Tutorial 5: SQL

By Chaofa Gao

Tables used in this note:

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>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>29</td>
<td>Brutus</td>
<td>1</td>
<td>33</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</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>9</td>
<td>40</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>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>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>1998-10-10</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>1998-10-10</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>1998-10-8</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>1998-10-7</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>1998-11-10</td>
</tr>
<tr>
<td>31</td>
<td>103</td>
<td>1998-11-6</td>
</tr>
<tr>
<td>31</td>
<td>104</td>
<td>1998-11-12</td>
</tr>
<tr>
<td>64</td>
<td>101</td>
<td>1998-9-5</td>
</tr>
<tr>
<td>64</td>
<td>102</td>
<td>1998-9-8</td>
</tr>
<tr>
<td>74</td>
<td>103</td>
<td>1998-9-8</td>
</tr>
</tbody>
</table>

Figure 1: Instances of Sailors, Boats and Reserves

1. Create the Tables:

CREATE TABLE sailors (sid integer not null,
                      sname varchar(32),
                      rating integer,
                      age real,
                      CONSTRAINT PK_sailors PRIMARY KEY(sid) );

CREATE TABLE reserves (sid integer not null,
                        bid integer not null,
                        day datetime not null,
                        CONSTRAINT PK_reserves PRIMARY KEY(sid, bid, day),
                        FOREIGN KEY(sid) REFERENCES sailors(sid),
                        FOREIGN KEY(bid) REFERENCES boats(bid) );
2. Insert Data

```sql
INSERT INTO sailors
    ( sid, sname, rating, age )
VALUES ( 22, 'Dustin', 7, 45.0 )

INSERT INTO reserves
    ( sid, bid, day )
VALUES ( 22, 101, '1998-10-10' )
```

Note the date can have one of the following formats:
- yyyy-mm-dd
- mm-dd-yyyy
- mm/dd/yyyy

In addition, DB2 allows to parse the date attribute using its month(), year() and day() functions.

```sql
e.g. select * from reserves where year(day) = 1998 and month(day) = 10
```

3. Simple SQL Query

The basic form of an SQL query:

```sql
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
```

**Ex1: Using DISTINCT**

**sname | age**
---|---
Dustin | 45
Brutus | 33
Lubber | 55.5
Andy | 25.5
Rusty | 35
Horatio | 35
Zorba | 16

```sql
SELECT sname, age
FROM sailors
```

```sql
or

SELECT S.sname, S.age
FROM sailors S
```

**sname | age**
---|---
Andy | 25.5
Art | 25.5
Bob | 63.5
Brutus | 33
Dustin | 45
Horatio | 35
Lubber | 55.5
Rusty | 35
Zorba | 16

**Ex2. Find all information of sailors who have reserved boat number 101.**

```sql
SELECT S.*
FROM Sailors S, Reserves R
```
WHERE S.sid = R.sid AND R.bid = 103
Or without using the range variables, S and R

SELECT Sailors.*
FROM Sailors, Reserves
WHERE Sailors.sid = Reserves.sid AND Reserves.bid = 103

* can be used if you want to retrieve all columns.

**Ex3. Find the names of sailors who have reserved a red boat, and list in the order of age.**

SELECT S.sname, S.age
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
ORDER BY S.age

ORDER BY S.age [ASC]  (default)
ORDER BY S.age DESC

**Ex4. Find the names of sailors who have reserved at least one boat.**

SELECT sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid

The join of Sailors and Reserves ensure that for each select sname, the sailor has made some reservation.

**Ex5. Find the ids and names of sailors who have reserved two different boats on the same day.**

SELECT DISTINCT S.sid, S.sname
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

**Ex6. Using Expressions and Strings in the SELECT Command.**

SELECT sname, age, rating + 1 as sth
FROM Sailors
WHERE 2* rating -  1 < 10 AND sname like 'B_%b'

SQL provides for pattern matching through LIKE operator, along with the use of symbols: % (which stands for zero or more arbitrary characters) and _ (which stands for exactly one, arbitrary, characters)
4. Union, Intersect and Except

Note that Union, Intersect and Except can be used on only two tables that are union-compatible, that is, have the same number of columns and the columns, taken in order, have the same types.

Ex7. Find the ids of sailors who have reserved a red boat or a green boat.

SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid = B.bid AND B.color = ‘red’
UNION
SELECT R2.sid
FROM Boats B2, Reserves R2
WHERE R2.bid = B2.bid AND B2.color = ‘green’

The answer contains: SID----------22 31 64 74
The default for UNION queries is that duplicates are eliminated. To retain duplicates, use UNION ALL.
Replace UNION with UNION ALL. The answer contains: 22 31 74 22 31 64 22 31
Replace UNION with INTERSECT. The answer contains: 22 31.
Replace UNION with EXCEPT. The answer contains just the id 64.

