Database Applications (15-415)

SQL-Part II Lecture 7, February 3, 2014

Mohammad Hammoud

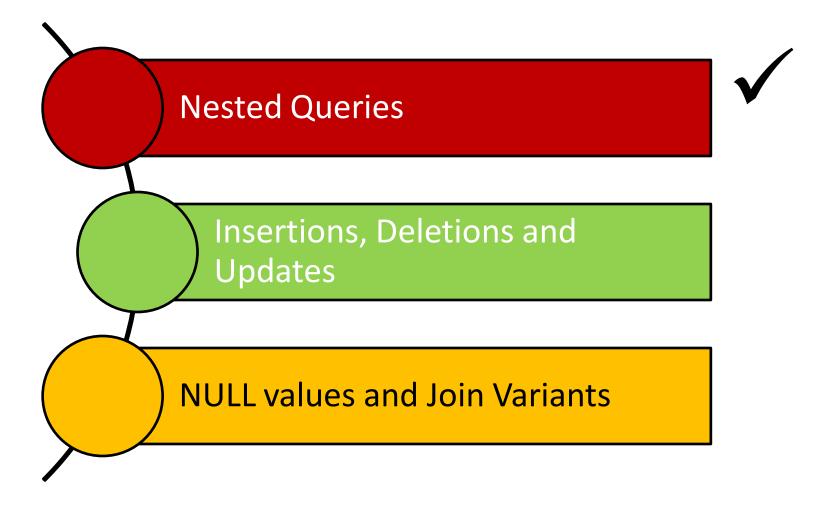


Today...

- Last Session:
 - Standard Query Language (SQL)- Part I
- Today's Session:
 - Standard Query Language (SQL)- Part II
- Announcements:
 - PS2 is due on Feb 07, 2014 by midnight
 - Quiz I is on Monday 10, 2014 (all topics included except next lecture's material on storing data)
 - Project I will be posted by tomorrow. It is due on Feb
 18 by midnight

Carnegie Mellon University Qatar

Outline



A Join Query

Find the names of sailors who have reserved boat 101

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

Reserves			
Sid Bid Day			
22	101	10/10/2013	
22	102	10/10/2013	

select S.sname
from Sailors S, Reserves R
where S.sid = R.sid
and R.bid = 101

Nested Queries

Find the names of sailors who have reserved boat 101

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

Reserves			
Sid Bid Day			
22	101	10/10/2013	
22	102	10/10/2013	

OR...

SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=101)

IN compares a value with a set of values

Nested Queries

Find the names of sailors who have <u>not</u> reserved boat 101

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

Reserves			
Sid Bid Day			
22	101	10/10/2013	
22	102	10/10/2013	

SELECT S.sname
FROM Sailors S...
WHERE S.sid NOT IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=101)

Deeply Nested Queries

Find the names of sailors who have reserved a red boat

Sailors			
Sid Sname Rating age			
22	Dustin	7	45.0
29	Brutus	1	33.0

Reserves		
Sid Bid		Day
22	101	10/10/2013
22	102	10/10/2013

Boats			
Bid	Bname	Color	
101	Interlake	Red	
102	Clipper	Green	

```
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'))
```

In principle, queries with very deeply nested structures are possible!



Reserves instance:

Boats 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

sid	<u>bid</u>	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

<u>Bid</u>	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

Deeply Nested Queries

Find the names of sailors who have <u>not</u> reserved a red boat

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

Reserves			
Sid	Bid	Day	
22	101	10/10/2013	
22	102	10/10/2013	

Boats			
Bid	Bname	Color	
101	Interlake	Red	
102	Clipper	Green	

```
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'))
```



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
7 1	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Reserves instance:

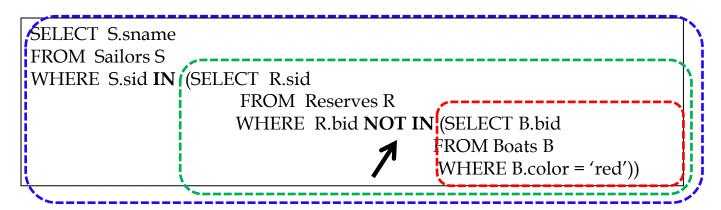
<u>sid</u>	<u>bid</u>	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats instance:

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

This returns the names of sailors who have <u>not</u> reserved a red boat!





