

SQL joins

CIS1152 Adv Web Dev

Lecture 9

Steve Ruegsegger

Outline

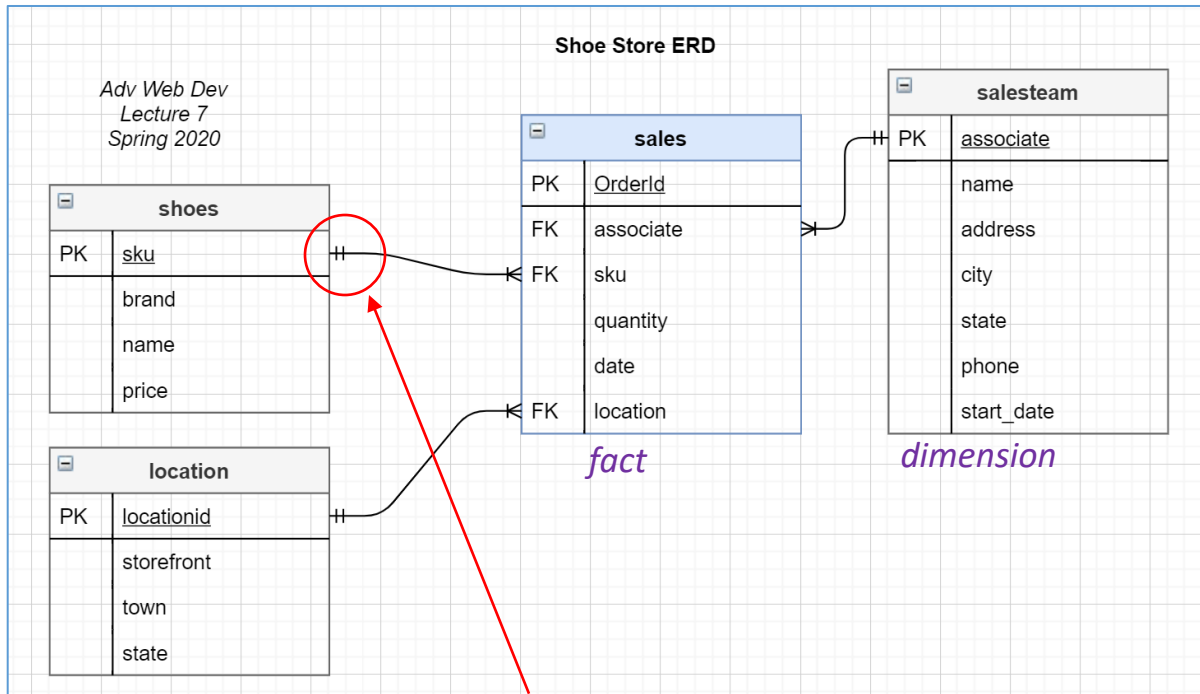
1. (postlude) many-to-many relationships
2. SQL joining process
3. SQL joining syntax
4. Detailed JOIN examples

1. Many to Many...

Junction table

Recall our previous shoe store

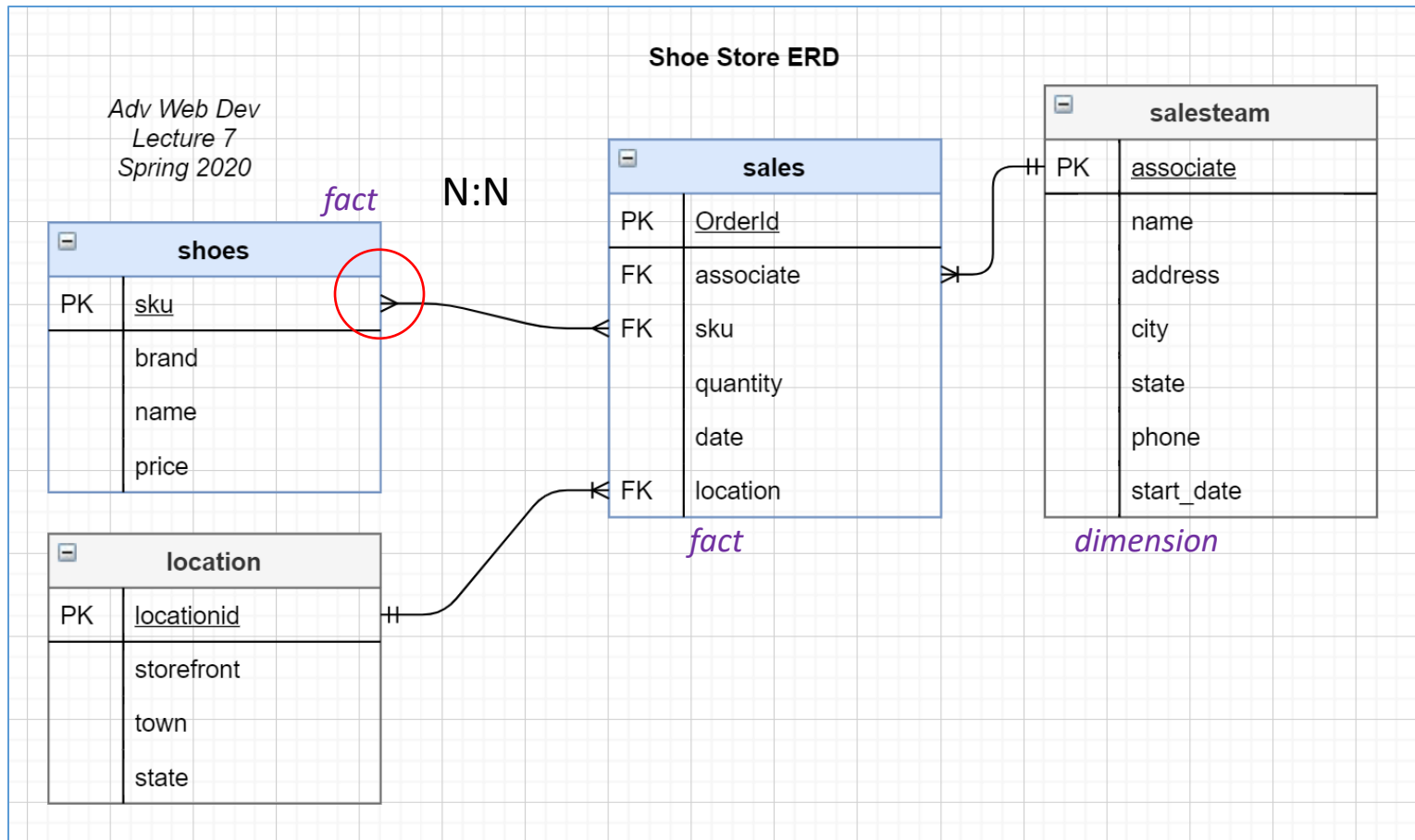
- Was anything bothering you?



