CSCI-GA.2433-001

Database Systems

Lecture 5: SQL I

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com
SQL?

- Structured query language
- Originally developed by IBM
- It is now de facto standard.
- Continues to evolve in response to changing needs in DB area
- Latest standard so far SQL:2011
SQL?

- Data Manipulation Language
- Data Definition Language
- Triggers and Advanced Integrity Constraints
- Embedded and Dynamic SQL
- Client-Server Execution and Remote Access
- Transaction Management
- Security
- Advanced Features

• Queries
  • add/delete/modify rows

Lecture 3
Our Running Example

Sailors($sid$: integer, $sname$: string, $rating$: integer, $age$: real)
Boats($bid$: integer, $bname$: string, $color$: string)
Reserves($sid$: integer, $bid$: integer, $day$: date)

<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>

<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>45.0</td>
</tr>
<tr>
<td>29</td>
<td>Brutus</td>
<td>1</td>
<td>33.0</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>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>9</td>
<td>35.0</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>
</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>10/10/98</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>10/10/98</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>10/8/98</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>10/7/98</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>11/10/98</td>
</tr>
<tr>
<td>31</td>
<td>103</td>
<td>11/6/98</td>
</tr>
<tr>
<td>31</td>
<td>104</td>
<td>11/12/98</td>
</tr>
<tr>
<td>64</td>
<td>101</td>
<td>9/5/98</td>
</tr>
<tr>
<td>64</td>
<td>102</td>
<td>9/8/98</td>
</tr>
<tr>
<td>74</td>
<td>103</td>
<td>9/8/98</td>
</tr>
</tbody>
</table>
A Basic SQL Query

**SELECT**

- Mandatory
- Specifies columns to be retained in the result
- A list of attributes of relations in *relation-list*
- *[DISTINCT]* is optional and means no duplication in result

**FROM**

- Mandatory
- Specifies a cross-product of tables

**WHERE**

- Optional
- Specifies the selection conditions on the tables mentioned in FROM

---

*Mandatory*

- Specifies columns to be retained in the result
- A list of attributes of relations in *relation-list*
- *[DISTINCT]* is optional and means no duplication in result

*Mandatory*

- Specifies a cross-product of tables

*Optional*

- Specifies the selection conditions on the tables mentioned in FROM
Q: Find the names and ages of all sailors

SELECT DISTINCT S.sname, S.age
FROM    Sailors S

If we omit the keyword DISTINCT
Q: Find all sailors with rating > 7

SELECT S.sid, S.sname, S.rating, S.age
FROM Sailors AS S
WHERE S.rating > 7

A shorthand for choosing all attributes is SELECT *

Optional, for introducing range variable

Do you see that:
• SELECT is actually doing projection of relational algebra
• FROM is the selection of the relational algebra
More About SQL “WHERE”

• WHERE is followed by qualification
• Qualification is:
  – Boolean combination (using AND, OR, and NOT) of conditions.
  – Condition is of form:
    • expression op expression
    • expression can be:
      – column name
      – constant
      – arithmetic or string expression
    • op is a comparison operator: <, >, <=, >=, >
Logically, How Is An SQL Query Executed?

1. Compute the cross-product of the tables in the from-list.
2. Delete rows in the product that fail the qualification condition.
3. Delete all columns that do not appear in the select-list.
4. If DISTINCT is specified, eliminate duplicate rows.

Is DBMS doing something more efficient?
Logically, How Is An SQL Query Executed?

Q: Find the names of sailors who have reserved boat number 103

```
SELECT S.sname 
FROM   Sailors S, Reserves R 
WHERE  S.sid=R.sid AND R.bid=103
```

**STEP 1: Construct the cross product**

<table>
<thead>
<tr>
<th>(sid)</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
<th>(sid)</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>
Logically, How Is An SQL Query Executed?

Q: Find the names of sailors who have reserved boat number 103

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

STEP 2: Apply the qualification
Logically, How Is An SQL Query Executed?

