CSE 344

MARCH 30TH – INTRO TO JOINS

ADMINISTRATIVE MINUTIAE

Online Quizzes

- First quiz out
- Due next Friday (11:00 pm)

Coding HW

- Due next Wednesday (11:30 pm)
- HW2 out next Wednesday
- Office hours
 - Listed on course website



- What operations should we expect SQLite (or any DBMS) to support just on what we know right now?
 - create table
 - insert into
 - select
 - delete from

• What sorts of inputs do these functions need to have?

- create table: table name, schema
- insert into: table name, tuple
- select: table name, attributes
- delete from: table name, condition

DEMO 1

- Common Syntax
 - CREATE TABLE [tablename] ([att1] [type1], [att2] [type2]...);
 - INSERT INTO [tablename] VALUES ([val1],[val2]...);
 - SELECT [att1],[att2],... FROM [tablename] WHERE [condition]
 - DELETE FROM [tablename]
 WHERE [condition]



- Two other operations we want to support
 - ALTER TABLE: Adds a new attribute to the table
 - UPDATE: Change the attribute for a particular tuple in the table.
- Common Syntax
 - ALTER TABLE [tablename] ADD [attname] [atttype]
 - UPDATE [tablename] SET [attname]=[value]

- Two other operations we want to support
 - ALTER TABLE: Adds a new attribute to the table
 - UPDATE: Change the attribute for a particular tuple in the table.
- Common Syntax
 - ALTER TABLE [tablename] ADD [attname] [atttype]
 - UPDATE [tablename] SET [attname]=[value]
 WHERE [condition]



Tables are NOT ordered

• they are sets or multisets (bags)

Tables are FLAT

• No nested attributes

Tables DO NOT prescribe how they are implemented / stored on disk

• This is called **physical data independence**



• Tables may not be ordered, but data can be returned in an order with the ORDER BY modifier

- Tables may not be ordered, but data can be returned in an order with the ORDER BY modifier
- Whew, today's been a lot of coding... I know what you're thinking...



• We can think of accessing information through queries as some combination of functions

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):
 - How would we need to get the birth year of all UWBW students from California?

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):
 - How would we need to get the birth year of all UWBW students from California?
 - Think of the file as a set of tuples

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):
 - How would we need to get the birth year of all UWBW students from California?
 - Think of the file as a set of tuples
 - Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):
 - How would we need to get the birth year of all UWBW students from California?
 - Think of the file as a set of tuples
 - Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set
 - What does this return?

- We can think of accessing information through queries as some combination of functions
 - Consider a table of UW students (with all relevant info):
 - How would we need to get the birth year of all UWBW students from California?
 - Think of the file as a set of tuples
 - Find the set of UWBW students and the set of students from California; Find the intersection of these sets, return just the year from the birthday values of this set
 - What does this return?
 - Years, but with many duplicates. Even though sets don't allow duplicates, the objects are unique.

- If we only want to return unique elements, we can use the DISTINCT modifier
 - Even if we hide some attributes from the output, the data is all still there.
 - When we select a subset of the attributes, this function is called a *projection*

- This was all for a single table.
- Data models specify how our data are stored and how the data are related
- Need to utilize these relations, or the database was pointless
- This involves a JOIN

JOIN: INTRO

- The JOIN is the way we indicate in a query how multiple tables are related.
 - Example, if we want all of the products and their relevant company information, we need to *join* those two tables.
 - The result of the join is all of the relevant information from both tables
 - Join occurs based on the join condition.
 - This allows us to access information that comes from multiple tables

JOINS IN SQL

pname	price	category	et Canon ography Canon		cname	country
MultiTouch	199.99	gadget			GizmoWorks	USA
SingleTouch	49.99	photography			Canon	Japan
Gizom	50	gadget			Hitachi	Japan
SuperGizmo	250.00	gadget	GizmoWorks			I

Retrieve all Japanese products that cost < \$150

JOINS IN SQL

pname	price	category	Canon		cname	country		
MultiTouch	199.99	gadget			Canon		GizmoWorks	USA
SingleTouch	49.99	photography			Canon	Japan		
Gizom	50	gadget			Hitachi	Japan		
SuperGizmo	250.00	gadget	GizmoWorks			I		

Retrieve all Japanese products that cost < \$150

```
SELECT pname, price
FROM Product, Company
WHERE ...
```

JOINS IN SQL

pname	price	category	manufacturerCanonCanonGizmoWorks		cname	country		
MultiTouch	199.99	gadget			Canon		GizmoWorks	USA
SingleTouch	49.99	photography			Canon	Japan		
Gizom	50	gadget			Hitachi	Japan		
SuperGizmo	250.00	gadget	GizmoWorks			· · · · · · · · · · · · · · · · · · ·		

Retrieve all Japanese products that cost < \$150

SELECT pname, price
FROM Product, Company
WHERE manufacturer=cname AND
 country='Japan' AND price < 150</pre>

