = 'Downtown' AND
-> main_branch = 'South Hill';
+----+-------------+-----+------------------+
| id | select_type | ... | Extra |
+----+-------------+-----+------------------+
| 1 | SIMPLE | ... | Impossible WHERE |
```

FROM tables

Can “access” multiple tables in a single query ...

```sql
SELECT *
FROM Branch, Account
WHERE branch_name = main_branch
```

Can think of queries over multiple tables as nested for loops ...

1: for each row \( r_1 \) in Branch
2: for each row \( r_2 \) in Account
3: if \( r_1\.branch\_name == r_2\.main\_branch \)
4: add row \( r_1r_2 \) to result

Note: This is not how the DBMS executes the query!
FROM tables (cont)

<table>
<thead>
<tr>
<th>branch_name</th>
<th>address</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>906 Main</td>
<td>444-5300</td>
</tr>
<tr>
<td>South Hill</td>
<td>3324 Perry</td>
<td>444-5301</td>
</tr>
<tr>
<td>Shadle</td>
<td>2111 Wellesley</td>
<td>444-5302</td>
</tr>
<tr>
<td>Hillyard</td>
<td>4110 Cook</td>
<td>444-5303</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>acct_id</th>
<th>acct_name</th>
<th>main_branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Alice</td>
<td>Central</td>
</tr>
<tr>
<td>102</td>
<td>Bob</td>
<td>Central</td>
</tr>
<tr>
<td>103</td>
<td>Alice</td>
<td>Shadle</td>
</tr>
<tr>
<td>104</td>
<td>Chuck</td>
<td>South Hill</td>
</tr>
</tbody>
</table>

Check in: What does this query return?

```sql
SELECT *
FROM Branch, Account
WHERE branch_name = main_branch AND acct_name = 'Alice'
```

The table:

<table>
<thead>
<tr>
<th>branch_name</th>
<th>address</th>
<th>phone</th>
<th>acct_id</th>
<th>acct_name</th>
<th>main_branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>906 Main</td>
<td>444-5300</td>
<td>101</td>
<td>Alice</td>
<td>Central</td>
</tr>
<tr>
<td>Shadle</td>
<td>2111 Wellesley</td>
<td>444-5302</td>
<td>103</td>
<td>Alice</td>
<td>Shadle</td>
</tr>
</tbody>
</table>

 Attribute Disambiguation

Often need to state what tables attributes are from ...

```sql
SELECT acct_name, barcode, due_date
FROM Loan, Account
WHERE acct_id = acct_id
```

Approach 1: prefix attribute names with table names:

```sql
SELECT acct_name, barcode, due_date -- can also prefix w/ table name
FROM Loan, Account
WHERE Loan.acct_id = Account.acct_id
```

Approach 2: use “correlation” names:

```sql
SELECT a.acct_name, a.barcode, a.due_date -- can leave off if clear
FROM Loan l, Account a
WHERE l.acct_id = a.acct_id
```
Correlation Name Example

Check in: What is this query asking?

```
SELECT a1.acct_name, a1.acct_id, a2.acct_name, a2.acct_id
FROM account a1, account a2
WHERE a1.acct_name = a2.acct_name
```

In this example, correlation names are required
• that is, whenever the same table appears more than once in the FROM
• however, correlation names generally easier to read ...

JOIN Syntax

Q: What does this query do?

```
SELECT a.acct_id, a.main_branch, l.barcode
FROM Loan l, Account a
WHERE a.acct_id = 101
```

It returns the "cartesian product" of Loan and Account
• that is, every combination of rows
• in this case, we aren't "joining" the tables (connecting rows)

SQL provides special syntax to denote a cartesian product

```
SELECT a.acct_id, a.main_branch, l.barcode
FROM Loan l CROSS JOIN Account a
WHERE a.acct_id = 101
```
JOIN Syntax (cont)

For this query ...  

```
SELECT a.acct_id, a.main_branch, l.barcode
FROM Loan l, Account a
WHERE l.acct_id = a.acct_id
```

We are “joining” Loan and Account using the acct_id attribute
- note this is a common pattern when querying multiple tables

SQL provides a special JOIN-USING syntax for this case ...

```
SELECT a.acct_id, a.main_branch, l.barcode
FROM Loan l JOIN Account a USING (acct_id)
```

JOIN Syntax (cont)

For this query ...

```
SELECT a.acct_id, a.acct_name, b.phone
FROM Branch b, Account a
WHERE b.branch_name = a.main_branch
```

We are “joining” Branch and Account using different attributes
- note this is also a common pattern when querying multiple tables
- in this case, the “join condition” is b.branch_name = a.main_branch

SQL provides a special JOIN-ON syntax for this case ...

```
SELECT a.acct_id, a.acct_name, b.phone
FROM Branch b JOIN Account a ON (b.branch_name = a.main_branch)
```
JOIN Syntax (cont)

(1) Can join on multiple conditions
\[ R_1 \text{ JOIN } R_2 \text{ USING } (x, y) \quad \ldots \quad R_1(x, y), R_2(x, y) \]
\[ R_1 \text{ JOIN } R_2 \text{ ON } (x = u \text{ AND } y = v) \quad \ldots \quad R_1(x, y), R_2(u, v) \]

(2) An “equi-join” if join conditions only involve equality
\[ R_1 \text{ JOIN } R_2 \text{ ON } (x < u) \text{ is not an equi-join} \]

(3) Can chain together JOINs ...
Loan JOIN Account USING (acct_id)
\[ \text{JOIN Branch ON (main_branch = branch_name)} \]

(4) If no JOIN syntax used, often called a “comma join”
better to use JOIN syntax for readability

S. Bowers CPSC 321, Fall 2022 11

JOIN Syntax (cont)

A NATURAL JOIN is a special equi-join ...

\[ \text{SELECT *} \]
FROM Branch NATURAL JOIN Account;

- automatically joins on attributes with the same name
- only includes one of the attributes in the output table

If \( R_1(x, y, z) \) and \( R_2(x, u, z) \) then:
\[ \text{SELECT * FROM } R_1 \text{ NATURAL JOIN } R_2 \]
is same as (in MariaDB):
\[ \text{SELECT } x, z, y, u \]
FROM \( R_1 \text{ JOIN } R_2 \text{ ON } (x, z) \)

S. Bowers CPSC 321, Fall 2022 12