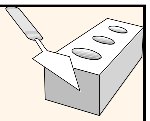
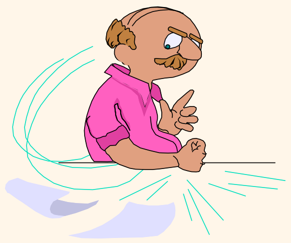


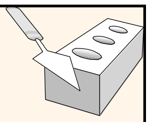
Introduction to Data Management



Lecture #15
(SQL, the Sequel…)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements



❖ HW stuff
- Hopefully everyone's feeling RelaXed today. ☺
- HW4's solution will appear tomorrow at 5pm.
- There will be no actual TRC HW (just quiz stuff).

❖ Other logistical stuff
- Midterm grading is still in progress…
- We will now continue our SQL adventure!

Grouped Aggregation Queries **(Review)**

```
SELECT      [DISTINCT]  target-list
FROM        relation-list
WHERE       qualification
GROUP BY    grouping-list
HAVING      group-qualification
```

For example…

```
SELECT  S.rating, MIN(S.age) AS minage
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT(*) >= 2
```

Group aggregate(s)

Grouping field(s)

Group predicate(s)

Find age of the youngest sailor with age ≥18 for each rating with at least 2 such sailors. **(Review)**

Query:

```
SELECT  S.rating, MIN(S.age) AS minage
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT(*) >= 2
```

Sailors instance:

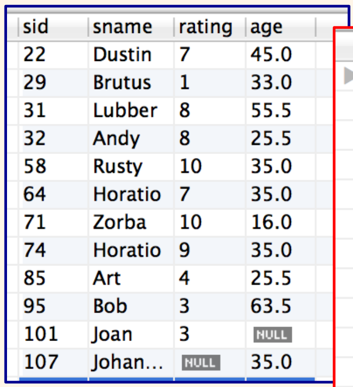
sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Answer relation:

rating	minage
3	25.5
7	35.0
8	25.5

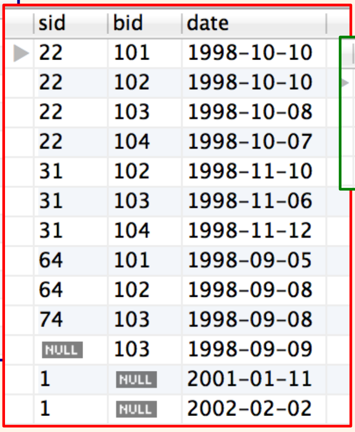
Example Data in MySQL

Sailors



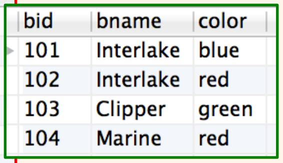
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johan...	NULL	35.0

Reserves



sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Boats



bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

For each red boat, find the number of reservations for this boat

for each的就是用group

```
SELECT  B.bid, COUNT(*) AS scount
FROM Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color='red'
GROUP BY B.bid
```

- ❖ We're grouping over a join of three relations!
- ❖ What do we get if we remove *B.color= 'red'* from the WHERE clause and add a HAVING clause with this condition? (**Hint**: Trick question... ☺)
- ❖ What if we drop Sailors and the condition involving S.sid?

*Find age of the youngest sailor with age > 18 for each rating with at least 2 sailors (of **any** age)*

```
SELECT  S.rating,  MIN(S.age)
FROM  Sailors S
WHERE  S.age > 18
GROUP BY  S.rating
HAVING  1  <  (SELECT  COUNT(*)
               FROM  Sailors S2
               WHERE  S.rating=S2.rating)
```

- Shows HAVING clause can also contain a subquery.
- Compare this with the query where we considered only ratings with 2 or more sailors over 18!
- What if HAVING clause were replaced by:
  - HAVING COUNT(*) >1

Find those ratings and average ages for which the average Sailor age is the minimum age over all of the Sailors

