CSE544
Data Management

Lectures 4 and 5:
More SQL

Announcements

- Homework 1 is due this Friday (Oct. 10)
- Homework 2: SQL on Snowflake
- Homework 3: lots of java!
- Homework 4, 5: later…

- Review 1 was due before this lecture

Recap

SQL

- SELECT-FROM-WHERE
- NULLs
- Joins, self-joins, outer-joins

Discussion: A Case Against…

Main points?

Discussion: A Case Against…

Main points?

- NULL break simple query equivalences
- Cast operation has ad-hoc semantics
- Syntax: vendor dependent
- Semantics: vendor dependent
- Will explain “certain answers” in class

Aggregates

Aggregate Operator

Aggregate op: set of values to single value

Aggregates in SQL:

- sum(1, 4, 3, 4) = 1+4+3+4 = 12
- max(1, 4, 3, 4) = 4
- min(1, 4, 3, 4) = 1
- count(1, 4, 3, 4) = 4
- avg(1, 4, 3, 4) = 3

Aggregate Operator

Aggregate op: set of values to single value

Aggregates in SQL:

- sum(1, 4, 3, 4) = 1+4+3+4 = 12
- max(1, 4, 3, 4) = 4
- min(1, 4, 3, 4) = 1
- count(1, **4**, 3, **4**) = 4
- avg(1, **4**, 3, **4**) = 3

May have duplicates

Count

```
SELECT  count(*)
FROM    Supplier
```

Supplier

sno	sname	scity	sstate
11	ACME	Seattle	WA
12	Walmart	Portland	OR
13	Safeway	Seattle	WA
13	Walmart	Seattle	WA

Count

Supplier

sno	sname	scity	sstate
11	ACME	Seattle	WA
12	Walmart	Portland	OR
13	Safeway	Seattle	WA
13	Walmart	Seattle	WA

```
SELECT  count(*)
FROM    Supplier
```

4

Count

Supplier

sno	sname	scity	sstate
11	ACME	Seattle	WA
12	Walmart	Portland	OR
13	Safeway	Seattle	WA
13	Walmart	Seattle	WA

```
SELECT  count(*)
FROM    Supplier
```

4

```
SELECT  count(sstate)
FROM    Supplier
```

Count

Supplier

sno	sname	scity	sstate
11	ACME	Seattle	WA
12	Walmart	Portland	OR
13	Safeway	Seattle	WA
13	Walmart	Seattle	WA

```
SELECT  count(*)
FROM    Supplier
```

4

```
SELECT  count(sstate)
FROM    Supplier
```

4

Count

Supplier

sno	sname	scity	sstate
11	ACME	Seattle	WA
12	Walmart	Portland	OR
13	Safeway	Seattle	WA
13	Walmart	Seattle	WA

```
SELECT  count(*)
FROM    Supplier
```

