Connecting to Postgres

Clients

- SQL shell (psql)
- pgAdmin
- PHP program
- Java program

(local or remote)

Server

Postgres

(db-edlab)
Getting started with Postgres

• Resources available
  – course website, from “Assignments” page
    • http://www.postgresql.org/docs/8.1/static/index.html
    • especially, Part I: Tutorial
SQL Overview

• Query capabilities
  – SELECT-FROM-WHERE blocks,
  – Basic features, ordering, duplicates
  – Set operations (union, intersect, except)
  – Aggregation & Grouping
  – Nested queries (correlation)
  – Null values
Example database

Sailors (sid, sname, rating, age)
Boats (bid, bname, color)
Reserves (sid, bid, day)

Key for each table indicated by underlined attributes.
<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>brutus</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>85</td>
<td>art</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>95</td>
<td>bob</td>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>96</td>
<td>frodo</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45</td>
</tr>
<tr>
<td>64</td>
<td>horatio</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>32</td>
<td>andy</td>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>74</td>
<td>horatio</td>
<td>9</td>
<td>35</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>71</td>
<td>zorba</td>
<td>10</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>10/10</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>10/8</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>10/7</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>11/10</td>
</tr>
<tr>
<td>31</td>
<td>103</td>
<td>11/6</td>
</tr>
<tr>
<td>31</td>
<td>104</td>
<td>11/12</td>
</tr>
<tr>
<td>64</td>
<td>101</td>
<td>9/5</td>
</tr>
<tr>
<td>64</td>
<td>102</td>
<td>9/8</td>
</tr>
<tr>
<td>74</td>
<td>103</td>
<td>9/8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Interlake</td>
<td>blue</td>
</tr>
<tr>
<td>102</td>
<td>Interlake</td>
<td>red</td>
</tr>
<tr>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>104</td>
<td>Marine</td>
<td>red</td>
</tr>
</tbody>
</table>
SQL Query

Basic form: (plus many many extensions)

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification conditions
```

For example:

```
SELECT sid, sname, rating, age
FROM Sailors
WHERE age > 21
```
Basic SQL Query

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
```

- **target-list** A list of attributes of relations in `relation-list`.
- **relation-list** A list of relation names (possibly with a range-variable after each name).
- **qualification** Comparisons (Attr \( op \) const or Attr1 \( op \) Attr2, where \( op \) is one of \(<, >, =, \leq, \geq, \neq\) combined using AND, OR and NOT.
- **DISTINCT** is an optional keyword indicating that the answer should not contain duplicates. Default is that duplicates are *not* eliminated!
Simple SQL Query

SELECT * FROM Sailors WHERE age > 21

Sailors

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>71</td>
<td>zorba</td>
<td>10</td>
<td>16</td>
</tr>
</tbody>
</table>

Conditions in the WHERE clause are like selection: $\sigma_{age > 21}$
Selection conditions

What goes in the **WHERE** clause:

- $x = y$, $x < y$, $x \leq y$, $x \neq y$, etc
  - For number, they have the usual meanings
  - For CHAR and VARCHAR: lexicographic ordering
  - For dates and times, what you expect...

- Also, pattern matching on strings: $s \text{ LIKE } p$
The **LIKE** operator

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = any sequence of characters
  - _ = any single character

Find all students whose name begins and ends with ‘b’:

```
SELECT * FROM Sailors WHERE sname LIKE 'b%b'
```

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>brutus</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>85</td>
<td>art</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>95</td>
<td>bob</td>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>96</td>
<td>frodo</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45</td>
</tr>
<tr>
<td>64</td>
<td>horatio</td>
<td>7</td>
<td>35</td>
</tr>
</tbody>
</table>
Simple SQL Query

SELECT sname, age
FROM Sailors
WHERE age > 21

Conditions in the SELECT clause are like projection: $\Pi_{\text{sname, age}}$
Note confusing terminology 😞

```
SELECT  sname, age
FROM    Sailors
WHERE   age > 21
```

Conditions in the WHERE clause are like selection: $\sigma_{\text{age}<21}$

Conditions in the SELECT clause are like projection: $\Pi_{\text{sname, age}}$
Eliminating Duplicates

SELECT DISTINCT sname
FROM Sailors

Compare to:

SELECT sname
FROM Sailors

sname

brutus
art
bob
frodo
dustin
horatio
lubber
andy

sname

brutus
art
bob
frodo
dustin
horatio
lubber
andy

Default behavior does not eliminate duplicates.
Ordering the Results

```
SELECT  sname, rating, age
FROM    Sailors
WHERE   age > 18
ORDER BY rating, sname
```

Ordering is ascending, unless you specify the DESC keyword.

Ties are broken by the second attribute on the ORDER BY list, etc.
Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of a conceptual evaluation strategy:
  - Compute the cross-product of `relation-list`.
  - Discard resulting tuples if they fail `qualifications`.
  - Delete attributes that are not in `target-list`.
  - If `DISTINCT` is specified, eliminate duplicate rows.

- Probably the least efficient way to compute a query -- optimizer will find more efficient plan.
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
Example

```
SELECT sname
FROM Sailors S, Reserves R, Boats B
    AND B.color = 'red'
```

What does this query compute?

*Find the names of sailors who have reserved a red boat*
Please write in SQL

*Find the colors of boats reserved by ‘Lubber’*

```sql
SELECT B.color 
FROM Sailors S, Reserves R, Boats B 
  AND S.sname = ‘Lubber’
```
### Renaming Columns

#### Sailors

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>71</td>
<td>zorba</td>
<td>10</td>
<td>16</td>
</tr>
</tbody>
</table>

#### SQL Query

```sql
SELECT sname AS name, age AS x
FROM Sailors
WHERE age > 21
```
Disambiguating Attributes

- Sometimes two relations have the same attr:
  Person(pname, address, worksfor)
  Company(cname, address)

```
SELECT DISTINCT pname, address
FROM Person, Company
WHERE worksfor = cname
```

```
SELECT DISTINCT Person.pname, Company.address
FROM Person, Company
WHERE Person.worksfor = Company.cname
```
Range Variables in SQL

Purchase (buyer, seller, store, product)

Find all stores that sold at least one product that was sold at ‘BestBuy’:

```
SELECT DISTINCT x.store
FROM Purchase AS x, Purchase AS y
WHERE x.product = y.product AND y.store = 'BestBuy'
```
Please write in SQL

Self-join on Flights:
The departure and arrival cities of trips consisting of two direct flights.

```sql
SELECT F1.depart, F2.arrive
FROM Flights as F1, Flights as F2
WHERE F1.arrive = F2.depart
```

<table>
<thead>
<tr>
<th>depart</th>
<th>arrive</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC</td>
<td>Reno</td>
</tr>
<tr>
<td>NYC</td>
<td>Oakland</td>
</tr>
<tr>
<td>Boston</td>
<td>Tampa</td>
</tr>
<tr>
<td>Oakland</td>
<td>Boston</td>
</tr>
<tr>
<td>Tampa</td>
<td>NYC</td>
</tr>
</tbody>
</table>