

# Introduction to Data Management CSE 344

## Lectures 4: Aggregates in SQL

CSE 344 - Winter 2016

1

## Announcements

- Did you remember the web quiz yesterday?
- HW 1 is due Tuesday (tomorrow), 11pm
- Next web quiz is out – due next Sunday
- HW2 coming out on Wednesday

CSE 344 - Winter 2016

2

## Outline

- Inner joins (6.2, review)
- Outer joins (6.3.8)
- Aggregations (6.4.3 – 6.4.6)
- Examples, examples, examples...

CSE 344 - Winter 2016

3

## (Inner) Joins

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

```
for x1 in R1:  
  for x2 in R2:  
    ...  
    for xm in Rm:  
      if Cond(x1, x2...):  
        output(x1.a1, x2.a2, ... xm.am)
```

Nested loop  
semantics

CSE 344 - Winter 2016

4

## (Inner) Joins

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

```
for x1 in R1:  
  for x2 in R2:  
    ...  
    for xm in Rm:  
      if Cond(x1, x2...):  
        output(x1.a1, x2.a2, ... xm.am)
```

Nested loop  
semantics

CSE 344 - Winter 2016

5

## (Inner) joins

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

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

CSE 344 - Winter 2016

6

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

CSE 344 - Winter 2016

7

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

CSE 344 - Winter 2016

8

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

CSE 344 - Winter 2016

9

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

pname	category	manufacturer	cname	country
Gizmo	gadget	GizmoWorks	GizmoWorks	USA

10

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

CSE 344 - Winter 2016

11

### (Inner) joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE 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

CSE 344 - Winter 2016

12

## (Inner) joins

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

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

CSE 344 - Winter 2016

13

## Self-Joins and Tuple Variables

- Find all companies that manufacture both products in the 'gadgets' category and in the 'photo' category
- Just joining Product with Company is insufficient: instead need to join Product, with Product, 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?)

CSE 344 - Winter 2016

14

## Self-joins

```
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 = cname
      AND y.manufacturer = cname;
```

Product

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

Company

cname	country
GizmoWorks	USA
Hitachi	Japan

## Self-joins

```
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 = cname
      AND y.manufacturer = cname;
```

Product

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

Company

cname	country
GizmoWorks	USA
Hitachi	Japan

## Self-joins

```
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 = cname
      AND y.manufacturer = cname;
```

Product

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

Company

cname	country
GizmoWorks	USA
Hitachi	Japan

## Self-joins

```
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 = cname
      AND y.manufacturer = cname;
```

Product

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

Company

cname	country
GizmoWorks	USA
Hitachi	Japan

## Self-joins

```
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 = cname
AND y.manufacturer = cname;
```

Product			Company		z
x	pname	category	manufacturer	cname	
y	Gizmo	gadget	GizmoWorks	GizmoWorks	USA
	SingleTouch	photo	Hitachi	Hitachi	Japan
	MultiTouch	Photo	GizmoWorks		

## Self-joins

```
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 = cname
AND y.manufacturer = cname;
```

Product			Company		z
x	pname	category	manufacturer	cname	
y	Gizmo	gadget	GizmoWorks	GizmoWorks	USA
	SingleTouch	photo	Hitachi	Hitachi	Japan
	MultiTouch	Photo	GizmoWorks		

## Self-joins

```
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 = cname
AND y.manufacturer = cname;
```

Product			Company		z
x	pname	category	manufacturer	cname	
y	Gizmo	gadget	GizmoWorks	GizmoWorks	USA
	SingleTouch	photo	Hitachi	Hitachi	Japan
	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

## Self-joins

```
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 = cname
AND y.manufacturer = cname;
```

Product			Company		z
x	pname	category	manufacturer	cname	
y	Gizmo	gadget	GizmoWorks	GizmoWorks	USA
	SingleTouch	photo	Hitachi	Hitachi	Japan
	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

## Outer joins

Product(name, category)  
Purchase(prodName, store) -- prodName is foreign key

An "inner join":  

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

Same as:

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

But some Products are not listed! Why?