- How about 1:N for sales to shoe?
- Sure, there can only be 1 location/sale, and only 1 sales associate / sale... But **only** 1 shoe-type per sale? (really?)
- Huh? What if someone wants to buy 2 different shoes in 1 order?

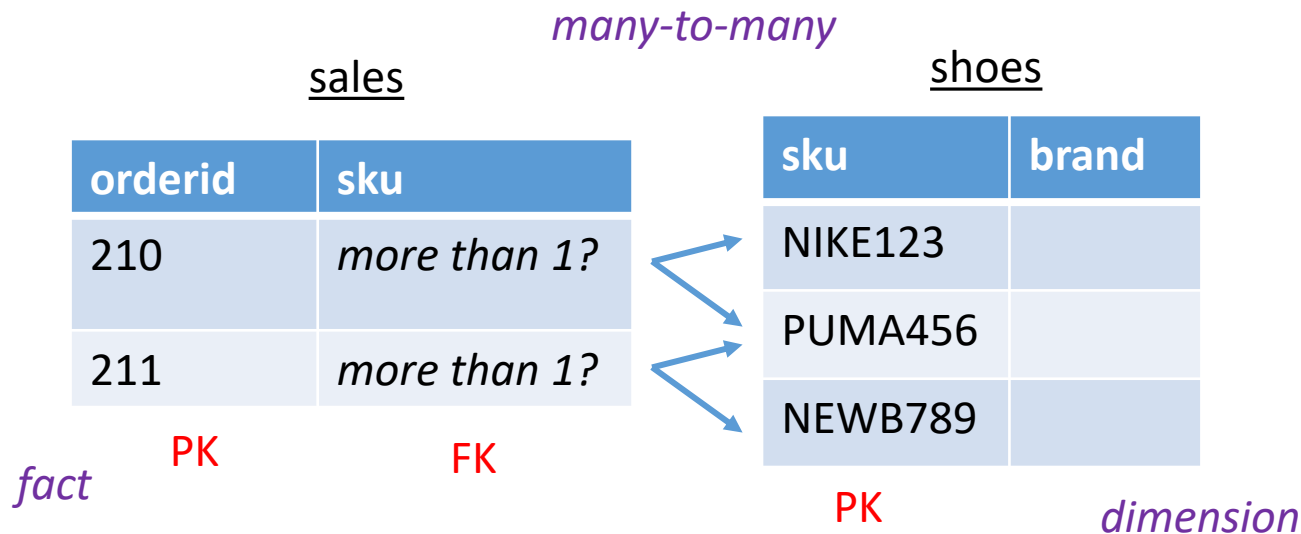
A more accurate data model

- A sale to a customer might have more than 1 (many) shoe types.
- e.g. order "123" might have a row for "Nike abc" and another row for "Puma def"