- Aggregate operations can't be nested! *(WRONG...)*

~~SELECT S.rating~~
~~FROM Sailors S~~
~~WHERE S.age = (SELECT MIN(AVG (S2.age)) FROM Sailors S2)~~

- A correct solution (in SQL/92):

```
SELECT  Temp.rating, Temp.avgage
FROM  (SELECT  S.rating, AVG(S.age) AS avgage
       FROM  Sailors S
       GROUP BY  S.rating) AS Temp
WHERE  Temp.avgage = (SELECT  MIN(age) FROM Sailors)
```

Compute the average age for *each rating*...

Find the *overall* minimum age

SQL's WITH Clause

Ex: *Find those ratings and average ages for which the average Sailor age is the minimum age over all of the Sailors*

- Our first solution was:

```
SELECT  Temp.rating, Temp.avgage
FROM  (SELECT  S.rating, AVG(S.age) AS avgage
          FROM  Sailors S
          GROUP BY  S.rating) AS Temp
WHERE  Temp.avgage = (SELECT  MIN(age) FROM Sailors)
```

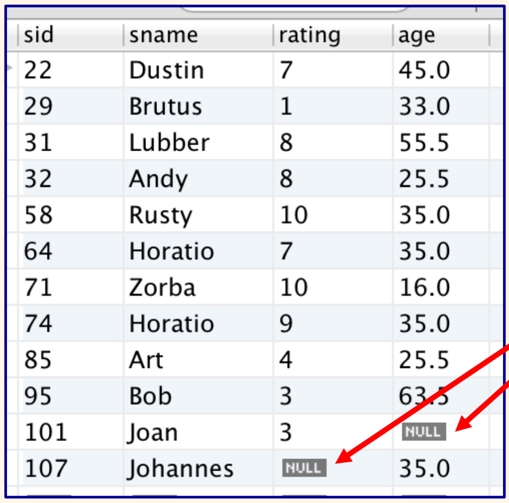
- We could use a ***WITH*** clause here for clarity:

```
WITH Temp AS (SELECT  S.rating, AVG(S.age) AS avgage
         FROM  Sailors S
         GROUP BY S.rating)
SELECT  Temp.rating, Temp.avgage
FROM Temp
WHERE  Temp.avgage = (SELECT  MIN(age) FROM Sailors)
```

Null Values in SQL

- Field values in a tuple are sometimes *unknown* (e.g., a rating has not yet been assigned) or *inapplicable* (e.g., there is no spouse's name).
  - SQL provides the special value *null* for such situations.
- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not *null*.
  - Is *rating>8* true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
  - We need a 3-valued logic (true, false and *unknown*).
  - Meaning of constructs must be defined carefully. (The WHERE clause eliminates rows that don't evaluate to ***true***.)
  - New operators (in particular, *outer joins*) possible/needed.

Ex: Sailors w/Some Null Values



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

Q: Which kind(s) of *null* are each of these null values?

Nulls and SQL's 3-Valued Logic

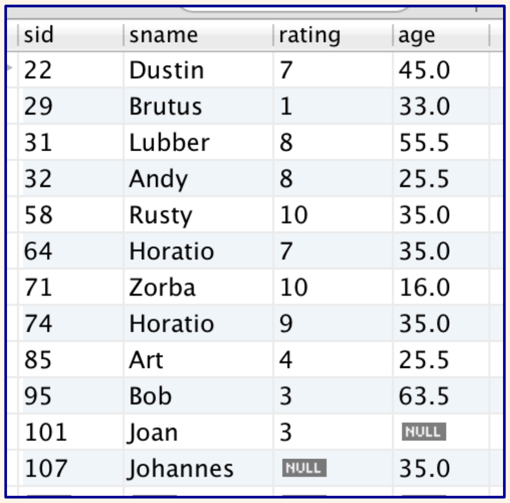
AND	**true**	**false**	**unknown**
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