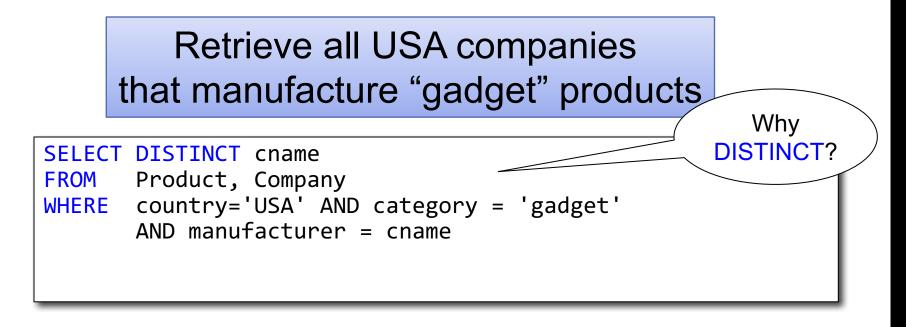
JOINS IN SQL

pname	price	category	manufacturer		cname	country
MultiTouch	199.99	gadget	Canon	, 1	GizmoWorks	USA
SingleTouch	49.99	photography	Canon		Canon	Japan
Gizom	50	gadget	GizmoWorks		Hitachi	Japan
SuperGizmo	250.00	gadget	GizmoWorks	L		

Retrieve all USA companies that manufacture "gadget" products

JOINS IN SQL

pname	price	category	manufacturer		cname	country
MultiTouch	199.99	gadget	Canon		GizmoWorks	USA
SingleTouch	49.99	photography	Canon		Canon	Japan
Gizom	50	gadget	GizmoWorks	-	Hitachi	Japan
SuperGizmo	250.00	gadget	GizmoWorks	L		•



JOINS IN SQL

The standard join in SQL is called an inner join

Each row in the result must come from both tables in the join

Sometimes we want to include rows from only one of the two table: outer join

Employee

id	name		employeeID	productID					
1	Joe		1	344					
2	Jack		1	355					
3	Jill		2	544					

Retrieve employees and their sales

Employee

id	name		employeeID	productID				
1	Joe		1	344				
2	Jack		1	355				
3	Jill		2	544				

Retrieve employees and their sales

SELECT *
FROM Employee E, Sales S
WHERE E.id = S.employeeID

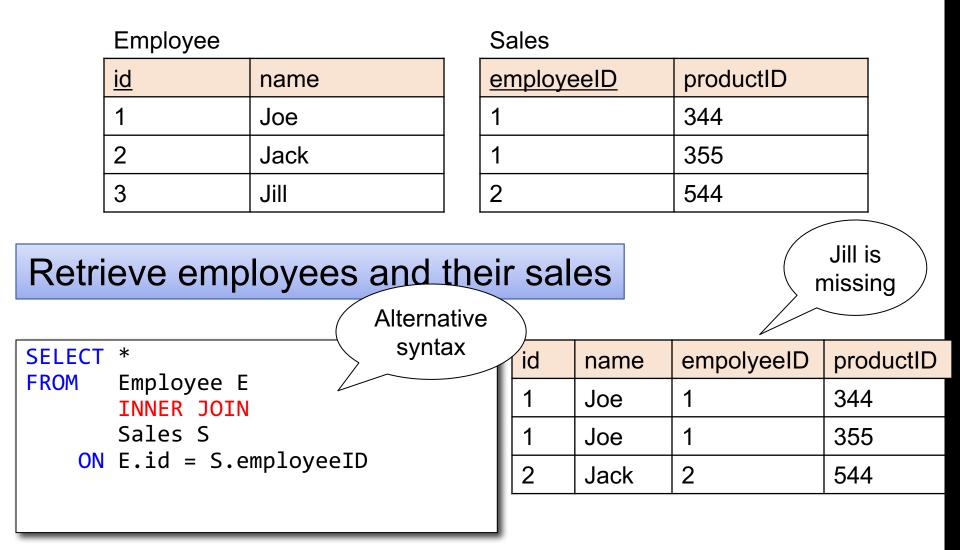
Employee

I J						
<u>id</u>	name		employeeID	productID		
1	Joe		1	344		
2	Jack		1	355		
3	Jill		2	544		

Retrieve employees and their sales

<pre>SELECT * FROM Employee E, Sales S WHERE E.id = S.employeeID</pre>	id	name	empolyeeID	productID
	1	Joe	1	344
	1	Joe	1	355
	2	Jack	2	544

	Employee			Sa	ales			
	<u>id</u>	name		<u>er</u>	nploye	<u>eeID</u>	productID	
	1	Joe		1			344	
	2	Jack		1			355	
	3	Jill	Γ	2			544	
Retrieve employees and their sales								
<pre>SELECT * FROM Employee E, Sales S WHERE E.id = S.employeeID</pre>				1	id	name	empolyeeID	productID
			I	1	Joe	1	344	
			I	1	Joe	1	355	
			_	1	2	Jack	2	544