R	Leserves	inst	tance:

sid	sname	rating	age	<u>sid</u>	<u>bid</u>	day
22	dustin	7	45.0	 22	101	10/10/98
29	brutus	1	33.0	22	102	10/10/98
31	lubber	8	55.5	22	103	10/8/98
32	andy	8	25.5	22	104	10/7/98
58	rusty	10	35.0	31	102	11/10/98
64	horatio	7	35.0		100	
71	zorba	10	16.0	31	103	11/6/98
/ 1	2010a	10	10.0	31	104	11/12/98
74	horatio	9	35.0	,	101	11/12/90
85	art	3	25.5	64	101	9/5/98
				64	102	9/8/98
95	bob	3	63.5	04	102	9/0/90
96	frodo	3	25.5	74	103	9/8/98
96	frodo	3	25.5	74	103	9/8/98

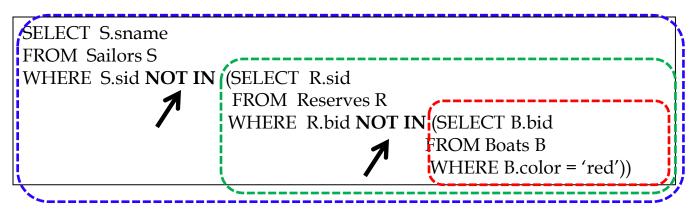
Boats instance:

<u>Bid</u>	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

This returns the names of sailors who have reserved a boat that is <u>not</u> red.



The previous one returns the names of sailors who have <u>not</u> reserved a red boat!



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

Reserves instance:

<u>sid</u>	<u>bid</u>	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats instance:

Bid	Bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

This returns the names of sailors who have <u>not</u> reserved a boat that is <u>not</u> red!



As such, it returns names of sailors who have reserved <u>only</u> red boats (*if any*)

Correlated Nested Queries

Find the names of sailors who have reserved boat 101

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

Reserves			
Sid Bid Day			
22	101	10/10/2013	
22	102	10/10/2013	

Compares a value with a set of values

SELECT S.sname
FROM Sailors S.
WHERE S.sid IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=101)

Allows us to test whether a set is "nonempty"

SELECT S.snamé
FROM Sailors S
WHERE (EXISTS) (SELECT *
FROM Reserves R
WHERE R.bid=101
AND R.sid = S.sid)

A correlation

Correlated Nested Queries

Find the names of sailors who have <u>not</u> reserved boat 101

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

Reserves			
Sid Bid Day			
22	101	10/10/2013	
22	102	10/10/2013	

SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=101)

SELECT S.sname
FROM Sailors S
WHERE (NOT EXISTS) (SELECT *
FROM Reserves R
WHERE R.bid=101
AND R.sid = S.sid)

Find sailors whose rating is better than <u>some</u> sailor called Dustin

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

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

Q: What if there were *no* sailors called Dustin?

A: An empty set is returned!

Find sailors whose rating is better than <u>every</u> sailor called Dustin

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

```
SELECT S.sname
FROM Sailors S
WHERE S.rating > ALL (SELECT S2. rating
FROM Sailors S2
WHERE S2.name = 'Dustin')
```

Q: What if there were no sailors called Dustin?

A: The names of *all* sailors will be returned! (*Be Careful*)

Find sailors with the highest sid

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

SELECT *
FROM Sailors S
WHERE S.sid

is greater than every other sid

Find sailors with the highest sid

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

SELECT *
FROM Sailors S
WHERE S.sid

is greater than every

(SELECT S2.sid FROM Sailors S2)

Find sailors with the highest sid

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

SELECT *
FROM Sailors S
WHERE S.sid

> ALL

(SELECT S2.sid FROM Sailors S2)



Find sailors with the highest sid

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

SELECT *
FROM Sailors S
WHERE S.sid

>= **ALL**

(SELECT S2.sid FROM Sailors S2)

Now Correct!