OR	**true**	**false**	**unknown**
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

NOT	
true	false
false	true
unknown	unknown

Note: SQL arithmetic expressions involving *null* values will yield *null* values (*Ex:* EMP.sal + EMP.bonus)

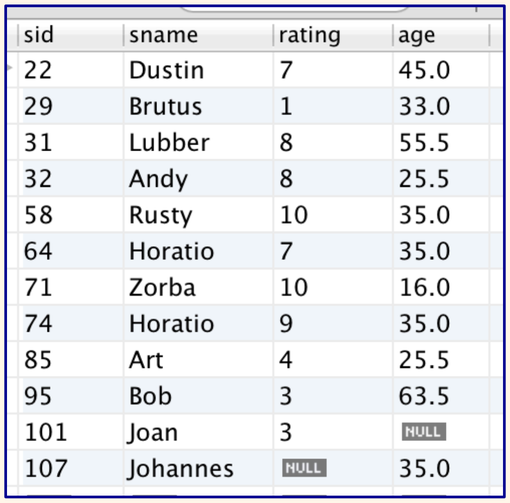
Basic SQL Queries w/Nulls



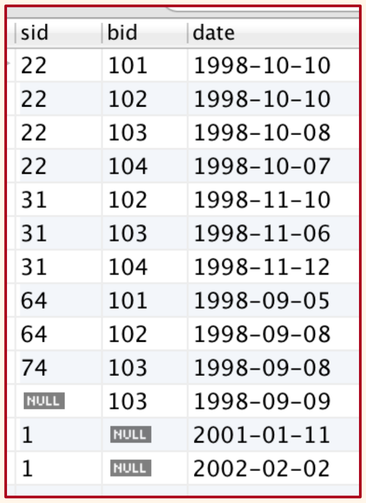
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

```
SELECT  *
FROM  Sailors S
WHERE age > 35.0

SELECT  *
FROM  Sailors S
WHERE age <= 35.0

SELECT COUNT(*)
FROM  Sailors S
WHERE age > 35.0
      OR age <= 35.0
      OR age IS NULL
```

Ex: Sailors ***and Reserves*** *w/Nulls*

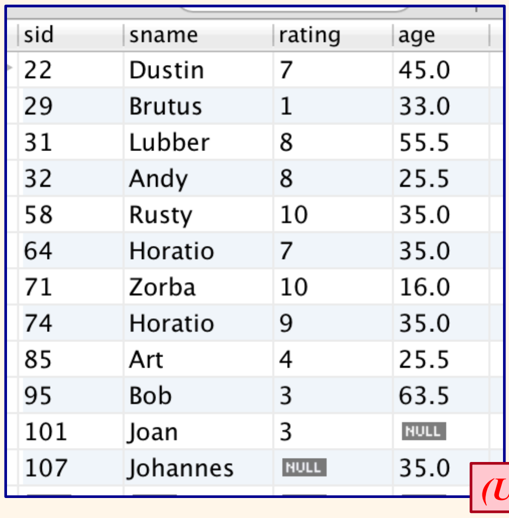


sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0



sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Nulls w/Aggregates



sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

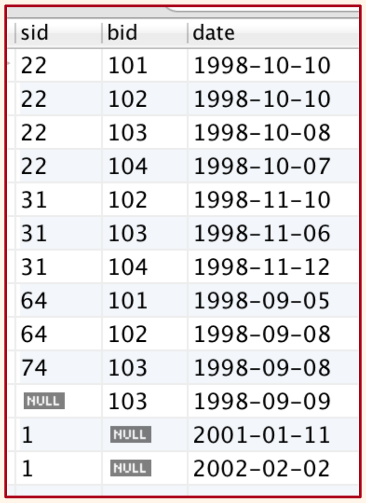
SELECT COUNT(rating)
FROM Sailors (11) Null不算

SELECT
COUNT (DISTINCT rating)
FROM Sailors (7)

SELECT SUM(rating), Null不算
COUNT(rating),
AVG(rating)
FROM Sailors
(70, 11, 6.3636)

(Useful, but logically "wrong"!)

