Information Systems
(Informationssysteme)

Jens Teubner, TU Dortmund
jens.teubner@cs.tu-dortmund.de

Summer 2018
Part III

A Very Brief Introduction to SQL
SQL—Structured Query Language

By far the most important query language today is SQL.

- Structured Query Language
- Originally meant to be used by end users 😊
- Today supported by virtually any database system

SQL operates on relational data:

<table>
<thead>
<tr>
<th>Ingredients</th>
<th>Alcohol</th>
<th>InStock</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>0.0</td>
<td>12</td>
<td>2.99</td>
</tr>
<tr>
<td>Campari</td>
<td>25.0</td>
<td>5</td>
<td>12.95</td>
</tr>
<tr>
<td>Mineral Water</td>
<td>0.0</td>
<td>10</td>
<td>1.49</td>
</tr>
<tr>
<td>Bacardi</td>
<td>37.5</td>
<td>3</td>
<td>16.98</td>
</tr>
</tbody>
</table>

Real databases may contain 100s or 1000s of tables, sometimes with billions of rows (also: tuples).
Our First SQL Query

The key construct of SQL is the `SELECT-FROM-WHERE` clause:

```
SELECT Name, Price
FROM Ingredients
WHERE Alcohol = 0
```

**SELECT** Choose a **set of columns** to be reported in the query result.

We’ll later call this **projection**, not selection.

**FROM** Choose a **table** where rows should be taken from.

**WHERE** Additional **conditions** that rows must satisfy in order to appear in the result (the **WHERE** clause is optional).

→ **This** is what we call a **selection**.
### Ingredients

<table>
<thead>
<tr>
<th>Name</th>
<th>Alcohol</th>
<th>InStock</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>0.0</td>
<td>12</td>
<td>2.99</td>
</tr>
<tr>
<td>Campari</td>
<td>25.0</td>
<td>5</td>
<td>12.95</td>
</tr>
<tr>
<td>Mineral Water</td>
<td>0.0</td>
<td>10</td>
<td>1.49</td>
</tr>
<tr>
<td>Bacardi</td>
<td>37.5</td>
<td>3</td>
<td>16.98</td>
</tr>
</tbody>
</table>

\[\text{SELECT Name, Price} \quad \text{FROM Ingredients} \quad \text{WHERE Alcohol} = 0\]

\[\begin{array}{|c|c|}
\hline
\text{Name} & \text{Price} \\
\hline
\text{Mineral Water} & 1.49 \\
\text{Orange Juice} & 2.99 \\
\hline
\end{array}\]
Cocktail ingredients are sold by various suppliers (for a certain price), which could be represented as

<table>
<thead>
<tr>
<th>Ingredient</th>
<th>Supplier</th>
<th>DelTim(^3)</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>A&amp;P Supermarket</td>
<td>1</td>
<td>2.49</td>
</tr>
<tr>
<td>Orange Juice</td>
<td>Shop Rite</td>
<td>3</td>
<td>2.79</td>
</tr>
<tr>
<td>Campari</td>
<td>Joe’s Liquor Store</td>
<td>2</td>
<td>14.95</td>
</tr>
<tr>
<td>Bacardi</td>
<td>Liquor’s &amp; More</td>
<td>5</td>
<td>13.99</td>
</tr>
<tr>
<td>Mineral Water</td>
<td>Shop Rite</td>
<td>3</td>
<td>1.89</td>
</tr>
<tr>
<td>Bacardi</td>
<td>Joe’s Liquor Store</td>
<td>2</td>
<td>14.99</td>
</tr>
</tbody>
</table>

\(^3\)Delivery time in days.
When multiple tables are referenced in the `FROM` clause, this is interpreted as the **Cartesian product** of the referenced tables:

```
SELECT *
FROM Ingredients, SoldBy
```

---

4 Use * in the SELECT clause when you simply want to choose all columns.
Queries over Multiple Tables

In practice, you rarely want to see this Cartesian product in the final result.

→ Use a **WHERE** clause to select only semantically related data.

```sql
SELECT Name, InStock, Supplier
FROM Ingredients, SoldBy
WHERE Name = Ingredient
```

<table>
<thead>
<tr>
<th>Name</th>
<th>InStock</th>
<th>Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>A&amp;P Supermarket</td>
</tr>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>Shop Rite</td>
</tr>
<tr>
<td>Campari</td>
<td>5</td>
<td>Joe’s Liquor Store</td>
</tr>
<tr>
<td>Mineral Water</td>
<td>10</td>
<td>Shop Rite</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Liquors &amp; More</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Joe’s Liquor Store</td>
</tr>
</tbody>
</table>
Resolve ambiguities by prepending column names with their table name:

```
SELECT Name, InStock, Supplier, SoldBy.Price
FROM Ingredients, SoldBy
WHERE Name = Ingredient
    AND SoldBy.Price < Ingredients.Price
```

<table>
<thead>
<tr>
<th>Name</th>
<th>InStock</th>
<th>Supplier</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>A&amp;P Supermarket</td>
<td>2.49</td>
</tr>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>Shop Rite</td>
<td>2.79</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Liquors &amp; More</td>
<td>13.99</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Joe’s Liquor Store</td>
<td>14.99</td>
</tr>
</tbody>
</table>
... or introduce **tuple variables** for easier reference:

```sql
SELECT Name, InStock, Supplier, s.Price
FROM Ingredients AS i, SoldBy AS s
WHERE Name = Ingredient
AND s.Price < i.Price
```

<table>
<thead>
<tr>
<th>Name</th>
<th>InStock</th>
<th>Supplier</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>A&amp;P Supermarket</td>
<td>2.49</td>
</tr>
<tr>
<td>Orange Juice</td>
<td>12</td>
<td>Shop Rite</td>
<td>2.79</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Liquors &amp; More</td>
<td>13.99</td>
</tr>
<tr>
<td>Bacardi</td>
<td>3</td>
<td>Joe’s Liquor Store</td>
<td>14.99</td>
</tr>
</tbody>
</table>

(The keyword `AS` is optional; ‘SoldBy s’ would mean just the same.)
Semantics of SQL SELECT-FROM-WHERE Expressions

Conceptually, the query

\[
\text{SELECT AttList} \\
\text{FROM TableName}_1, \text{TableName}_2, \ldots \\
\text{WHERE Condition}
\]

does the following:

\[
\begin{align*}
\text{TableName}_1 & \quad \rightarrow \quad \times \\
\text{TableName}_2 & \quad \rightarrow \\
\vdots & \quad \rightarrow
\end{align*}
\]

\[
\begin{align*}
\text{FROM} & \quad \rightarrow \\
\text{WHERE} & \quad \rightarrow \\
\text{SELECT} & \quad \rightarrow
\end{align*}
\]

(But most likely, the database system will choose a better strategy to actually execute the query.)
Concluding Remarks

- SQL is **case insensitive**; use ’ as a **string delimiter**.
- It is okay to reference the **same table multiple times** in a FROM clause (→ “self-join”). Use **tuple variables** then to tell things apart.

**Never, never ever**, write queries where the correctness depends on the current table contents.

*E.g.*, the correct answer to “give me names and prices of all non-alcoholic ingredients” is **not**

```sql
SELECT Name, Price
FROM Ingredients
WHERE Name = 'Orange Juice' OR Name = 'Mineral Water'
```