Find sailors with the highest sid- without nested subquery

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

```
SELECT *
FROM Sailors S1, Sailors S2
WHERE S1.sid > S2.sid
```

Q: What does this give?

Find sailors with the highest sid- without nested subquery

S1 S2

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

S1.sid > S2.sid

S1.Sid	S2.sid	••••
22	22	
22	29	
29	22	
29	29	

Find sailors with the highest sid- without nested subquery

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

SELECT *
FROM Sailors S1, Sailors S2
WHERE S1.sid > \$2.sid

Q: What does this give?

A: All but the smallest sid!

Find sailors with the highest sid- without nested subquery

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

SELECT *
FROM Sailors S1, Sailors S2
WHERE S1.sid < \$2.sid

Q: What does this give?

A: All but the highest sid!

Find sailors with the highest sid- without nested subquery

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

Therefore...

(SELECT *

FROM Sailors)

EXCEPT

(SELECT S1.sid, S1.sname, S1.rating, S1.age

FROM Sailors S1, Sailors S2

WHERE S1.sid < S2.sid)

I.e., ALL - (ALL - Highest) = Highest



Alternative Ways

Find sailors with the highest sid

Sailors			
Sid Sname Rating age			
22	Dustin	7	45.0
29	Brutus	1	33.0

(SELECT *
FROM Sailors)
EXCEPT
(SELECT S1.sid, S1.sname, S1.rating, S1.age
FROM Sailors S1, Sailors S2
WHERE S1.sid < S2.sid)



SELECT *
FROM Sailors S
WHERE S.sid
>= ALL
(SELECT S2.sid
FROM Sailors S2)

Revisit: Another Example

 Find the names of sailors who have reserved both 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 = 'green')
intersect
(select S2.sname from Sailors S2, Reserves R2, Boats B2
where S2.sid = R2.sid and R2.bid = B2.bid and B2.color = 'red')
```

The query contains a "subtle bug" which arises because we are using sname to identify Sailors, and "sname" is not a key for Sailors!

If we ought to compute the names of such Sailors, we would need a NESTED QUERY

A Correct Way

 Find the names of sailors who have reserved both 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 = 'green')
AND S.sid IN)
(select S2.sid from Sailors S2, Reserves R2, Boats B2
where S2.sid = R2.sid and R2.bid = B2.bid and B2.color = 'red')
```

Similarly, queries using EXCEPT can be re-written using NOT IN

Revisit: Another Example

Find the name and age of the oldest sailor

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

select S.sname, max (S.age) from Sailors S

This query is illegal in SQL- If the "select" clause uses an aggregate function, it must use ONLY aggregate function unless the query contains a "group by" clause!

A Correct Way

Find the name and age of the oldest sailor

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

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

Alternative Ways

Find the name and age of the oldest sailor

Sailors				
Sid Sname Rating age				
22	Dustin	7	45.0	
29	Brutus	1	33.0	

SELECT S.sname, MAX(S.age)VS. FROM Sailors SGROUP BY S.sname

Revisit: Another Example

Find age of the youngest sailor with age ≥ 18, for each rating level with at least 2 such sailors

Sailors			
Sid Sname Rating age			
22	Dustin	7	45.0
29	Brutus	1	33.0

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

An Alternative Way

Find age of the youngest sailor with age ≥ 18, for each rating level with at least 2 such sailors

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

The HAVING clause can include subqueries!

OR...

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

Yet Another Way

Find age of the youngest sailor with age ≥ 18, for each rating level with at least 2 such sailors

Sailors			
Sid	Sname	Rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0

The FROM clause can include subqueries!

Necessary!

OR...