23

## Outer joins

Product(name, category)  
Purchase(prodName, store) -- prodName is foreign key

If we want to include products that never sold,  
then we need an "outerjoin":

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

CSE 344 - Winter 2016

24

```
SELECT Product.name, Purchase.store
FROM   Product LEFT OUTER 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

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

CSE 344 - Winter 2016

25

## Outer Joins

- Left outer join:
  - Include the left tuple even if there's no match
- Right outer join:
  - Include the right tuple even if there's no match
- Full outer join:
  - Include both left and right tuples even if there's no match

CSE 344 - Winter 2016

26

## Aggregation in SQL

```
>sqlite3 lecture04
```

```
sqlite> create table Purchase
(pid int primary key,
 product text,
 price float,
 quantity int,
 month varchar(15));
```

```
sqlite> -- download data.txt
sqlite> .import lec04-data.txt Purchase
```

Specify a filename  
where the database  
will be stored

Other DBMSs have  
other ways of  
importing data

CSE 344 - Winter 2016

27

## Comment about SQLite

- One cannot load NULL values such that they are actually loaded as null values
- So we need to use two steps:
  - Load null values using some type of special value
  - Update the special values to actual null values

```
update Purchase
set price = null
where price = 'null'
```

CSE 344 - Winter 2016

28

## Simple Aggregations

Five basic aggregate operations in SQL

```
select count(*) from Purchase
select sum(quantity) from Purchase
select avg(price) from Purchase
select max(quantity) from Purchase
select min(quantity) from Purchase
```

Except count, all aggregations apply to a single attribute

CSE 344 - Winter 2016

29

## Aggregates and NULL Values

Null values are not used in aggregates

```
insert into Purchase
values(12, 'gadget', NULL, NULL, 'april')
```

Let's try the following

```
select count(*) from Purchase
select count(quantity) from Purchase
```

```
select sum(quantity) from Purchase
```

```
select sum(quantity)
from Purchase
where quantity is not null;
```

30

## Aggregates and NULL Values

Null values are not used in aggregates

```
insert into Purchase
values(12, 'gadget', NULL, NULL, 'april')
```

Let's try the following

```
select count(*) from Purchase
select count(quantity) from Purchase
```

```
select sum(quantity) from Purchase
```

```
select sum(quantity)
from Purchase
where quantity is not null;
```

31

## Counting Duplicates

COUNT applies to duplicates, unless otherwise stated:

```
SELECT Count(product)
FROM Purchase
WHERE price > 4.99
```

same as Count(\*) if no nulls

We probably want:

```
SELECT Count(DISTINCT product)
FROM Purchase
WHERE price > 4.99
```

CSE 344 - Winter 2016

32

## More Examples

```
SELECT Sum(price * quantity)
FROM Purchase
```

```
SELECT Sum(price * quantity)
FROM Purchase
WHERE product = 'bagel'
```

What do they mean ?

CSE 344 - Winter 2016

33

## Simple Aggregations

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

```
SELECT Sum(price * quantity)
FROM Purchase
WHERE product = 'Bagel'
```



90 (= 60+30)

CSE 344 - Winter 2016

34

## Simple Aggregations

Purchase

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

```
SELECT Sum(price * quantity)
FROM Purchase
WHERE product = 'Bagel'
```



90 (= 60+30)

CSE 344 - Winter 2016

35

## Grouping and Aggregation

Purchase(product, price, quantity)

Find total quantities for all sales over \$1, by product.

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

Let's see what this means...

CSE 344 - Winter 2016

36

## Grouping and Aggregation

1. Compute the **FROM** and **WHERE** clauses.
2. Group by the attributes in the **GROUPBY**
3. Compute the **SELECT** clause: grouped attributes and aggregates.

FWGS

CSE 344 - Winter 2016

37

## 1&2. FROM-WHERE-GROUPBY

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

FWGS

WHERE price > 1

CSE 344 - Winter 2016

38

## 3. SELECT

FWGS

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

Product	TotalSales
Bagel	40
Banana	20

```
SELECT product, Sum(quantity) AS TotalSales
FROM Purchase
WHERE price > 1
GROUP BY product
```

39

## Other Examples

Compare these two queries:

```
SELECT product, count(*)
FROM Purchase
GROUP BY product
```

```
SELECT month, count(*)
FROM Purchase
GROUP BY month
```

```
SELECT product,
sum(quantity) AS SumQuantity,
max(price) AS MaxPrice
FROM Purchase
GROUP BY product
```

What does it mean ?

CSE 344 - Winter 2016

40

## Need to be Careful...

Product	Price	Quantity
Bagel	3	20
Bagel	1.50	20
Banana	0.5	50
Banana	2	10
Banana	4	10

```
SELECT product, max(quantity)
FROM Purchase
GROUP BY product
```

```
SELECT product, quantity
FROM Purchase
GROUP BY product
```

sqlite is WRONG on this query.

Advanced DBMS (e.g. SQL Server) gives an error

CSE 344 - Winter 2016

41

## Ordering Results

```
SELECT product, sum(price*quantity) as rev
FROM purchase
GROUP BY product
ORDER BY rev desc
```

CSE 344 - Winter 2016

42