OUTER JOIN

	Employee id name		1	Sales				
				e	employeeID		productID	
	1	Joe		1			344	
	2	Jack		1			355	
	3	Jill		2	2		544	
Retrieve employees and their sales								
SELECT * FROM Employee E LEFT OUTER JOIN Sales S					id	name	empolyeeID	productID
					1	Joe	1	344
					1	Joe	1	355
ON E.id = S.employeeID				2	Jack	2	544	
			3	Jill	NULL	NULL		

(INNER) JOINS

Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = cname
```

(INI	NER) JOINS
FROM	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

Product

pname	category	manufacturer		
Gizmo	gadget	GizmoWorks		
Camera	Photo	Hitachi		
OneClick	Photo	Hitachi		

Company

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INI	NER) JOINS
FROM	<pre>DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname</pre>

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

<u>(IN</u>	NER) JOINS
FROM	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

pname	category	manufacturer	
Gizmo	gadget	GizmoWorks	
Camera	Photo	Hitachi	 Ī
OneClick	Photo	Hitachi	

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INI	NER) JOINS
FROM	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

pname	category	manufacturer		cname	country
Gizmo	gadget	GizmoWorks		GizmoWorks	USA
Camera	Photo	Hitachi	. .	Canon	Japan
OneClick	Photo	Hitachi		Hitachi	Japan

pname	category	manufacturer	cname	country
Gizmo	gadget	GizmoWorks	GizmoWorks	USA

(IN	NER) JOINS
FROM	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

pna	me	category	manufacturer
Gizr	no	gadget	GizmoWorks
Cam	era	Photo	Hitachi
OneC	Click	Photo	Hitachi

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INN	NER) JOINS
FROM WHERE	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
Camera	Photo	Hitachi
OneClick	Photo	Hitachi

cname	country
GizmoWorks	USA
Canon	Japan
Hitachi	Japan

(INI	NER) JOINS
FROM	DISTINCT cname Product, Company country='USA' AND category = 'gadget' AND manufacturer = cname

```
SELECT DISTINCT cname
FROM Product JOIN Company ON
country = 'USA' AND category = 'gadget'
AND manufacturer = cname
```

(INNER) JOINS

SELECT x1.a1, x2.a2, ... xm.am FROM R1 as x1, R2 as x2, ... Rm as xm WHERE Cond

This is called nested loop semantics since we are interpreting what a join means using a nested loop

Product(pname, price, category, manufacturer)
Company(cname, country)

-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both 'gadget' and 'photography' categories

Product(pname, price, category, manufacturer)
Company(cname, country)

-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both 'gadget' and 'photography' categories