4

```
SELECT  count(sstate)
FROM    Supplier
```

4

```
SELECT  count(DISTINCT sstate)
FROM    Supplier
```

2

Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

First compute the query w/o aggregates:

```
select A, B, …
from…
where…
```

Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

First compute the query w/o aggregates:

```
select A, B, …
from…
where…
```

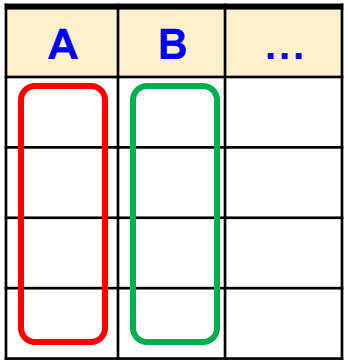
A	B	…

Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

First compute the query w/o aggregates:

```
select A, B, …
from…
where…
```



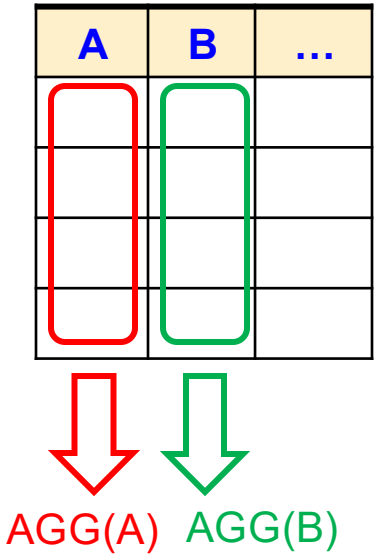
Then compute the aggregates

Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

First compute the query w/o aggregates:

```
select A, B, …
from…
where…
```



Then compute the aggregates

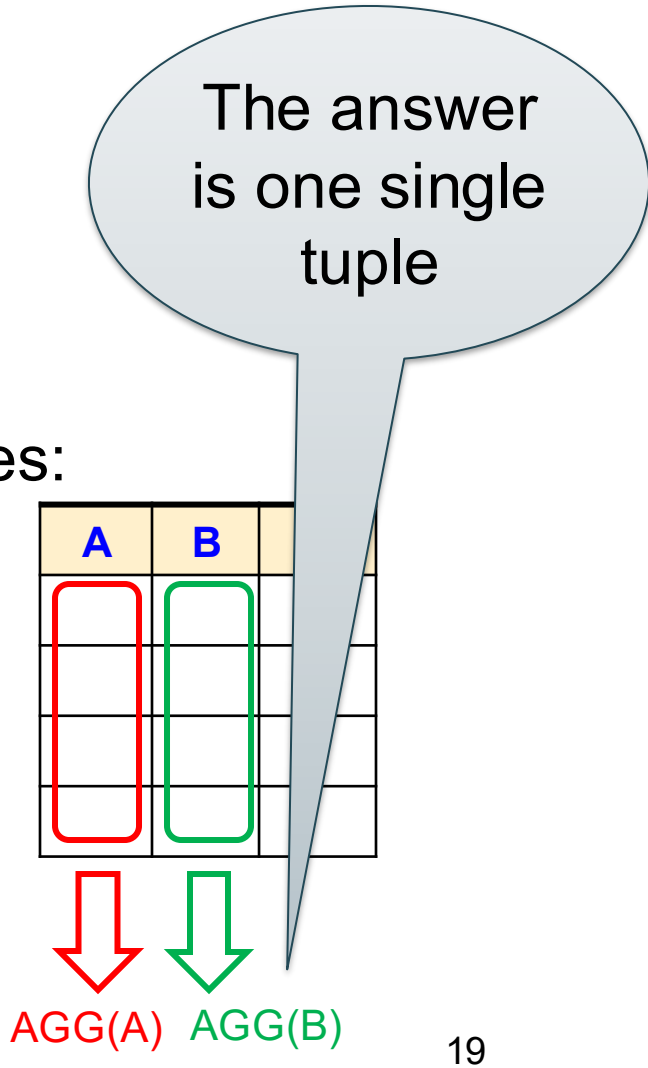
Semantics

```
select AGG1(A), AGG2(B), …
from…
where…
```

First compute the query w/o aggregates:

```
select A, B, …
from…
where…
```

Then compute the aggregates



Aggregates and Nulls

Reasonable semantics for NULL values

- Sum(1,5,NULL,3) = 9
- Count(1,5,NULL,3) = 3
- Count(NULL,NULL,NULL) = 0

GROUP-BY

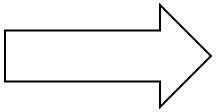
GROUP-BY

```
SELECT …
FROM   …
WHERE  …
GROUP BY …
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Examples

```
SELECT  max(psize)
FROM    Part
```

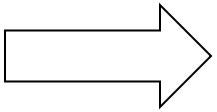


max
50

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

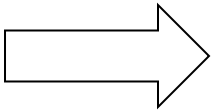
Examples

```
SELECT  max(psize)
FROM    Part
```



max
50

```
SELECT pcolor, max(psize)
FROM   Part
GROUP BY pcolor
```

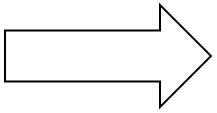


color	max
green	12
blue	50
gray	9
red	25
…	

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

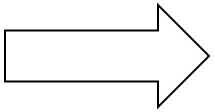
Examples

```
SELECT  max(psize)
FROM    Part
```



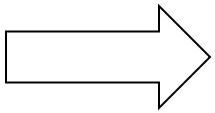
max
50

```
SELECT pcolor, max(psize)
FROM   Part
GROUP BY pcolor
```



color	max
green	12
blue	50
gray	9
red	25
…	

```
SELECT pcolor, max(psize), sum(psize)
FROM   Part
GROUP BY pcolor
```

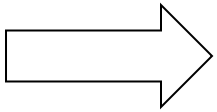


. . .

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subtleties

```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```

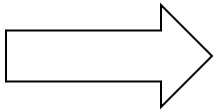


?