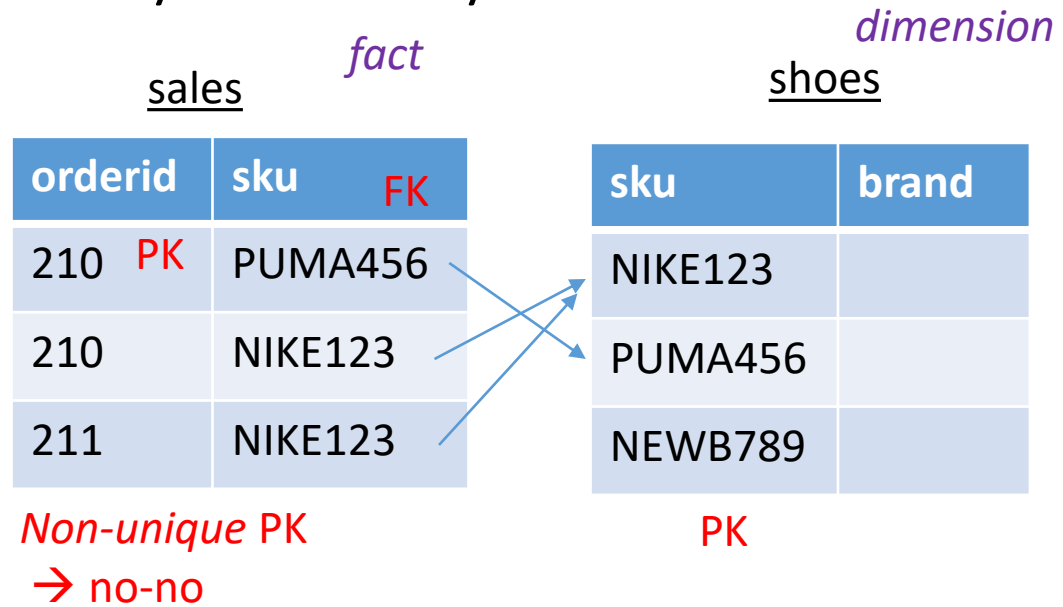
many-to-many

- What happens if we desire a sales order to have *many* different shoes?
 - That certainly makes sense – out there in the real world.
 - We don't want a data model which forces the sales person to ring up a different order for each shoe sku. That's odd...
- So, now a sales PK can point to many shoe SKUs, and a shoe SKU would be in many sales (of course). This is a *many-to-many relationship*.



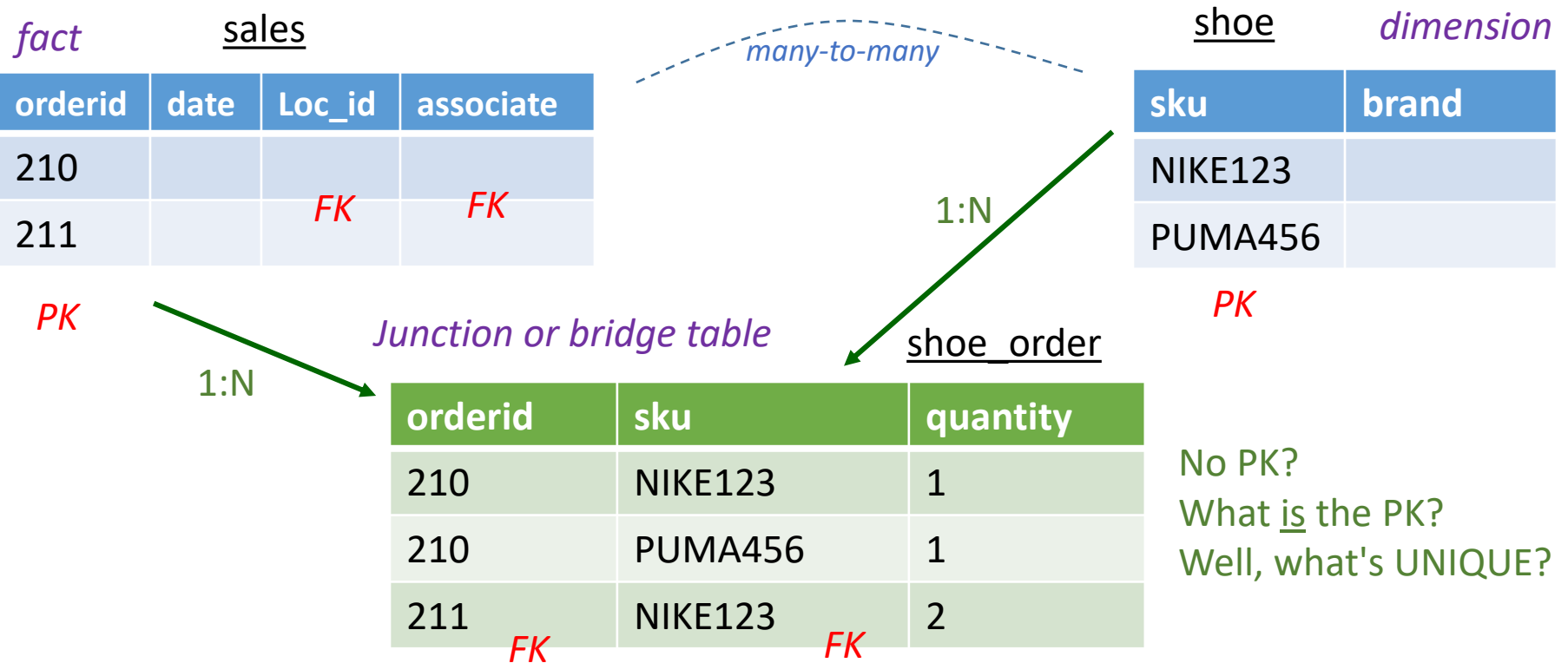
Q: What to do with many-to-many?

- We want an order to have multiple shoe types.
- We could...
 - Put each SKU on it's own line in sales table...
But you end up with non-unique PK's. (that's a *no-no*)
 - Or you could put a delimited string of FK's. YUK^3.
- Conclusion: **We don't want to design many-to-many relationships in a database!**
Don't do it!
- We want to design a different, *better solution*...