```
SELECT DISTINCT z.cname
FROM Product x, Company z
WHERE z.country = 'USA'
AND x.manufacturer = z.cname
AND x.category = 'gadget'
AND x.category = 'photography;
```



Product(pname, price, category, manufacturer) Company(cname, country)

-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both 'gadget' and 'photography' categories

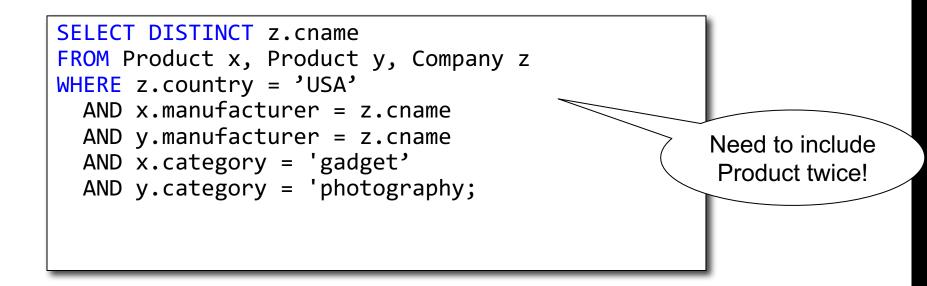
```
SELECT DISTINCT z.cname
FROM Product x, Company z
                                                      What about
WHERE z.country = 'USA'
  AND x.manufacturer = z.cname
 AND (x.category = 'gadget'
       OR x.category = 'photography);
```

this?

Product(pname, price, category, manufacturer)
Company(cname, country)

-- manufacturer is foreign key to Company

Retrieve all USA companies that manufacture products in both 'gadget' and 'photography' categories



SELF-JOINS AND TUPLE VARIABLES

Find USA companies that manufacture both products in the 'gadgets' and 'photo' category

Joining Product with Company is insufficient: need to join Product, with Product, and with Company

When a relation occurs twice in the FROM clause we call it a self-join; in that case we must use tuple variables (why?)

SELECT	DISTINCT z.cname