FROM (SELECT S.rating, MIN(S.age) AS minage,
COUNT(*) AS ratingcount
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating) AS (Temp)
WHERE Temp.ratingcount > 1

Expressing the Division Operator in SQL

■ Find the names of sailors who have reserved *all* boats

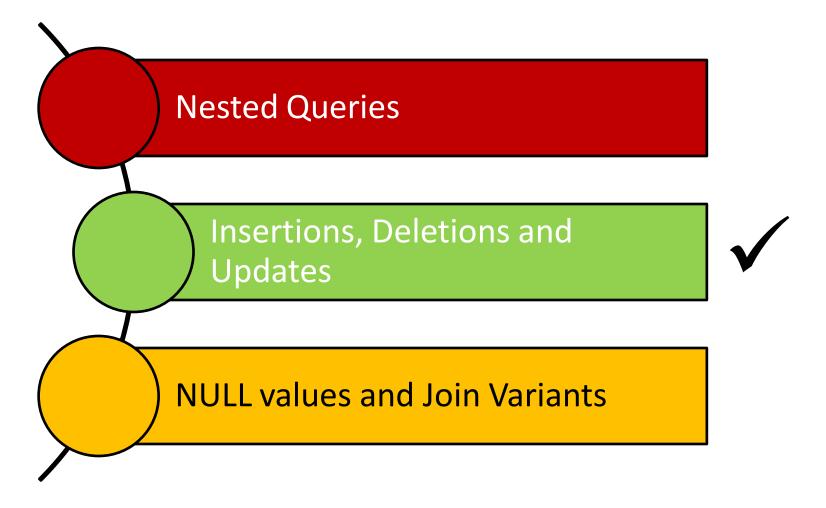
Sailors				
Sid	Sname	Rating	age	
22	Dustin	7	45.0	
29	Brutus	1	33.0	

Reserves				
Sid	Bid	Day		
22	101	10/10/2013		
22	102	10/10/2013		

Boats				
Bid	Bname	Color		
101	Interlake	Red		
102	Clipper	Green		

```
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))
```

Outline



Reminder: Our Mini-U DB

STUDENT		
<u>Ssn</u>	Name	Address
123	smith	main str
234	jones	QF ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
<u>SSN</u>	c-id	grade
123	15-413	Α
234	15-413	В

Revisit: Insertions

insert into student(ssn, name, address)
values (123, 'smith', 'main')

OR...

insert into student
values (123, 'smith', 'main')

Bulk Insertions

How to insert, say, a table of 'foreignstudent's, in bulk?

insert into student
 select ssn, name, address
 from foreign-student

Revisit: Deletions

Delete the record of 'smith'

delete from student
where name='smith'

Be careful - it deletes ALL the 'smith's!

Revisit: Updates

 Update the grade to 'A' for ssn=123 and course 15-415

```
update takes
set grade='A'
where ssn = 123 and c-id= '15-415'
```

Updating Views

Consider the following view:

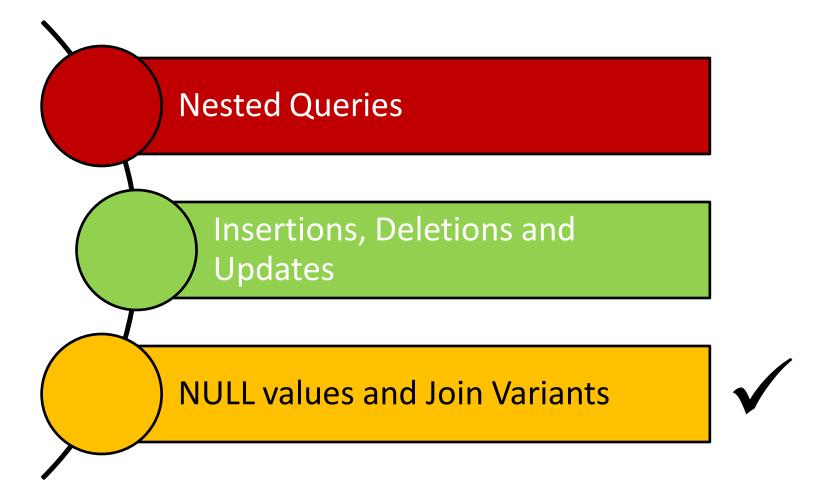
```
create view db-takes as
  (select * from takes where c-id="15-415")
```

- What if c-id is modified to '15-440'?
- What if c-id is deleted?