Q: Find the names of sailors who have reserved boat number 103

```sql
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=103
```

STEP 3: Eliminate unwanted columns
Simple SQL Examples

The use of range variables is optional and used in two cases:
- To remove ambiguity
- For readability
- FROM relation contains more than one occurrence of a relation

Note:
Table name cannot be used as range variable, if a range variable is introduce at FROM
Simple SQL Examples

Sailors(sid: integer, sname: string, rating: integer, age: real)
Boats(bid: integer, bname: string, color: string)
Reserves(sid: integer, bid: integer, day: date)

Q: Find the colors of boats reserved by Lubber

SELECT  B.color
FROM     Sailors S, Reserves R, Boats B
More Sophisticated select-list

• Can be more sophisticated than just column names.
• Each item in the select list can be in the form:
  • `expression AS column_name`
    – Expression is any arithmetic or string expression over column names and constants
    – column_name is a new name for this column in the output
Example:

Q: Compute increments for the ratings of persons who have sailed two different boats on the same day.

```sql
SELECT S.sname, S.rating+1 AS rating
FROM Sailors S, Reserves R1, Reserves R2
WHERE S.sid = R1.sid AND S.sid = R2.sid
    AND R1.day = R2.day AND R1.bid <> R2.bid
```
More Sophisticated qualification

Can be as general as expression1 = expression2

Example:

SELECT S1.sname AS name1, S2.sname AS name2
FROM Sailors S1, Sailors S2
WHERE 2*S1.rating = S2.rating-1
String Operations

- Give more strength to the qualification
- Comparison operators can be used for string comparisons
  - Alphabetical order is assumed by default
  - But SQL has a concept of collation for a character set to allow the user to specify which characters are less than which others.
- Pattern matching through **LIKE** operator
- Regular expression with **SIMILAR** operator
- Wild-card symbols:
  - `%` zero or more arbitrary characters
  - `_` exactly one arbitrary character
  - example: `_AB%`
  - blanks can be significant for the like operator
String Operations

Q: Find the ages of sailors whose name begins and ends with B and has at least 3 characters

SELECT S.age
FROM Sailors S
WHERE S.name LIKE 'B_%B'
Set Operations

• More capabilities to the qualifications
• UNION
• INTERSECT
• EXCEPT
• op ANY
• op ALL
• [NOT] EXISTS
• [NOT] IN
Set Operations

Sailors(sid: integer, sname: string, rating: integer, age: real)
Boats(bid: integer, bname: string, color: string)
Reserves(sid: integer, bid: integer, day: date)

Q: Find the names of sailors who have reserved a red or a green boat

SELECT S.sname
FROM Sailors S, Reserves R, Boats B
AND (B.color = 'red' OR B.color = 'green')
Set Operations

Sailors (sid: integer, sname: string, rating: integer, age: real)
Boats (bid: integer, bname: string, color: string)
Reserves (sid: integer, bid: integer, day: date)

Q: Find the names of sailors who have reserved a red and a green boat

```
SELECT  S.sname
FROM     Sailors S, Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE    S.sid = R1.sid AND R1.bid = B1.bid
         AND S.sid = R2.sid AND R2.bid = B2.bid
         AND B1.color='red' AND B2.color = 'green'
```

This one is correct BUT difficult to understand and quite inefficient to execute too!
Set Operations

Q: Find the names of sailors who have reserved a red and a green boat

SELECT S.sname
FROM   Sailors S, Reserves R, Boats B
WHERE  S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S2.sname
FROM   Sailors S2, Boats B2, Reserves R2
WHERE  S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green'
Set Operations

Q: Find the sids of all sailors who have reserved red boats but not green boats

\[
\begin{align*}
\text{SELECT} & \quad S.\text{sid} \\
\text{FROM} & \quad \text{Sailors } S, \text{ Reserves } R, \text{ Boats } B \\
\text{WHERE} & \quad S.\text{sid} = R.\text{sid} \land R.\text{bid} = B.\text{bid} \land B.\text{color} = \text{‗red‘} \\
\text{EXCEPT} & \quad \text{SELECT} \quad S2.\text{sid} \\
\text{FROM} & \quad \text{Sailors } S2, \text{ Reserves } R2, \text{ Boats } B2 \\
\text{WHERE} & \quad S2.\text{sid} = R2.\text{sid} \land R2.\text{bid} = B2.\text{bid} \land B2.\text{color} = \text{‗green‘}
\end{align*}
\]

Result: 22, 64, and 31

Result: 64
Comments

• UNION, INTERSECT, and EXCEPT can be used with any two union-compatible

• Duplicates are removed.
  – If you want to keep them, use UNION ALL, INTERSECT ALL, EXCEPT ALL

```
SELECT S.sid
FROM Sailors S
WHERE S.rating = 10
UNION
SELECT R.sid
FROM Reserves R
WHERE R.bid = 104
```
Question: Suppose there are two sets where a row is duplicated m times in the first and n times in the second. How many duplicates we will have after using: UNION ALL INTERSECT ALL EXCEPT ALL
Nested Queries

