SQL Joins and More

The SELECT Statement

- We have seen the SELECT statement in the past
- Looked like this:
  - SELECT [list of attributes] FROM [table of things] WHERE [some condition];
- Works for selecting things from a single table
- May have seen that there are some things we can't do

Selecting Related Things

- In particular we have no way to select a bunch of related things

bsg_cert_people

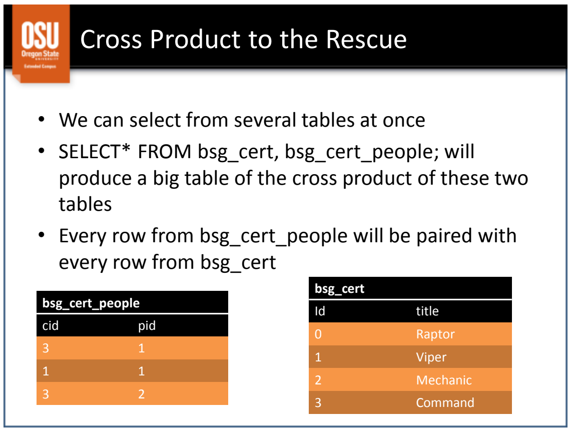
cid	pid
3	1
1	1
3	2

bsg_cert

Id	title
0	Raptor
1	Viper
2	Mechanic
3	Command

bsg_people

Id	Fname	Lname	Homeworld	Age
0	William	Adama	2	61
1	Lee	Adama	2	30
2	Laura	Roslin	2	NULL
3	Kara	Thrace	2	NULL



Cross Product to the Rescue

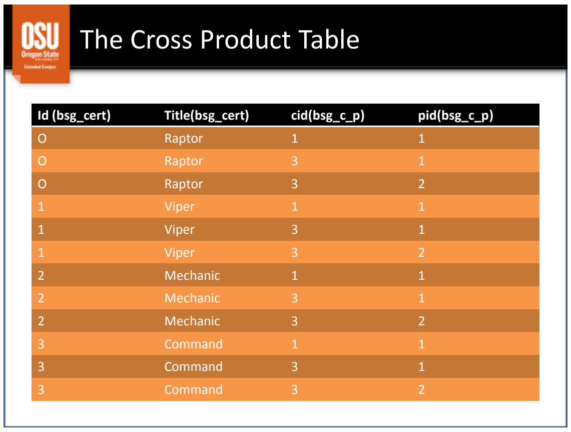
- We can select from several tables at once
- SELECT* FROM bsg_cert, bsg_cert_people; will produce a big table of the cross product of these two tables
- Every row from bsg_cert_people will be paired with every row from bsg_cert

bsg_cert_people

cid	pid
3	1
1	1
3	2

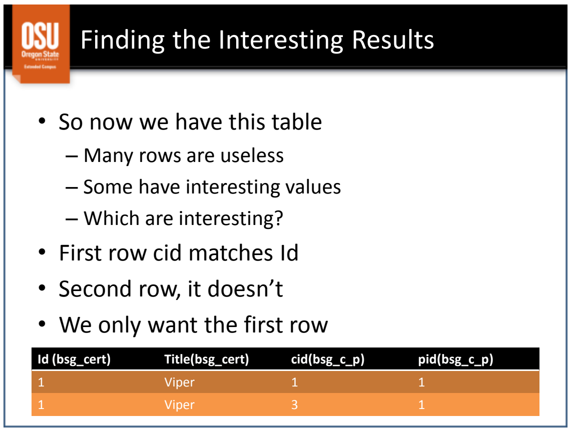
bsg_cert

Id	title
0	Raptor
1	Viper
2	Mechanic
3	Command



The Cross Product Table

Id (bsg_cert)	Title(bsg_cert)	cid(bsg_c_p)	pid(bsg_c_p)
O	Raptor	1	1
O	Raptor	3	1
O	Raptor	3	2
1	Viper	1	1
1	Viper	3	1
1	Viper	3	2
2	Mechanic	1	1
2	Mechanic	3	1
2	Mechanic	3	2
3	Command	1	1
3	Command	3	1
3	Command	3	2



Finding the Interesting Results

- So now we have this table
  - Many rows are useless
  - Some have interesting values
  - Which are interesting?
- First row cid matches Id
- Second row, it doesn't
- We only want the first row

Id (bsg_cert)	Title(bsg_cert)	cid(bsg_c_p)	pid(bsg_c_p)
1	Viper	1	1
1	Viper	3	1

Working with Multiple Tables

Id (bsg_cert)	Title(bsg_cert)	cid(bsg_c_p)	pid(bsg_c_p)
1	Viper	1	1
1	Viper	3	1

- We want rows where id = cid
- We can change the query to this:
- SELECT * FROM bsg_cert, bsg_cert_people WHERE id = cid;

Id (bsg_cert)	Title(bsg_cert)	cid(bsg_c_p)	pid(bsg_c_p)
1	Viper	1	1
3	Command	3	1
3	Command	3	2

Now to Actually Get Something Meaningful

- We still need to pair it with the people table
- SELECT P.fname, P.lname, C.title FROM bsg_cert C, bsg_cert_people CP, bsg_people P WHERE C.id = CP.cid AND CP.pid = P.id;

What is Happening Here?

- SELECT P.fname, P.lname, C.title FROM
  - This is familiar but now we are saying what table we are selecting from
- FROM bsg_cert C, bsg_cert_people CP, bsg_people P
  - This gives us a huge cross product of the three tables
  - “bsg_cert C” aliases bsg_cert as C so we can type C.title instead of bsg_cert.title

What is Happening Here? Continued

- WHERE C.id = CP.cid AND CP.pid = P.id;
  - We only care about lines where the id's all match up
  - Remember the cross product gives us a bunch of meaningless rows

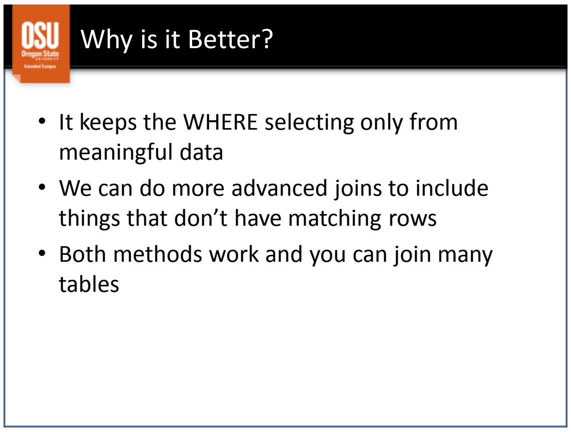
Surely there is a Better Way!

- What we had works OK for a table like this
- You see it a lot in older databases, but there is a better way

```
SELECT P.fname, P.lname, C.title FROM bsg_cert C
INNER JOIN bsg_cert_people CP ON CP.cid = C.id
INNER JOIN bsg_people P ON P.id = CP.pid;
```

So Now What Is Happening?

- SELECT P.fname, P.lname, C.title
  - Same as before
- FROM bsg_cert C INNER JOIN bsg_cert_people CP ON CP.cid = C.id
  - Selects rows where cert and cert_people IDs match up
- INNER JOIN bsg_people P ON P.id = CP.pid;
  - Selects frows where cert_people and people IDs match up



Why is it Better?

- It keeps the WHERE selecting only from meaningful data
- We can do more advanced joins to include things that don’t have matching rows
- Both methods work and you can join many tables