6. Nested Query

IN and NOT IN
EXISTS and NOT EXISTS
UNIQUE and NOT UNIQUE
op ANY
op ALL

EX8: Find the names of sailors who have reserved boat 103.

SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECT R.sid
FROM Reserves R
WHERE R.bid = 103 )

The inner subquery has been completely independent of the outer query.
**Correlated Nested Queries**

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS ( SELECT *
    FROM Reserves R
    WHERE R.bid = 103
    AND R.sid = S.sid )
```

The inner query depends on the row that is currently being examined in the outer query.

**EX9: Find the name and the age of the youngest sailor.**

```
SELECT S.sname, S.age
FROM Sailors S
WHERE S.age <= ALL ( SELECT age
    FROM Sailors )
```

**EX10: Find the names and ratings of sailor whose rating is better than some sailor called Horatio.**

```
SELECT S.sname, S.rating
FROM Sailors S
WHERE S.rating > ANY ( SELECT S2.rating
    FROM Sailors S2
    WHERE S2.sname = 'Horatio' )
```

Note that **IN** and **NOT IN** are equivalent to **= ANY** and **<> ALL**, respectively.

**EX11: 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 ) )
```

An alternative solution:

```
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.sid = S.sid ) )
```
WHERE R.bid = B.bid
AND R.sid = S.sid )

7. Aggregation Operators

COUNT ([DISTINCT] A): The number of (unique) values in the A column.
SUM ([DISTINCT] A): The sum of all (unique) values in the A column.
AVG ([DISTINCT] A): The average of all (unique) values in the A column.
MAX (A): The maximum value in the A column.
MIN (A): The minimum value in the A column.

**EX12: Count the number of different sailor names.**

SELECT COUNT( DISTINCT S.sname )
FROM Sailors S

**EX13: Calculate the average age of all sailors.**

SELECT AVG(s.age)
FROM Sailors S

**EX14: Find the name and the age of the youngest sailor.**

SELECT S.sname, S.age
FROM Sailors S
WHERE S.age = (SELECT MIN(S2.age)
FROM Sailors S2 )

**EX15: Find the average age of sailors for each rating level.**

<table>
<thead>
<tr>
<th>Rating</th>
<th>avg_age</th>
<th>SELECT S.rating, AVG(S.age) AS avg_age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
<td>FROM Sailors S</td>
</tr>
<tr>
<td>3</td>
<td>44.5</td>
<td>GROUP BY S.rating</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>40.5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>25.5</td>
<td></td>
</tr>
</tbody>
</table>
EX16: Find the average age of sailors for each rating level that has at least two sailors.

<table>
<thead>
<tr>
<th>Rating</th>
<th>avg_age</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>44.5</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>40.5</td>
</tr>
<tr>
<td>10</td>
<td>25.5</td>
</tr>
</tbody>
</table>

EX16: An example shows difference between WHERE and HAVING:

<table>
<thead>
<tr>
<th>Rating</th>
<th>avg_age</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>7</td>
<td>45</td>
</tr>
<tr>
<td>8</td>
<td>55.5</td>
</tr>
</tbody>
</table>

5. NULL value and OUTER JOIN

In the presence of null values, any row that evaluates to false or to unknown is eliminated.

The two rows are duplicates if corresponding columns are either equal, or both contain null.
(If we compare two null values using =, the result is unknown)

The arithmetic operation +, -, *, and / all return null if one of their arguments is null.

Count(*) handle null values just like other values. All the other aggregate operations (COUNT, SUM, AVG, MAX, MIN, and variations using DISTINCT) simply discard null values.

After: INSERT INTO sailors
          ( sid, sname, rating, age )
     VALUES ( 99, 'Dan', null, 48.0 ) ,

     SELECT COUNT(*) FROM Sailors    will return 11
     SELECT COUNT(rating) FROM Sailors will return 10
     SELECT COUNT(age) FROM Sailors    will return 11
An example of OUTER JOIN:

```
SELECT sailors.sid, sailors.sname, reserves.bid
FROM sailors LEFT OUTER JOIN reserves ON reserves.sid = sailors.sid
ORDER BY sailors.sid
```

<table>
<thead>
<tr>
<th>sid</th>
<th>surname</th>
<th>bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>101</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>102</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>103</td>
</tr>
<tr>
<td>22</td>
<td>Dustin</td>
<td>104</td>
</tr>
<tr>
<td>29</td>
<td>Brutus</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>102</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>103</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>104</td>
</tr>
<tr>
<td>32</td>
<td>Andy</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>101</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>102</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>103</td>
</tr>
<tr>
<td>85</td>
<td>Art</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>Bob</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>Dan</td>
<td></td>
</tr>
</tbody>
</table>