FROM	Product x, Product y, Company z
WHERE	z.country = 'USA'
	AND x.category = 'gadget'
	AND y.category = 'photo'
	AND x.manufacturer = z.cname
	AND y.manufacturer = z.cname;

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
SingleTouch	photo	Hitachi
MultiTouch	Photo	GizmoWorks

cname	country
GizmoWorks	USA
Hitachi	Japan

SEI E. IOINS

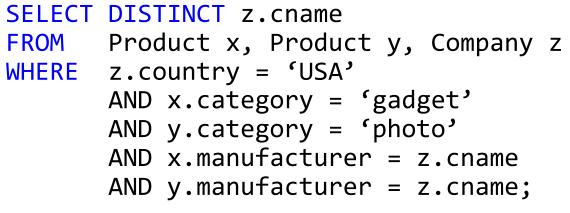
SELECT	DISTINCT z.cname
FROM	Product x, Product y, Company z
WHERE	z.country = 'USA'
	AND x.category = 'gadget'
	AND y.category = 'photo'
	AND x.manufacturer = z.cname
	AND y.manufacturer = z.cname;

Product

pname	category	manufacturer
Gizmo	gadget	GizmoWorks
SingleTouch	photo	Hitachi
MultiTouch	Photo	GizmoWorks

cname	country
GizmoWorks	USA
Hitachi	Japan

SELE INING

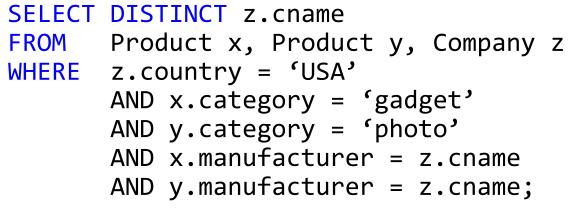


Product

x	pname	category	manufacturer
у	Gizmo	gadget	GizmoWorks
	SingleTouch	photo	Hitachi
	MultiTouch	Photo	GizmoWorks

cname	country
GizmoWorks	USA
Hitachi	Japan

SELE INING

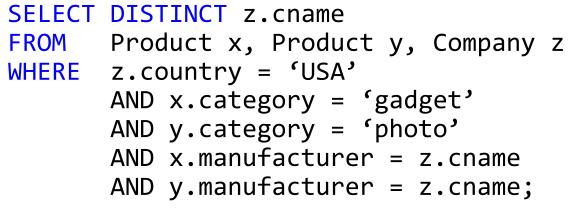


Product

x	pname	category	manufacturer	
у	Gizmo	gadget	GizmoWorks	
	SingleTouch	photo	Hitachi	
	MultiTouch	Photo	GizmoWorks	

Company

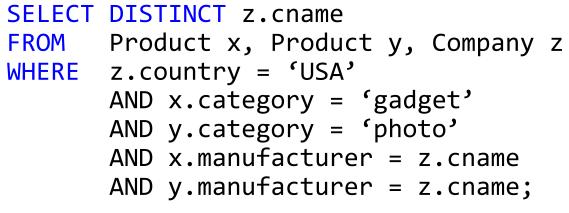
cname	country		
GizmoWorks	USA		
Hitachi	Japan		