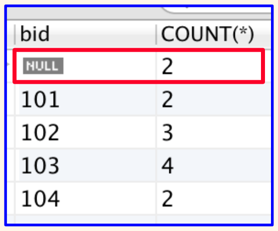
Nulls w/Aggregates & ***Grouping***



sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

SELECT COUNT(DISTINCT bid)
FROM Reserves (4) grouping会有

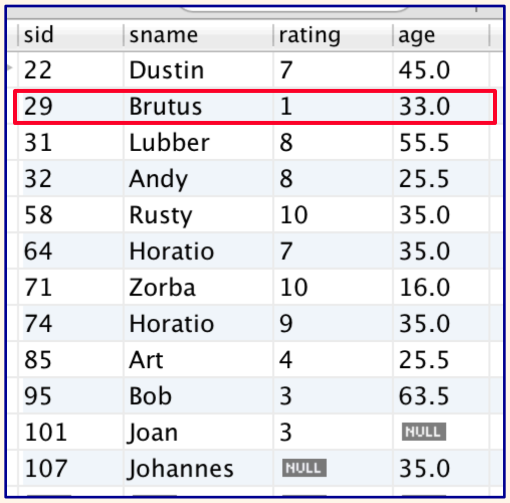
SELECT bid, COUNT(*)
FROM Reserves
GROUP BY bid



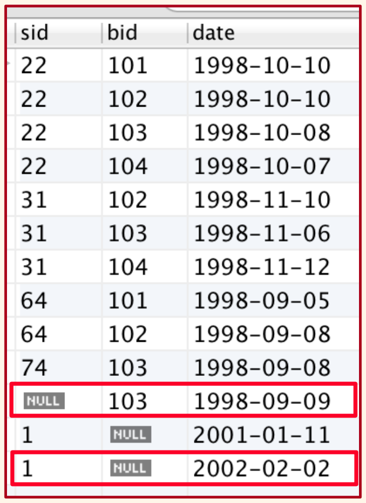
bid	COUNT(*)
NULL	2
101	2
102	3
103	4
104	2

Some "dangling" tuple examples

Nulls w/***Joins*** → Inner vs. Outer Joins



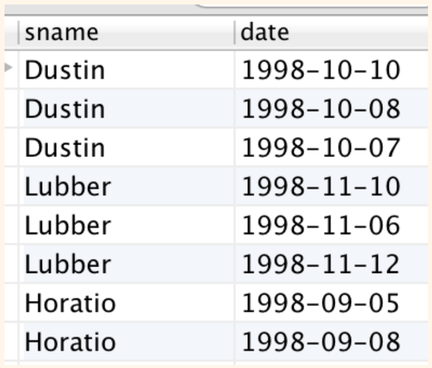
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0



sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Inner vs. Outer Joins in SQL

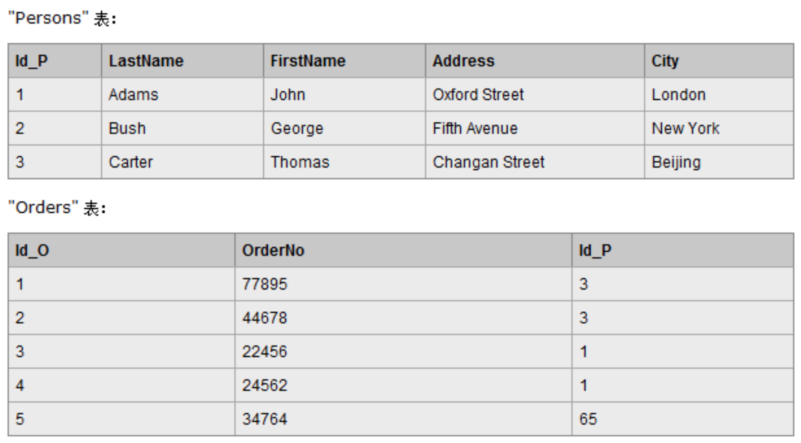
```
SELECT DISTINCT s.sname, r.date
FROM Sailors s, Reserves r
WHERE s.sid = r.sid
```