A Rule of thumb: A command that affects a row in the view affects all corresponding rows in underlying tables!

View updates are tricky - typically, we can only update views that have no joins, nor aggregates!

Outline



NULL Values

- Column values can be unknown (e.g., a sailor may not yet have a rating assigned)
- Column values may be inapplicable (e.g., a maiden-name column for men!)
- The NULL value can be used in such situations
- However, the NULL value complicates many issues!
 - Using NULL with aggregate operations
 - COUNT (*) handles NULL values like any other values
 - SUM, AVG, MIN, and MAX discard NULL values
 - Comparing NULL values to valid values
 - Comparing NULL values to NULL values

Comparing Values In the Presence of NULL

- Considering a row with rating = NULL and age = 20; what will be the result of comparing it with the following rows?
 - Rating = 8 OR age < 40 → TRUE</p>
 - Rating = 8 AND age < 40

 unknown
- In general:
 - NOT unknown → unknown
 - True OR unknown

 True
 - False OR unknown → unknown
 - False AND unknown → False
 - True AND unknown → unknown
 - Unknown [AND|OR(=) unknown → unknown

In the context of *duplicates*, the comparison of two NULL values is implicitly treated as TRUE (Anomaly!)

Comparing Values In the Presence of NULL

- Considering a row with rating = NULL and age = 20; what will be the result of comparing it with the following rows?
 - Rating = 8 OR age < 40 → TRUE</p>
 - Rating = 8 AND age < 40
 unknown
- In general:
 - NOT unknown
 unknown
 - True OR unknown
 True
 - False Three-Valued Logic!
 - False AND unknown False
 - True AND unknown

 unknown
 - Unknown [AND|OR|=] unknown

 unknown

Inner Join

- Tuples of a relation that do not match some rows in another relation (according to a join condition c) do not appear in the result
 - Such a join is referred to as "Inner Join" (so far, all inner joins)

select ssn, c-name
from takes, class
where takes.c-id = class.c-id

Equivalently:

select ssn, c-name
from takes join class on takes.c-id = class.c-id

Inner Join

• Find all SSN(s) taking course s.e.

TAKES		
<u>SSN</u>	c-id	grade
123	15-413	Α
234	15-413	В

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

<u>SSN</u>	<u>c-name</u>
123	s.e
234	s.e

o.s.: gone!

Outer Join

- But, tuples of a relation that do not match some rows in another relation (according to a join condition c) can still appear exactly once in the result
 - Such a join is referred to as "Outer Join"
 - Result columns will be assigned NULL values

select ssn, c-name
from takes outer join class
on takes.c-id=class.c-id

Outer Join

Find all SSN(s) taking course s.e.

TAKES		
<u>SSN</u>	c-id	grade
123	15-413	Α
234	15-413	В

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

<u>SSN</u>	<u>c-name</u>
123	s.e
234	s.e.
null	o.s.

Joins

In general:

```
select [column list]
from table_name
  [inner | {left | right | full} outer ] join
  table_name
  on qualification_list
Where ...
```

Summary

- Nested Queries
 - IN, NOT IN, EXISTS, NOT EXISTS, op ANY and op ALL where op ϵ {<. <=, =, <>, >=, >}
 - Re-writing INTERSECT using IN
 - Re-writing EXCEPT using NOT IN
 - Expressing the division operation using NOT EXISTS and EXCEPT (there are other ways to achieve that!)
- Other DML commands: INSERT (including bulk insertions), DELETE and UPDATE (for tables and views)
- Null values and inner vs. outer Joins

Next Class

SQL- Part III & Storing Data: Disks and Files (*if time allows*)