Product

X	pname	category	manufacturer	
у	Gizmo	gadget	GizmoWorks	
	SingleTouch	photo	Hitachi	
	MultiTouch	Photo	GizmoWorks	

cname	country		
GizmoWorks	USA		
Hitachi	Japan		



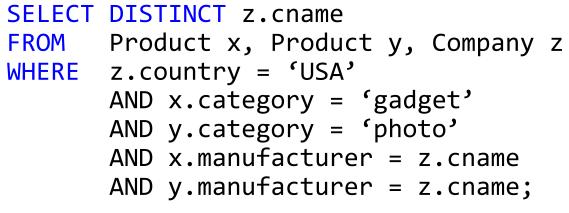
Product

Χ

У

pnamecategorymanufacturerGizmogadgetGizmoWorksSingleTouchphotoHitachiMultiTouchPhotoGizmoWorks

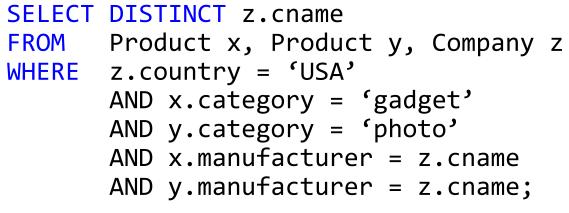
cnamecountryGizmoWorksUSAHitachiJapan



Product

x	pname	category	manufacturer
	Gizmo	gadget	GizmoWorks
у	SingleTouch	photo	Hitachi
	MultiTouch	Photo	GizmoWorks

cname	country	
GizmoWorks	USA	
Hitachi	Japan	



Product

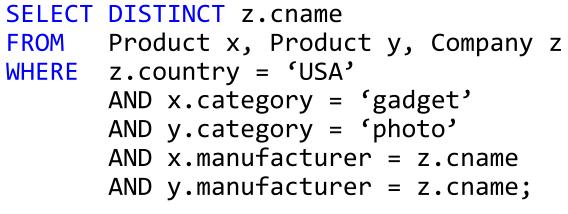
Χ

У

Company

pname	category	manufacturer	
Gizmo	gadget	GizmoWorks	
SingleTouch	photo	Hitachi	
MultiTouch	Photo	GizmoWorks	

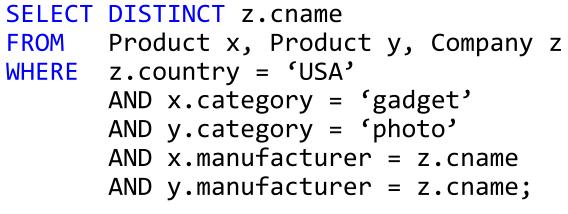
cname	country		
GizmoWorks	USA		
Hitachi	Japan		



Company

×	pname	category	manufacturer	cname	country	
	Gizmo	gadget	GizmoWorks	GizmoWorks	USA	
	SingleTouch	photo	Hitachi	Hitachi	Japan	
У	MultiTouch	Photo	GizmoWorks			•

x.pname	x.category	x.manufacturer	y.pname	y.category	y.manufacturer	z.cname	z.country
Gizmo	gadget	GizmoWorks	MultiTouch	Photo	GizmoWorks	GizmoWorks	USA



Company

x	pname	categ	ory	manu	facturer		cname	cou	untry	