sname	date
Dustin	1998-10-10
Dustin	1998-10-08
Dustin	1998-10-07
Lubber	1998-11-10
Lubber	1998-11-06
Lubber	1998-11-12
Horatio	1998-09-05
Horatio	1998-09-08

SQL中INNER JOIN、LEFT JOIN、RIGHT JOIN、FULL JOIN区别

sql中的连接查询有inner join(内连接）、left join(左连接)、right join（右连接）、full join（全连接）四种方式，它们之间其实并没有太大区别，仅仅是查询出来的结果有所不同。
例如我们有两张表：



"Persons" 表：

Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

"Orders" 表：

Id_O	OrderNo	Id_P
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	65

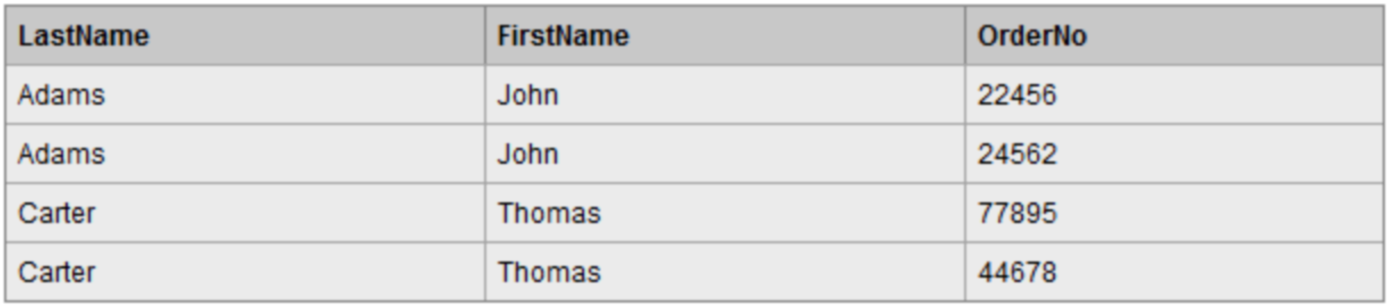
Orders表通过外键Id_P和Persons表进行关联。

1.inner join，在两张表进行连接查询时，只保留两张表中完全匹配的结果集。

我们使用inner join对两张表进行连接查询，sql如下：

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.Id_P=Orders.Id_P
ORDER BY Persons.LastName
```

查询结果集：



LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678

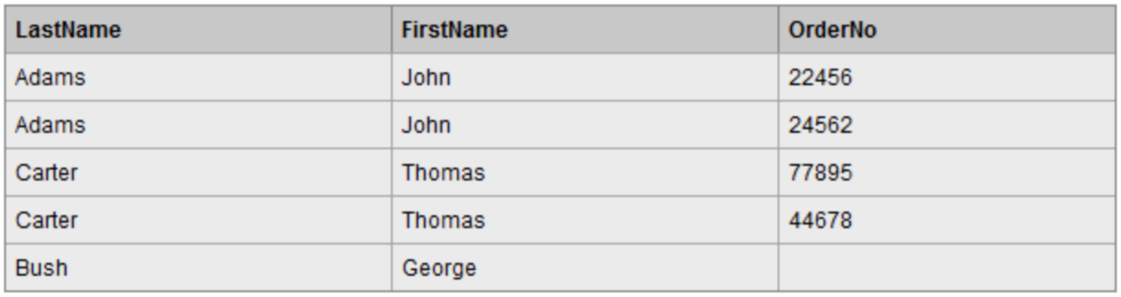
此种连接方式Orders表中Id_P字段在Persons表中找不到匹配的，则不会列出来。

2.left join,在两张表进行连接查询时，会返回左表所有的行，即使在右表中没有匹配的记录。

我们使用left join对两张表进行连接查询，sql如下：

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.Id_P=Orders.Id_P
ORDER BY Persons.LastName
```