A: The junction table

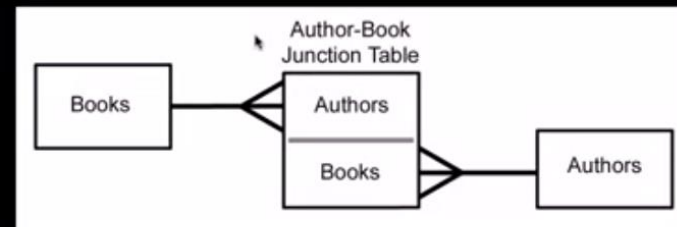
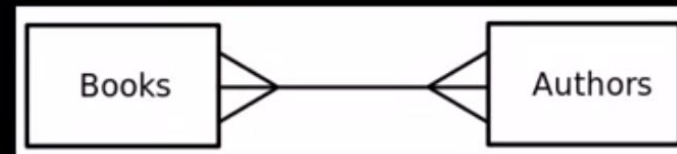
- One many-to-many relationship is the same as **two 1:many relationships!**
- The new table in the middle is a *bridge* table or *junction* table or *join* table or a *connection* table.



Another view

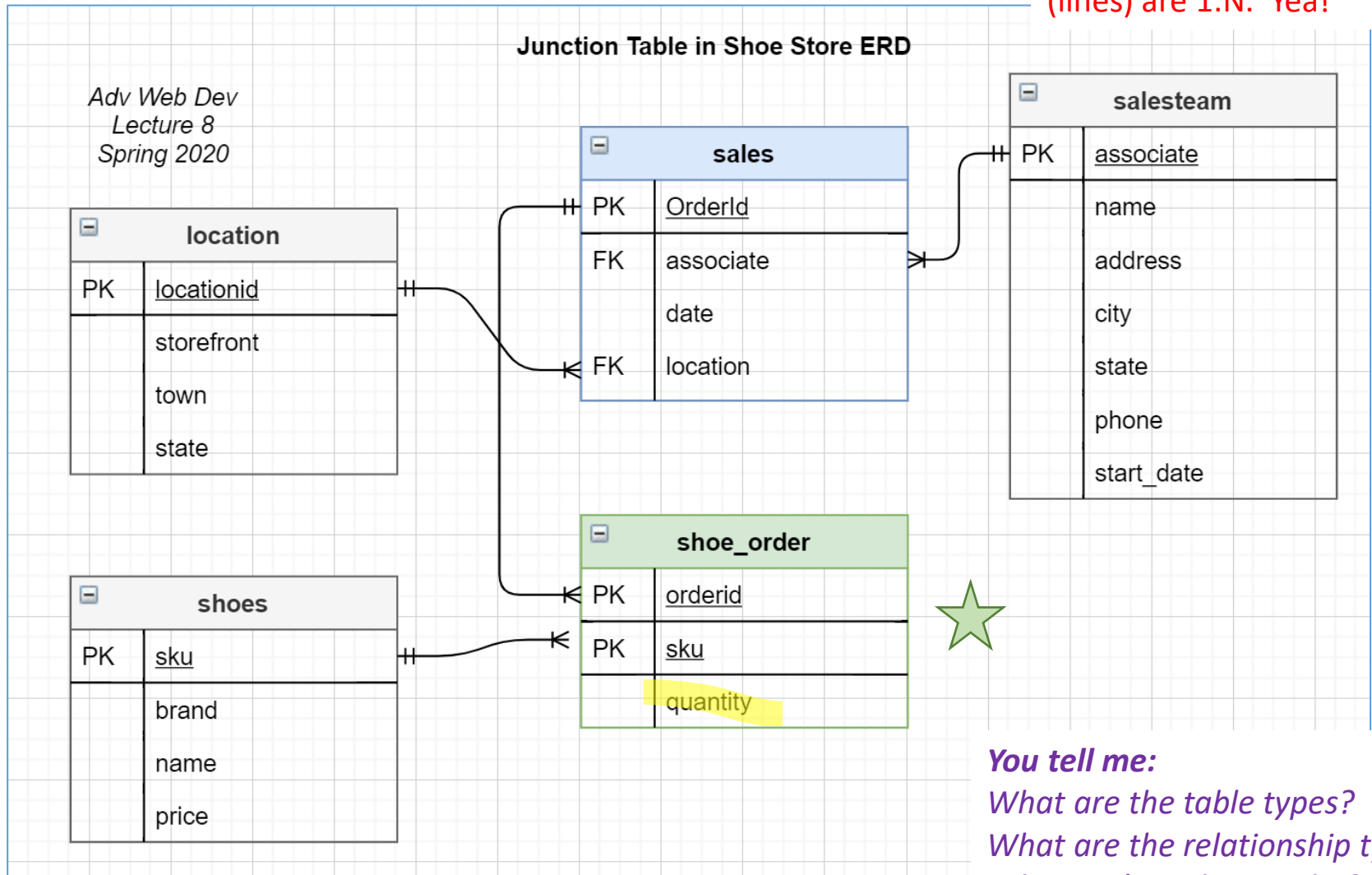
Many to Many

- Sometimes we need to model a relationship that is many to many.
- We need to add a “connection” table with two foreign keys.
- There is usually no separate primary key.



New ERD with our junction table

Notice: ALL relationships (lines) are 1:N. Yea!



You tell me:
 What are the table types?
 What are the relationship types?
 Why 2 PK's in shoe_order?