- A query that has another query (called subquery) embedded within it.
- The process is recursive. That is, the subquery may be nested query itself.

```sql
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
    FROM Reserves R
    WHERE R.bid = 103 )
```
Q: Find the names of sailors who have reserved a red boat

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
    FROM Reserves R
    WHERE R.bid IN (SELECT B.bid
        FROM Boats B
        WHERE B.color = 'red'))
```
Nested Queries

Q: Find the names of sailors who have NOT reserved a red boat

```sql
SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN (SELECT R.sid
FROM Reserves R
WHERE R.bid IN (SELECT B.bid
FROM Boats B
WHERE B.color = 'red')
)
Correlated Nested Queries

• What if the inner subquery depends on the row currently being examined in the outer query?

Q: Find the names of sailors who reserved boat number 103

Sailors\((sid: \text{integer}, \ sname: \text{string}, \ rating: \text{integer}, \ age: \text{real})\)

Boats\((bid: \text{integer}, \ bname: \text{string}, \ color: \text{string})\)

Reserves\((sid: \text{integer}, \ bid: \text{integer}, \ day: \text{date})\)

SELECT S.sname
FROM Sailors S
WHERE EXISTS ( SELECT *
FROM Reserves R
WHERE R.bid = 103
AND R.sid = S.sid )
Set-Comparison Operators

- \text{op\ ANY\ and\ op\ ALL}
- \text{op is one of \{<, \leq, =, \neq, \geq, >\}}

Q: Find sailors whose rating is better than some sailor called Horatio

If there is no sailor with that name, ANY is defined to return FALSE
Set-Comparison Operators

• op ANY and op ALL
• op is one of {<, <=, =, <>}, >=, >}

Q: Find sailors whose rating is better than every sailor called Horatio

```
Sailors(sid: integer, sname: string, rating: integer, age: real)
Boats(bid: integer, bname: string, color: string)
Reserves(sid: integer, bid: integer, day: date)

SELECT S.sid
FROM   Sailors S
WHERE  S.rating > ALL

SELECT S2.rating
FROM   Sailors S2
WHERE  S2.sname = 'Horatio'
```

If there is no sailor with that name, ALL is defined to return TRUE
Set-Comparison Operators

• **op ANY** and **op ALL**
• **op** is one of \{<, <=, =, <> , >= , >\}

---

**Sailors** *(sid: integer, sname: string, rating: integer, age: real)*

**Boats** *(bid: integer, bname: string, color: string)*

**Reserves** *(sid: integer, bid: integer, day: date)*

Q: Find the sailors with the highest rating

```sql
SELECT S.sid
FROM Sailors S
WHERE S.rating >= ALL ( SELECT S2.rating
                         FROM Sailors S2 )
```
Q: Find the names of sailors who have reserved both a red and a green boat

```
SELECT  S.sname
FROM     Sailors S, Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE    S.sid = R1.sid AND R1.bid = B1.bid
AND      S.sid = R2.sid AND R2.bid = B2.bid
AND      B1.color='red' AND B2.color = 'green'
```

```
SELECT  S2.sname
FROM     Sailors S2, Boats B2, Reserves R2
WHERE    S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green'
```

```
SELECT  S.sname
FROM     Sailors S, Reserves R, Boats B
WHERE    S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT  S2.sname
FROM     Sailors S2, Boats B2, Reserves R2
WHERE    S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green'
```

```
SELECT  S.sname
FROM     Sailors S, Reserves R, Boats B
WHERE    S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
AND      S.sid IN ( SELECT  S2.sid
                        FROM     Sailors S2, Boats B2, Reserves R2
                        WHERE    S2.sid = R2.sid AND R2.bid = B2.bid
                        AND      B2.color = 'green' )
```
Sailors($sid$: integer, $sname$: string, $rating$: integer, $age$: real)  
Boats($bid$: integer, $bname$: string, $color$: string) 
Reserves($sid$: integer, $bid$: integer, $day$: date)

Q: Find the names of sailors who have reserved all boats

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS ((
    SELECT B.bid
    FROM Boats B
    EXCEPT
    (SELECT R.bid
    FROM Reserves R
    WHERE R.sid = S.sid )))
```

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (
    SELECT B.bid
    FROM Boats B
    WHERE NOT EXISTS ( 
        SELECT R.bid
        FROM Reserves R
        WHERE R.bid = B.bid
        AND R.sid = S.sid ))
```
Conclusions

• SQL is by far the most widely used query language
• There are many standards, the latest is SQL:2011
• The best way to train yourself on SQL is to do a lot of queries by hand. Do not just read and understand queries