查询结果如下：



LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678
Bush	George	

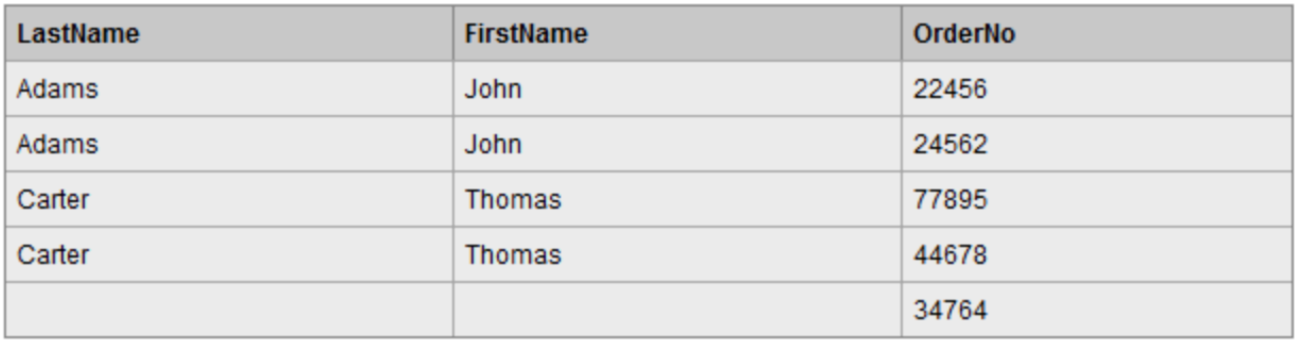
可以看到，左表（Persons表）中LastName为Bush的行的Id_P字段在右表（Orders表）中没有匹配，但查询结果仍然保留该行。

3.right join,在两张表进行连接查询时，会返回右表所有的行，即使在左表中没有匹配的记录。

我们使用right join对两张表进行连接查询，sql如下：

```sql
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.Id_P=Orders.Id_P
ORDER BY Persons.LastName
```

查询结果如下：



LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678
		34764

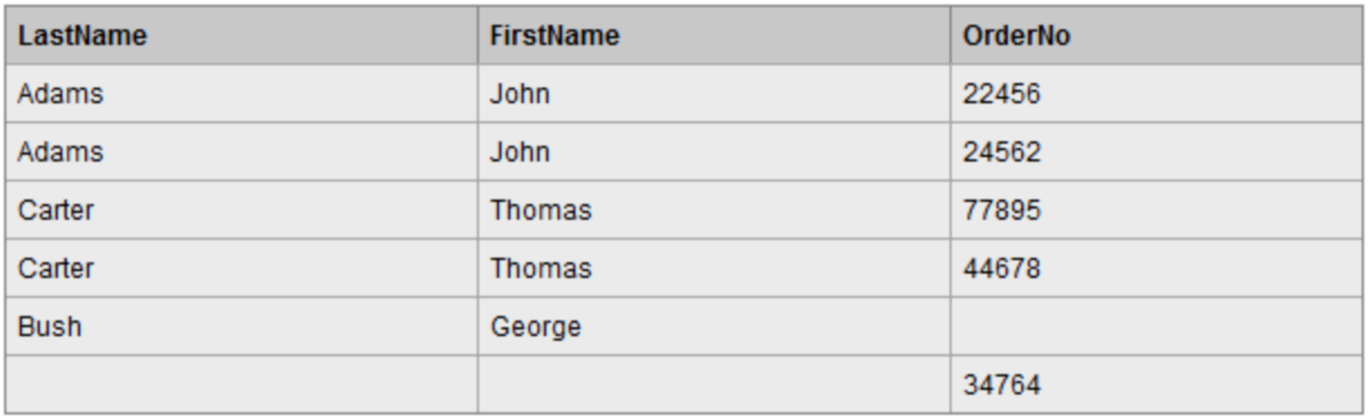
Orders表中最后一条记录Id_P字段值为65，在左表中没有记录与之匹配，但依然保留。