SQL code for junction table

- We now have 1 fact table and 1 junction table.
- The SALES tables "loses" SKU and QUANTITY columns. They go to the junction table.
- Why did SALES lose SKU?
- Why did SALES lose QUANTITY?
- The junction table has a compound primary key (line 56).
- It doesn't need it's own PK; it just needs two other PKs.
- If there are more than 1 PK, then a sales order (ordered) simply had multiple shoe types (SKU).

```
40 # ----- create fact table and junction tables -----
41
42 drop table sales;
43 create table sales (
44    orderid int unsigned not null unique,
45     associate int unsigned not null,
46     date date not null,
47     location int unsigned,
48     primary key (orderid));
49
50
51 drop table shoe_order;
52 create table shoe_order (
53    orderid int unsigned not null,
54     sku varchar(10) not null,
55     quantity int unsigned,
56     primary key (orderid,sku)
57 );
58
```

Inserting data into junction table

- Less cols into SALES fact table.
- Notice how *tall* the SHOE_ORDER table is.
- Some ORDERIDs are repeated, some are not. That's not a problem. ORDERID is a FK, not a PK.

```
80
87 # ---- insert order events into fact tables -----
88
89 insert into sales values
90     (3001, 5004, '2019-03-30', 101 ),
91     (3002, 5001, '2019-03-30', 102 ),
92     (3003, 5003, '2019-03-30', 103 ),
93     (3004, 5005, '2019-03-30', 102 ),
94     (3005, 5002, '2019-03-30', 102 ),
95     (3006, 5004, '2019-03-30', 101 ),
96     (3007, 5003, '2019-03-30', 103 ),
97     (3008, 5005, '2019-03-30', 102 ),
98     (3009, 5001, '2019-03-30', 102 );
99
100 select * from sales;
101
102 insert into shoe_order values
103     (3001, 'PU737SUR', 1 ),
104     (3002, 'AD073DUR', 2 ),
105     { (3003, 'PU737SUR', 1 ),
106       (3003, 'AS082NIM', 1 ),
107     } (3004, 'NI826QUE', 2 ),
108     (3005, 'AD073DUR', 2 ),
109     { (3006, 'AS082NIM', 1 ),
110       (3006, 'NI938AIR', 1 ),
111     } (3007, 'NI938AIR', 1 ),
112     (3007, 'NI826QUE', 1 ),
113     (3007, 'AD073DUR', 1 ),
```

2. SQL joining process

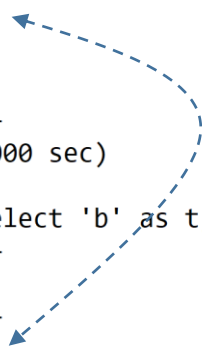
How does SQL do this?

The join template

- Add a JOIN statement *after* FROM and *before* WHERE

```
select <vars1>, <vars2>
from <table1>
join <table2>
where <conditions>
```

```
MariaDB [test]>
MariaDB [test]> select 'a' as t, a.* from a;
+-----+-----+
| t | id1 | att1 |
+-----+-----+
| a | 1 | foo |
| a | 2 | xxx |
| a | 3 | aaa |
+-----+-----+
3 rows in set (0.000 sec)
```



```
MariaDB [test]> select 'b' as t, b.* from b;
+-----+-----+
| t | id1 | att2 |
+-----+-----+
| b | 1 | bar |
| b | 2 | yyy |
| b | 3 | bbb |
+-----+-----+
3 rows in set (0.000 sec)
```

```
MariaDB [test]>
MariaDB [test]> select a.*, b.*
-> from a
-> join b
-> where a.id1 = b.id1 ;
+-----+-----+-----+-----+
| id1 | att1 | id1 | att2 |
+-----+-----+-----+-----+
| 1 | foo | 1 | bar |
| 2 | xxx | 2 | yyy |
| 3 | aaa | 3 | bbb |
+-----+-----+-----+-----+
3 rows in set (0.000 sec)
```

How does SQL join tables

- **Cross product**
- SQL simply joins every row of one table to every row of another table.
- i.e. every row of table a is "put next to" every row of table b
- ...even if the join doesn't make sense.
- You, the designer, ***must*** tell SQL if the joining of two rows makes sense with WHERE or ON clauses.

no WHERE

```

MariaDB [test]> select a.*, b.*
  -> from a
  -> join b;
+-----+-----+-----+-----+
| id1 | att1 | id1 | att2 |
+-----+-----+-----+-----+
|  1  | foo  |  1  | bar  | correct
|  2  | xxx  |  1  | bar  | wrong
|  3  | aaa  |  1  | bar  |
|  1  | foo  |  2  | yyy  |
|  2  | xxx  |  2  | yyy  |
|  3  | aaa  |  2  | yyy  |
|  1  | foo  |  3  | bbb  |
|  2  | xxx  |  3  | bbb  |
|  3  | aaa  |  3  | bbb  |
+-----+-----+-----+-----+
9 rows in set (0.000 sec)

```

3x3 - every combination of a.id1 with b.id1

Join where it makes sense

- There are two ways to tell SQL where it makes sense.
- 1. Use the WHERE clause
- The key (PK/FK) of one table must match the key (PK/FK) of another table.
- The keys are unique representations of a data entity.
- If they numbers, then this match is *really fast*.