	Gizmo	gadg	et	Gizm	oWorks		GizmoWorks	U	SA	
	SingleTouc	h phot	photo		Hitachi		Hitachi	Ja	pan	Ť
У	MultiTouch	MultiTouch Photo		Gizm	oWorks	E				-
x.pnam	e x.category	x.manufacturer	y.pna	ame	y.category		y.manufacturer	z.cname	Z.C	ountry

x.pname	x.category	x.manufacturer	y.pname	y.category	y.manufacturer	z.cname	z.c	country
Gizmo	gadget	GizmoWorks	MultiTouch	Photo	GizmoWorks	GizmoWorks	l	USA



Product(<u>name</u>, category)
Purchase(prodName, store)

-- prodName is foreign key

SELECT Product.name, Purchase.store
FROM Product, Purchase
WHERE Product.name = Purchase.prodName

We want to include products that are never sold, but some are not listed! Why?



Product(<u>name</u>, category)
Purchase(prodName, store)

-- prodName is foreign key

SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
Product.name = Purchase.prodName

SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON
Product.name = Purchase.prodName

Product

Name	Category		
Gizmo	gadget		
Camera	Photo		
OneClick	Photo		

Purchase

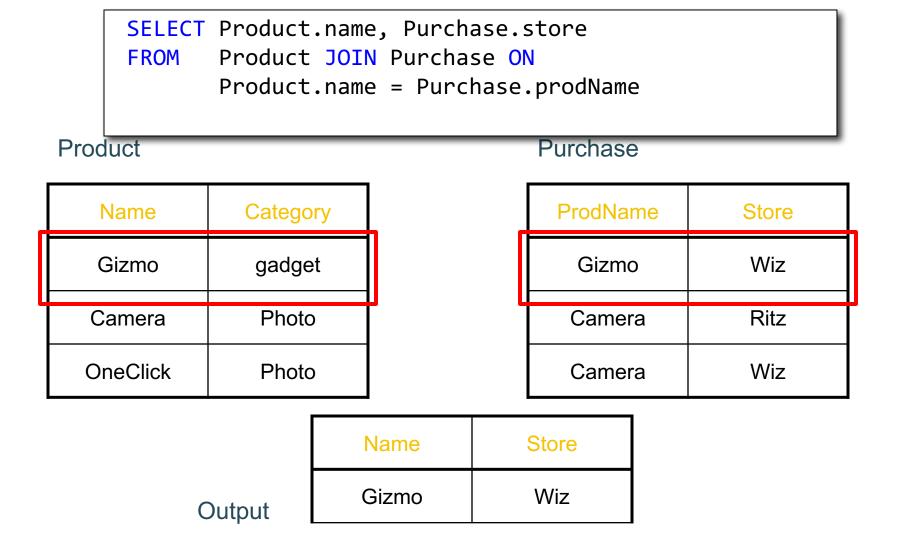
ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

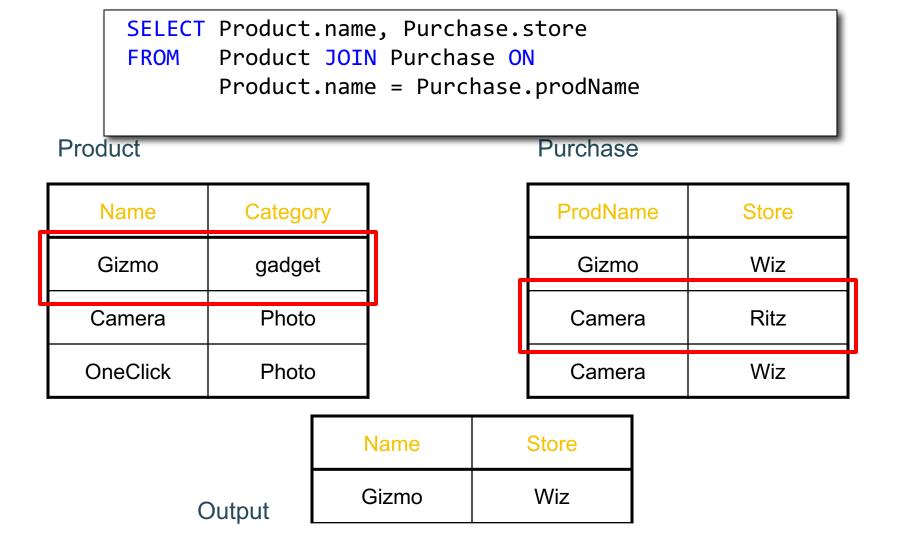
SELECT	Product.name, Purchase.store
FROM	Product JOIN Purchase ON