4.full join,在两张表进行连接查询时，返回左表和右表中所有没有匹配的行。

我们使用full join对两张表进行连接查询，sql如下：

```sql
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.Id_P=Orders.Id_P
ORDER BY Persons.LastName
```

查询结果如下：



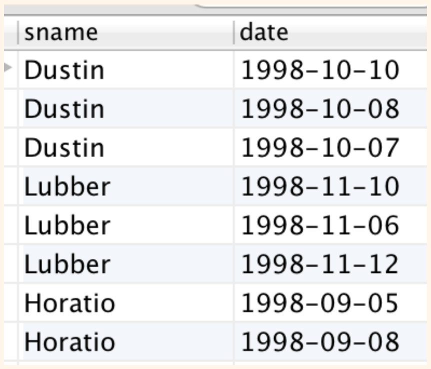
LastName	FirstName	OrderNo
Adams	John	22456
Adams	John	24562
Carter	Thomas	77895
Carter	Thomas	44678
Bush	George	
		34764

查询结果是left join和right join的并集。

这些连接查询的区别也仅此而已。

Inner vs. Outer Joins in SQL (2)

SELECT DISTINCT s.sname, r.date
FROM Sailors s INNER JOIN Reserves r ON s.sid = r.sid



sname	date
Dustin	1998-10-10
Dustin	1998-10-08
Dustin	1998-10-07
Lubber	1998-11-10
Lubber	1998-11-06
Lubber	1998-11-12
Horatio	1998-09-05
Horatio	1998-09-08

(“INNNER” is optional, and will be the default type of JOIN assumed if one isn’t specified)

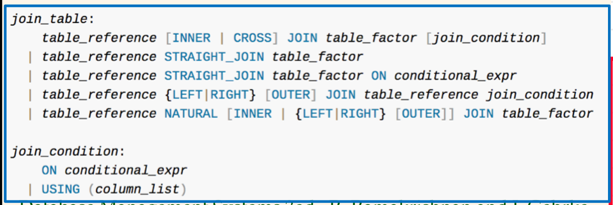
本来就只显示inner join

Inner vs. *Outer* Joins in SQL (3)

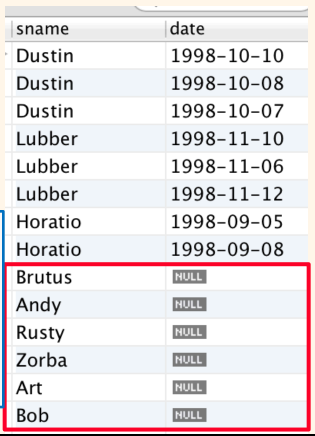
(1) SELECT DISTINCT s.sname, r.date
FROM Sailors s LEFT OUTER JOIN Reserves r ON s.sid = r.sid

(2) SELECT DISTINCT s.sname, r.date
FROM Reserves r RIGHT OUTER JOIN Sailors s ON s.sid = r.sid

- Variations on a theme:
  - JOIN (or INNER JOIN)
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN



```
join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [INNER | {LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
  | USING (column_list)
```



sname	date
Dustin	1998-10-10
Dustin	1998-10-08
Dustin	1998-10-07
Lubber	1998-11-10
Lubber	1998-11-06
Lubber	1998-11-12
Horatio	1998-09-05
Horatio	1998-09-08
Brutus	NULL
Andy	NULL
Rusty	NULL
Zorba	NULL
Art	NULL
Bob	NULL