A

```
MariaDB [test]>
MariaDB [test]> select a.*, b.*
  -> from a
  -> join b
  -> where a.id1 = b.id1 ;
```

id1	att1	id1	att2
1	foo	1	bar
2	xxx	2	yyy
3	aaa	3	bbb

```
3 rows in set (0.000 sec)
```


Join where it makes sense

- The prof prefers this syntax!
- 2. use an ON after the JOIN statement
- The *result* is identical to #1 using WHERE.
- The ON immediately follows the JOIN. To me, having the key relationship immediately after the JOIN makes a lot of sense.

```
MariaDB [test]> select a.*, b.*  
-> from a  
-> inner join b on a.id1 = b.id1  
-> ;  
  
+-----+-----+-----+-----+  
| id1 | att1 | id1 | att2 |  
+-----+-----+-----+-----+  
| 1 | foo | 1 | bar |  
| 2 | xxx | 2 | yyy |  
| 3 | aaa | 3 | bbb |  
+-----+-----+-----+-----+  
3 rows in set (0.000 sec)
```

B

Simple join for shoe store ex

2

```
MariaDB [test]> select * from shoes;
+-----+-----+-----+-----+
| sku      | brand | name  | price |
+-----+-----+-----+-----+
| AD073DUR | Adidas | Duramo | 45.99 |
| AS082NIM | ASICS  | Nimbus | 56.99 |
| NI826QUE | Nike   | Quest  | 78.99 |
| NI938AIR | Nike   | Air    | 67.99 |
| PU737SUR | Puma   | Surin  | 34.99 |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)
```

```
MariaDB [test]> select * from shoe_order;
+-----+-----+-----+
| orderid | sku      | quantity |
+-----+-----+-----+
| 3001    | PU737SUR | 1         |
| 3002    | AD073DUR | 2         |
| 3003    | AS082NIM | 1         |
| 3003    | PU737SUR | 1         |
| 3004    | NI826QUE | 2         |
| 3005    | AD073DUR | 2         |
| 3006    | AS082NIM | 1         |
| 3006    | NI938AIR | 1         |
| 3007    | AD073DUR | 1         |
| 3007    | NI826QUE | 1         |
| 3007    | NI938AIR | 1         |
| 3008    | AD073DUR | 1         |
| 3008    | PU737SUR | 1         |
| 3009    | NI938AIR | 2         |
+-----+-----+-----+
14 rows in set (0.000 sec)
```

1

The SQL query.
Boss: "Show me all the Nike sales!"

```
136
137 # ----- some other inner joins -----
138 select e.sku, c.brand, c.name as shoename,
139        c.price, e.quantity
140 from shoe_order e
141 inner join shoes c on c.sku=e.sku
142 where c.brand = "Nike"
143 ;
144
```

3

The results:

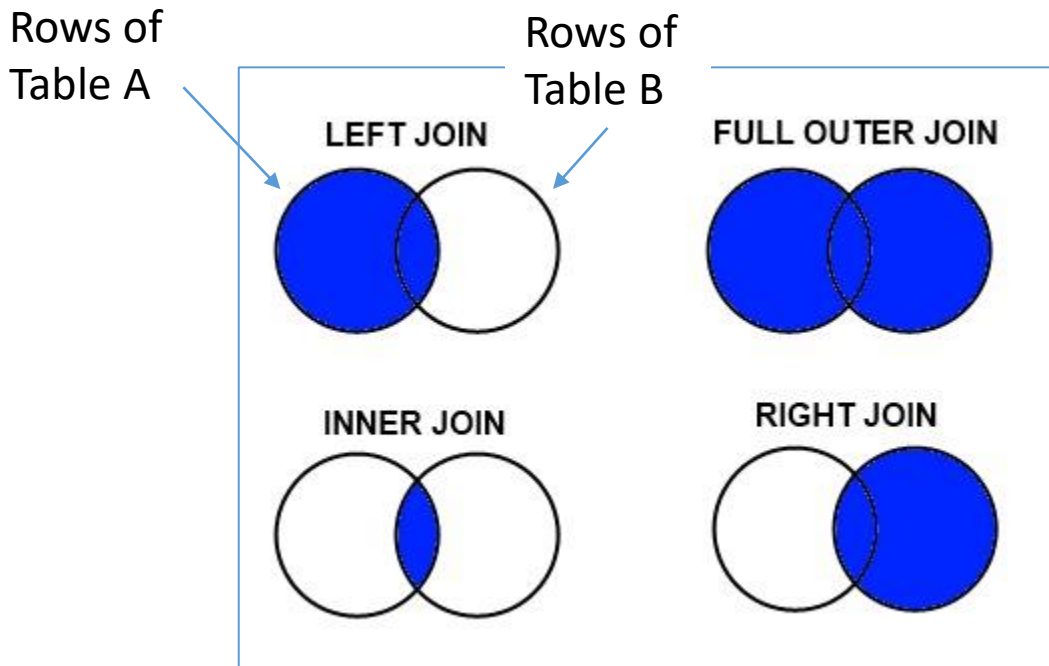
4

```
+-----+-----+-----+-----+-----+
| sku      | brand | shoename | price | quantity |
+-----+-----+-----+-----+-----+
| NI826QUE | Nike  | Quest    | 78.99 | 2         |
| NI938AIR | Nike  | Air      | 67.99 | 1         |
| NI826QUE | Nike  | Quest    | 78.99 | 1         |
| NI938AIR | Nike  | Air      | 67.99 | 1         |
| NI938AIR | Nike  | Air      | 67.99 | 2         |
+-----+-----+-----+-----+-----+
5 rows in set (0.000 sec)
```

3. Join Syntax

Why do we say Inner Join?

- There are several types of joins.
- **Inner Join** means only keep the rows where the keys match in both datasets.



