SQL Queries

- General form:

  ```sql
  SELECT attributes you want
  FROM relations
  WHERE conditions about tuples from relations;
  ```

- Read and write in this order: from-where-select

Running Example

- `Beers(name, manf)`
- `Bars(name, addr, license)`
- `Drinkers(name, addr, phone)`
- `Likes(drinker, beer)`
- `Sells(bar, beer, price)`
- `Frequents(drinker, bar)`

Example Query

- What beers are made by Anheuser-Busch?

  ```sql
  SELECT name
  FROM Beers
  WHERE manf = 'Anheuser-Busch';
  ```

- Result:

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
</tr>
<tr>
<td>Bud</td>
</tr>
<tr>
<td>Michelob</td>
</tr>
</tbody>
</table>

Formal Semantics of Single-Relation SQL Query

1. Start with the relation in the FROM clause.
2. Apply (bag) $\sigma$, using condition in WHERE clause.
3. Apply (extended, bag) $\pi$ using attributes in SELECT clause.
Equivalent Operational Semantics

- Imagine a tuple variable ranging over all tuples of the relation. For each tuple:
  - Check if it satisfies the WHERE clause.
  - Print the values of terms in SELECT, if so.

Star as List of All Attributes

- Beers(name, manf)
  SELECT *
  FROM Beers
  WHERE manf = 'Anheuser-Busch';
  • Result:

<table>
<thead>
<tr>
<th>name</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
<td>Anheuser-Busch</td>
</tr>
<tr>
<td>Bud</td>
<td>Anheuser-Busch</td>
</tr>
<tr>
<td>Michelob</td>
<td>Anheuser-Busch</td>
</tr>
</tbody>
</table>

Renaming Columns

- Beers(name, manf)
  SELECT name AS beer
  FROM Beers
  WHERE manf = 'Anheuser-Busch';
  • Result:

<table>
<thead>
<tr>
<th>beer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudLite</td>
</tr>
<tr>
<td>Bud</td>
</tr>
<tr>
<td>Michelob</td>
</tr>
</tbody>
</table>

Expressions as Values in Columns

- Sells(bar, beer, price)
  SELECT bar, beer, price*0.74 AS priceInEuros
  FROM Sells;
  • Note: no WHERE clause is OK.

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>priceInEuros</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoon</td>
<td>Amstel</td>
<td>2.96</td>
</tr>
<tr>
<td>Spoon</td>
<td>Guinness</td>
<td>5.18</td>
</tr>
<tr>
<td>Whiskey</td>
<td>Guinness</td>
<td>5.18</td>
</tr>
<tr>
<td>Whiskey</td>
<td>Bud</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Constant Values

- If you want an answer with a particular string in each row, use that constant as an expression.
- Likes(drinker, beer)
  SELECT drinker, 'connoisseur' AS status
  FROM Likes
  WHERE beer = 'Guinness';
  • Result:

<table>
<thead>
<tr>
<th>drinker</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul</td>
<td>connoisseur</td>
</tr>
<tr>
<td>Ryan</td>
<td>connoisseur</td>
</tr>
<tr>
<td>Paul</td>
<td>connoisseur</td>
</tr>
</tbody>
</table>

Example

- Find the price Spoon charges for Bud.
  Sells(bar, beer, price)
  SELECT price
  FROM Sells
  WHERE bar = 'Spoon' AND beer = 'Bud';
  • Conditions in WHERE clause can use logical operators AND, OR, NOT and parentheses in the usual way.
  • SQL is case insensitive. Keywords like SELECT or AND can be written upper/lower case as you like.
  • Only inside quoted strings does case matter.
Example 2

• Find the names of all bars that sell for less than $4 at least one beer that's not Bud.

String Patterns

• % stands for any string.
• _ stands for any one character.
• “Attribute LIKE pattern” is a condition that is true if the string value of the attribute matches the pattern.
  – Also NOT LIKE for negation.

Example

• Find drinkers whose phone has exchange 555.
• Drinkers(name, addr, phone)
  SELECT name
  FROM Drinkers
  WHERE phone LIKE '%555-_ _ _ _';
• Note patterns must be quoted, like strings

Nulls

• In place of a value in a tuple's component.
• Interpretation is not exactly missing value.
• There could be many reasons why no value is present, e.g., value inappropriate.

Comparing Nulls to Values

• 3rd truth value UNKNOWN.
• A query only produces tuples if the WHERE-condition evaluates to TRUE (UNKNOWN is not sufficient).

Example

SELECT bar
FROM Sells
WHERE price < 2.00 OR price >= 2.00;

UNKNOWN    UNKNOWN    UNKNOWN

• The result is empty, even though the WHERE condition is a tautology.
3-Valued Logic

• Think of true = 1; false = 0, and unknown = 1/2.
• Then:
  – AND = min.
  – OR = max.
  – NOT(x) = 1 – x.

Some Key Laws Do Not Hold

• Example: Law of the excluded middle, i.e.,
  p OR NOT p = TRUE
• For 3-valued logic: if p = unknown, then
  left side = max(1/2, (1-1/2)) = 1/2 ≠ 1.
• Like bag algebra, there is no way known to
  make 3-valued logic conform to all the
  laws we expect for sets/2-valued logic,
  respectively.

Example Query

• Find all bars that do not sell Bud for more
  than $5.
  – Two interpretations?