	<pre>Product.name = Purchase.prodName</pre>

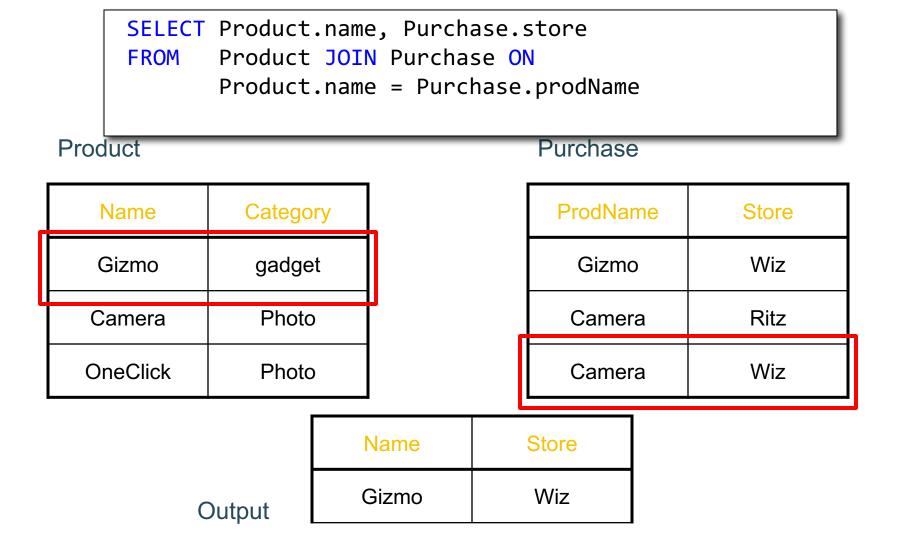
Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

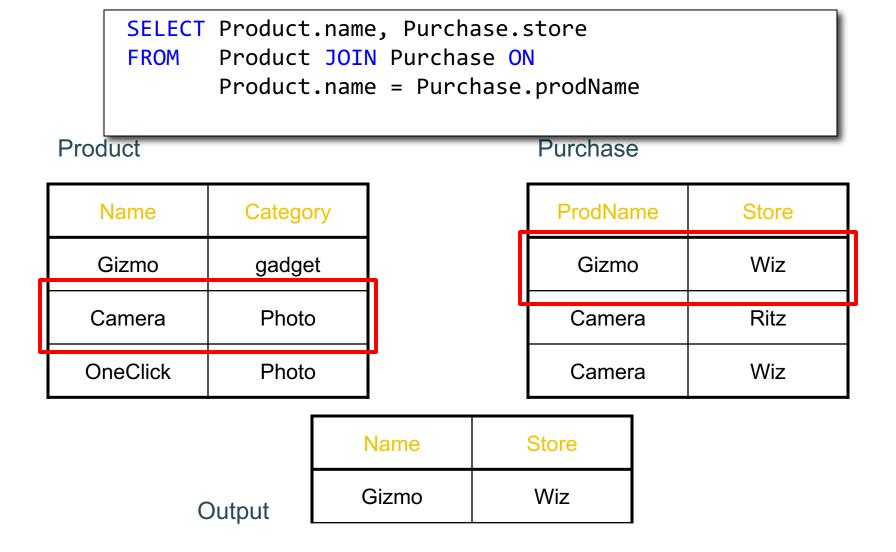
Purchase

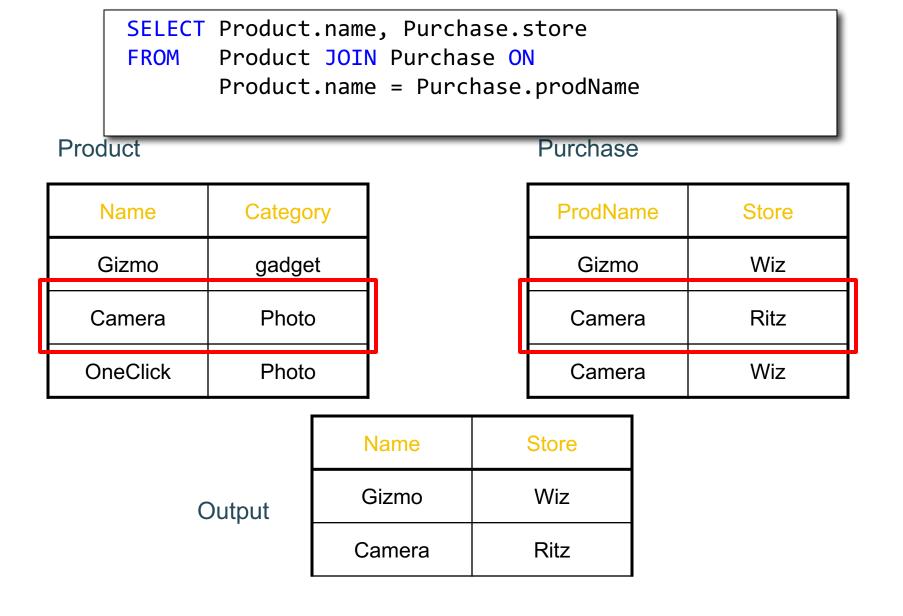
ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

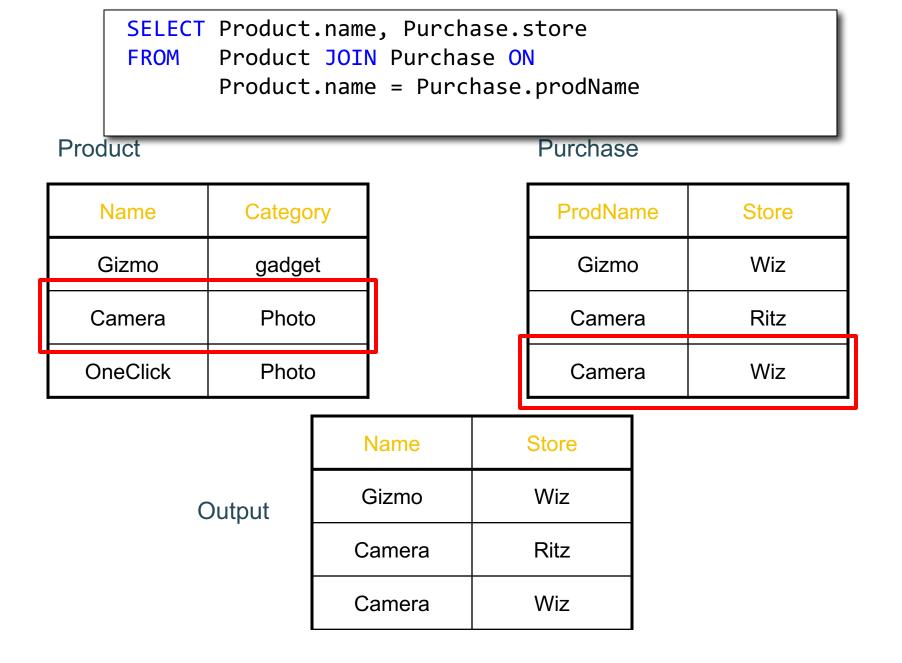


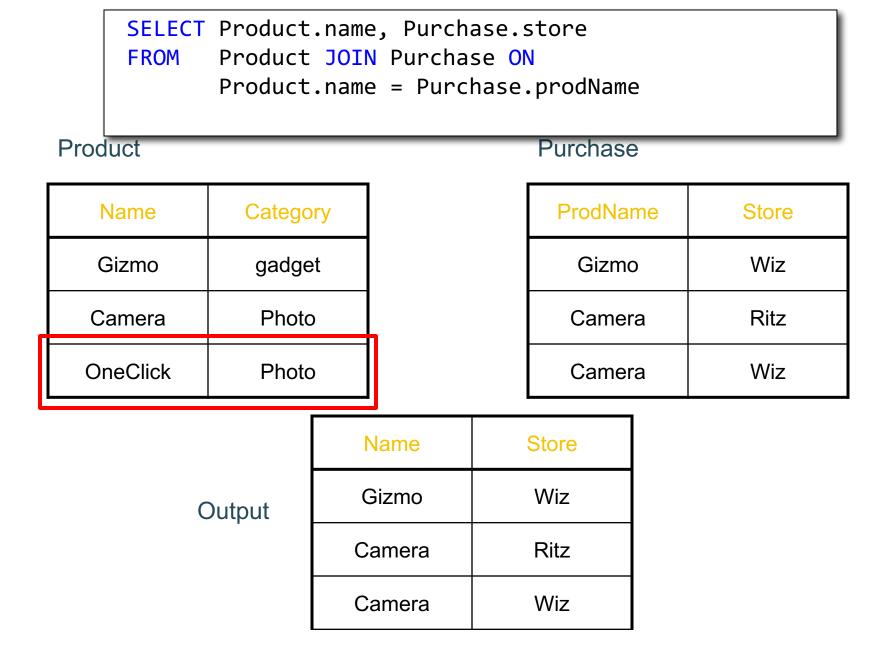


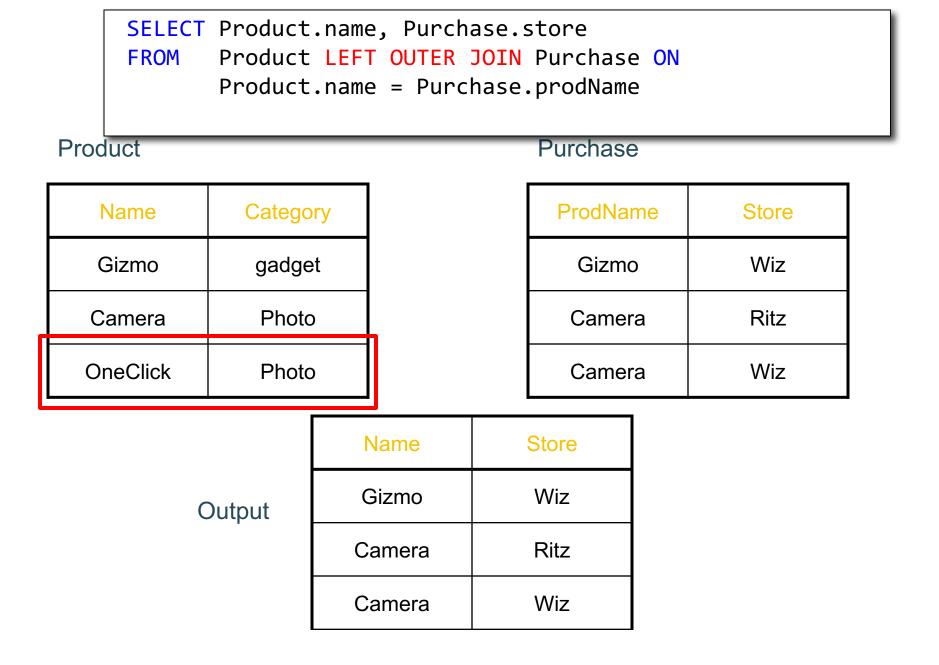


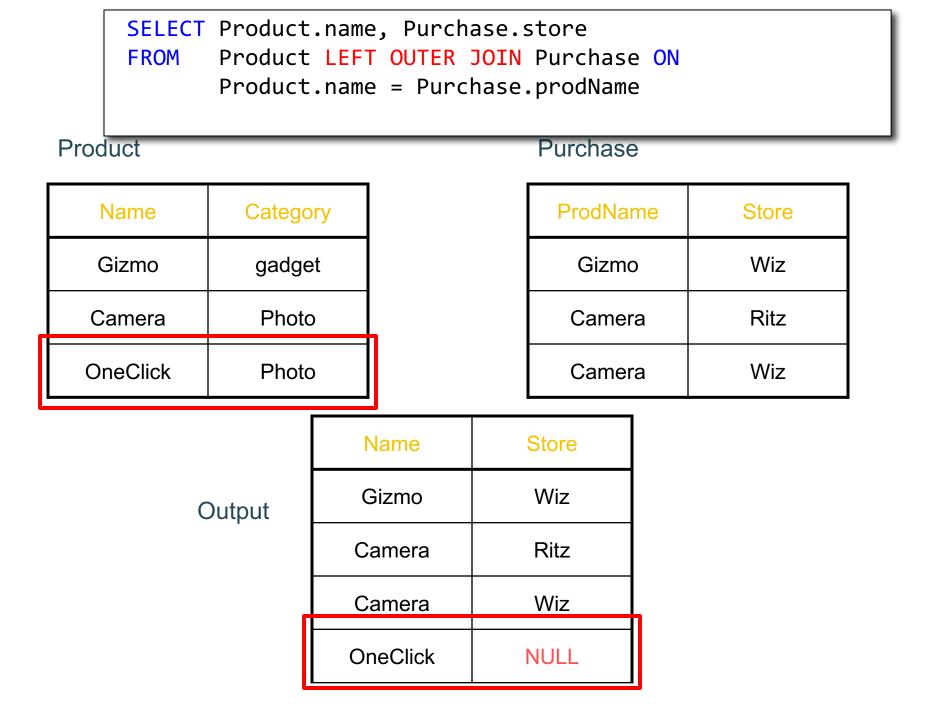












SELECT Product.name, Purchase.store FROM Product FULL OUTER JOIN Purchase ON Product.name = Purchase.prodName									
Product Purchase									
Name	Category		Γ	ProdName	Store				
Gizmo	gadget			Gizmo	Wiz				
Camera	Photo	_		Camera	Ritz				
OneClick	Photo			Camera	Wiz				
	Name	Store		Phone	Foo				
	Gizmo	Wiz		· · · · · · · · · · · · · · · · · · ·					
Output	Camera	Ritz							
	Camera	Wiz							
r	OneClick	NULL							
	NULL	Foo							

OUTER JOINS

tableA (LEFT/RIGHT/FULL) OUTER JOIN tableB ON p

Left outer join:

Include tuples from tableA even if no match

Right outer join:

- Include tuples from tableB even if no match
 Full outer join:
 - Include tuples from both even if no match

In all cases:

Patch tuples without matches using NULL