*I think, in this course, we'll only cover **inner join**.*

Review of big picture

- Remember *Normalization*? The whole point of which was to remove redundant data.
 - How? We **split-up** data into individual entities. We called these *dimension* tables; each one has a Primary Key (PK).
 - Other tables can get to that *entity* data by storing just that 1 key – called a Foreign Key (FK). That's the link or relationship to get to the other data *dimension*.
- So, you see, this JOIN-ing is the *other side* of the Normalization splitting coin.
- The JOINS are how we **recreate** the original combined data.
- It's OK to have redundant data in the JOIN *result* (we just *don't* want that redundant data in the actual database)

Recombine SQL

```
122 # ----- recreate the flat dataset -----
123
124 select a.orderid, b.storefront,
125        d.name as sales_associate,
126        e.sku, c.brand, c.name as shoename, c.price, e.quantity
127 from sales a
128 inner join shoe_order e on a.orderid = e.orderid
129 inner join location b on a.location=b.locationid
130 inner join shoes c on c.sku=e.sku
131 inner join salesteam d on a.associate=d.associate_id
132 order by e.orderid, e.sku
133 ;
134
```

- Can you see the ON after the join?
- Note: the ON's can only "look up" to an already defined alias.
- Can you see the PK's?
- Can you pick out the dimension tables from the fact tables?

Recombine result

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| orderid | storefront | sales_associate | sku | brand | shoename | price | quantity |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3001 | Tafts Corners | Superman | PU737SUR | Puma | Surin | 34.99 | 1 |
| 3002 | Factory Outlet | The Hulk | AD073DUR | Adidas | Duramo | 45.99 | 2 |
| 3003 | Jericho Center | Spiderman | AS082NIM | ASICS | Nimbus | 56.99 | 1 |
| 3003 | Jericho Center | Spiderman | PU737SUR | Puma | Surin | 34.99 | 1 |
| 3004 | Factory Outlet | Bat Man | NI826QUE | Nike | Quest | 78.99 | 2 |
| 3005 | Factory Outlet | Wonder Woman | AD073DUR | Adidas | Duramo | 45.99 | 2 |
| 3006 | Tafts Corners | Superman | AS082NIM | ASICS | Nimbus | 56.99 | 1 |
| 3006 | Tafts Corners | Superman | NI938AIR | Nike | Air | 67.99 | 1 |
| 3007 | Jericho Center | Spiderman | AD073DUR | Adidas | Duramo | 45.99 | 1 |
| 3007 | Jericho Center | Spiderman | NI826QUE | Nike | Quest | 78.99 | 1 |
| 3007 | Jericho Center | Spiderman | NI938AIR | Nike | Air | 67.99 | 1 |
| 3008 | Factory Outlet | Bat Man | AD073DUR | Adidas | Duramo | 45.99 | 1 |
| 3008 | Factory Outlet | Bat Man | PU737SUR | Puma | Surin | 34.99 | 1 |
| 3009 | Factory Outlet | The Hulk | NI938AIR | Nike | Air | 67.99 | 2 |
+-----+-----+-----+-----+-----+-----+-----+-----+
14 rows in set (0.002 sec)
```

- Can you see the PK's?
- Can you see the redundant strings from the recombine?
- Can you see the many-to-many in the shoes per order?
- Can you put the columns in the proper entity (dimension or fact)?

4. Detailed JOIN examples

Row by row

Consider the schema

```
MariaDB [test]> select * from shoes;
```

sku	brand	name	price
AD073DUR	Adidas	Duramo	45.99
AS082NIM	ASICS	Nimbus	56.99
NI826QUE	Nike	Quest	78.99
NI938AIR	Nike	Air	67.99
PU737SUR	Puma	Surin	34.99

```
5 rows in set (0.000 sec)
```

```
MariaDB [test]> select * from sales;
```

orderid	associate	date	location
3001	5004	2019-03-30	101
3002	5001	2019-03-30	102
3003	5003	2019-03-30	103
3004	5005	2019-03-30	102
3005	5002	2019-03-30	102
3006	5004	2019-03-30	101
3007	5003	2019-03-30	103
3008	5005	2019-03-30	102
3009	5001	2019-03-30	102

```
9 rows in set (0.000 sec)
```

```
MariaDB [test]> select * from shoe_order;
```

orderid	sku	quantity
3001	PU737SUR	1
3002	AD073DUR	2
3003	AS082NIM	1
3003	PU737SUR	1
3004	NI826QUE	2
3005	AD073DUR	2
3006	AS082NIM	1
3006	NI938AIR	1
3007	AD073DUR	1
3007	NI826QUE	1

```
MariaDB [test]> select * from salesteam;
```

associate_id	name	address	city	state
5001	The Hulk	Wall St	NYC	NY
5002	Wonder Woman	1 ocean drive	Paradise Island	
5003	Spiderman	Park St	NYC	NY
5004	Superman	Main St	metropolis	
5005	Bat Man	1 Bat cave	gotham city	

```
5 rows in set (0.000 sec)
```

```
MariaDB [test]> select * from location;
```

locationid	storefront	town	state	phone
101	Taft's Corners	Williston	VT	802-288-6666
102	Factory Outlet	Essex	VT	802-555-7777
103	Jericho Center	Jericho	VT	802-899-5555

```
3 rows in set (0.000 sec)
```

Tell me about Order 3003

```
124 v select a.orderid, b.storefront,  
125         d.name as sales_associate,  
126         e.sku, c.brand, c.name as shoename, c.price, e.quantity  
127 from sales a  
128 inner join shoe_order e on a.orderid = e.orderid  
129 inner join location b on a.location=b.locationid  
130 inner join shoes c on c.sku=e.sku  
131 inner join salesteam d on a.associate=d.associate_id  
132 where a.orderid = 3003  
133 order by e.orderid, e.sku  
134 ;  
135
```

orderid	storefront	sales_associate	sku	brand	shoename	price	quantity
3003	Jericho Center	Spiderman	AS082NIM	ASICS	Nimbus	56.99	1
3003	Jericho Center	Spiderman	PU737SUR	Puma	Surin	34.99	1

2 rows in set (0.004 sec)

Tell me about Order 3003

Connect the arrows...

```
MariaDB [test]> select * from shoes;
```

sku	brand	name	price
AD073DUR	Adidas	Duramo	45.99
AS082NIM	ASICS	Nimbus	56.99
NI826QUE	Nike	Quest	78.99
NI938AIR	Nike	Air	67.99
PU737SUR	Puma	Surin	34.99

5 rows in set (0.000 sec)

```
MariaDB [test]> select * from sales;
```

orderid	associate	date	location
3001	5004	2019-03-30	101
3002	5001	2019-03-30	102
3003	5003	2019-03-30	103
3004	5005	2019-03-30	102
3005	5002	2019-03-30	102
3006	5004	2019-03-30	101
3007	5003	2019-03-30	103
3008	5005	2019-03-30	102
3009	5001	2019-03-30	102

9 rows in set (0.000 sec)

```
MariaDB [test]> select * from shoe_order;
```

orderid	sku	quantity
3001	PU737SUR	1
3002	AD073DUR	2
3003	AS082NIM	1
3003	PU737SUR	1
3004	NI826QUE	2
3005	AD073DUR	2
3006	AS082NIM	1
3006	NI938AIR	1
3007	AD073DUR	1
3007	NI826QUE	1

```
MariaDB [test]> select * from salesteam;
```

associate_id	name	address	city	state
5001	The Hulk	Wall St	NYC	NY
5002	Wonder Woman	1 ocean drive	Paradise Island	
5003	Spiderman	Park St	NYC	NY
5004	Superman	Main St	metropolis	
5005	Bat Man	1 Bat cave	gotham city	

5 rows in set (0.000 sec)

```
MariaDB [test]> select * from location;
```

locationid	storefront	town	state	phone
101	Taft's Corners	Williston	VT	802-288-6666
102	Factory Outlet	Essex	VT	802-555-7777
103	Jericho Center	Jericho	VT	802-899-5555

3 rows in set (0.000 sec)

What has Superman sold?

```
124 v select a.orderid, b.storefront, a.date,  
125         d.name as sales_associate,  
126         e.sku, c.brand, c.name as shoename, c.price, e.quantity  
127 from sales a  
128 inner join shoe_order e on a.orderid = e.orderid  
129 inner join location b on a.location=b.locationid  
130 inner join shoes c on c.sku=e.sku  
131 inner join salesteam d on a.associate=d.associate_id  
132 where d.name = 'Superman'  
133 order by e.orderid, e.sku  
134 ;
```

orderid	storefront	date	sales_associate	sku	brand	shoename	price	quantity
3001	Taft's Corners	2019-03-30	Superman	PU737SUR	Puma	Surin	34.99	1
3006	Taft's Corners	2019-03-30	Superman	AS082NIM	ASICS	Nimbus	56.99	1
3006	Taft's Corners	2019-03-30	Superman	NI938AIR	Nike	Air	67.99	1

3 rows in set (0.004 sec)

What has Superman sold?

Connect the arrows...

```
MariaDB [test]> select * from shoes;
```

sku	brand	name	price
AD073DUR	Adidas	Duramo	45.99
AS082NIM	ASICS	Nimbus	56.99
NI826QUE	Nike	Quest	78.99
NI938AIR	Nike	Air	67.99
PU737SUR	Puma	Surin	34.99

5 rows in set (0.000 sec)

```
MariaDB [test]> select * from sales;
```

orderid	associate	date	location
3001	5004	2019-03-30	101
3002	5001	2019-03-30	102
3003	5003	2019-03-30	103
3004	5005	2019-03-30	102
3005	5002	2019-03-30	102
3006	5004	2019-03-30	101
3007	5003	2019-03-30	103
3008	5005	2019-03-30	102
3009	5001	2019-03-30	102

9 rows in set (0.000 sec)

```
MariaDB [test]> select * from shoe_order;
```

orderid	sku	quantity
3001	PU737SUR	1
3002	AD073DUR	2
3003	AS082NIM	1
3003	PU737SUR	1
3004	NI826QUE	2
3005	AD073DUR	2
3006	AS082NIM	1
3006	NI938AIR	1
3007	AD073DUR	1
3007	NI826QUE	1

```
MariaDB [test]> select * from salesteam;
```

associate_id	name	address	city	state
5001	The Hulk	Wall St	NYC	NY
5002	Wonder Woman	1 ocean drive	Paradise Island	NY
5003	Spiderman	Park St	NYC	NY
5004	Superman	Main St	metropolis	
5005	Bat Man	1 Bat cave	gotham city	

5 rows in set (0.000 sec)

```
MariaDB [test]> select * from location;
```

locationid	storefront	town	state	phone
101	Taft's Corners	Williston	VT	802-288-6666
102	Factory Outlet	Essex	VT	802-555-7777
103	Jericho Center	Jericho	VT	802-899-5555

3 rows in set (0.000 sec)