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subtleties

```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```



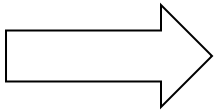
Same as distinct

```
SELECT DISTINCT pcolor
FROM   Part
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subtleties

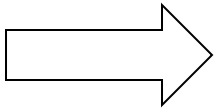
```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```



Same as distinct

```
SELECT DISTINCT pcolor
FROM   Part
```

```
SELECT pcolor, pname, max(psize)
FROM   Part
GROUP BY pcolor
```

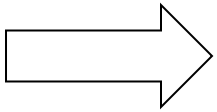


?

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subtleties

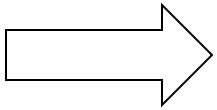
```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```



Same as distinct

```
SELECT DISTINCT pcolor
FROM   Part
```

```
SELECT pcolor, pname, max(psize)
FROM   Part
GROUP BY pcolor
```

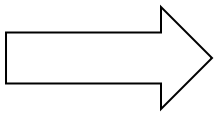


ERROR

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subtleties

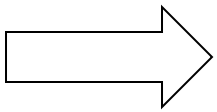
```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```



Same as distinct

```
SELECT DISTINCT pcolor
FROM   Part
```

```
SELECT pcolor, pname, max(psize)
FROM   Part
GROUP BY pcolor
```



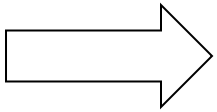
ERROR

Every attribute in SELECT must be listed in GROUP BY

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subtleties

```
SELECT pcolor
FROM   Part
GROUP BY pcolor
```



Same as distinct

```
SELECT DISTINCT pcolor
FROM   Part
```

```
SELECT pcolor, pname, max(psize)
FROM   Part
GROUP BY pcolor, pname
```

This is correct SQL

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Aggregates and Joins

Count # of Parts

```
SELECT count(*)
FROM   Part
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Aggregates and Joins

Count # of Parts

```
SELECT count(*) as C
FROM   Part
```

C
1540

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Aggregates and Joins

Count # of Parts

```
SELECT count(*) as C
FROM   Part
```

C
1540

Count # of Parts supplied by each city

Supplier(sno,sname,**scity**,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Aggregates and Joins

Count # of Parts

```
SELECT count(*) as C
FROM   Part
```

C
1540

Count # of Parts supplied by each city

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Aggregates and Joins

Count # of Parts

```
SELECT count(*) as C
FROM   Part
```

C
1540

Count # of Parts supplied by each city

```
SELECT x.scity, count(*)
FROM   Supplier x, Supply y, Part z
WHERE  x.sno = y.sno  and y.pno = z.pno
GROUP BY x.scity
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Aggregates and Joins

Count # of Parts

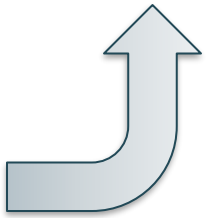
```
SELECT count(*) as C
FROM   Part
```

C
1540

City	C
Seattle	300
NYC	240
…	

Count # of Parts supplied by each city

```
SELECT x.scity, count(*) as C
FROM   Supplier x, Supply y, Part z
WHERE  x.sno = y.sno  and y.pno = z.pno
GROUP BY x.scity
```



Discussion

- GROUP-BY without an aggregate is equivalent to DISTINCT
- Every attribute in SELECT that is not aggregated must occur in GROUP-BY

HAVING

The HAVING Clause

```
SELECT …
FROM   …
WHERE  …
GROUP BY …
HAVING [condition w/ aggregates]
```

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

HAVING Clause

Compute the total quantity supplied by each supplier in 'WA'

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

HAVING Clause

Compute the total quantity supplied by each supplier in 'WA'

```
SELECT x.sno, x.sname, sum(y.qty)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno and x.sstate='WA'
GROUP BY x.sno, x.sname
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

HAVING Clause

Compute the total quantity supplied by each supplier in 'WA'

```
SELECT x.sno, x.sname, sum(y.qty)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno and x.sstate='WA'
GROUP BY x.sno, x.sname
```

Compute the total quantity supplied by each supplier who supplied > 100 parts

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

HAVING Clause

Compute the total quantity supplied by each supplier in 'WA'

```
SELECT x.sno, x.sname, sum(y.qty)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno and x.sstate='WA'
GROUP BY x.sno, x.sname
```

Compute the total quantity supplied by each supplier who supplied > 100 parts

```
SELECT x.sno, x.sname, sum(y.qty)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno, x.sname
HAVING count(*) > 100
```

Discussion

- WHERE clause:
  - Checks on tuple at a time
  - No aggregates allowed
- HAVING clause
  - Checks an entire group at a time
  - Aggregates OK
  - Other attributes must be in GROUP BY

Semantics

Semantics of SQL

- We know the nested loop semantics for
  SELECT-FROM-WHERE

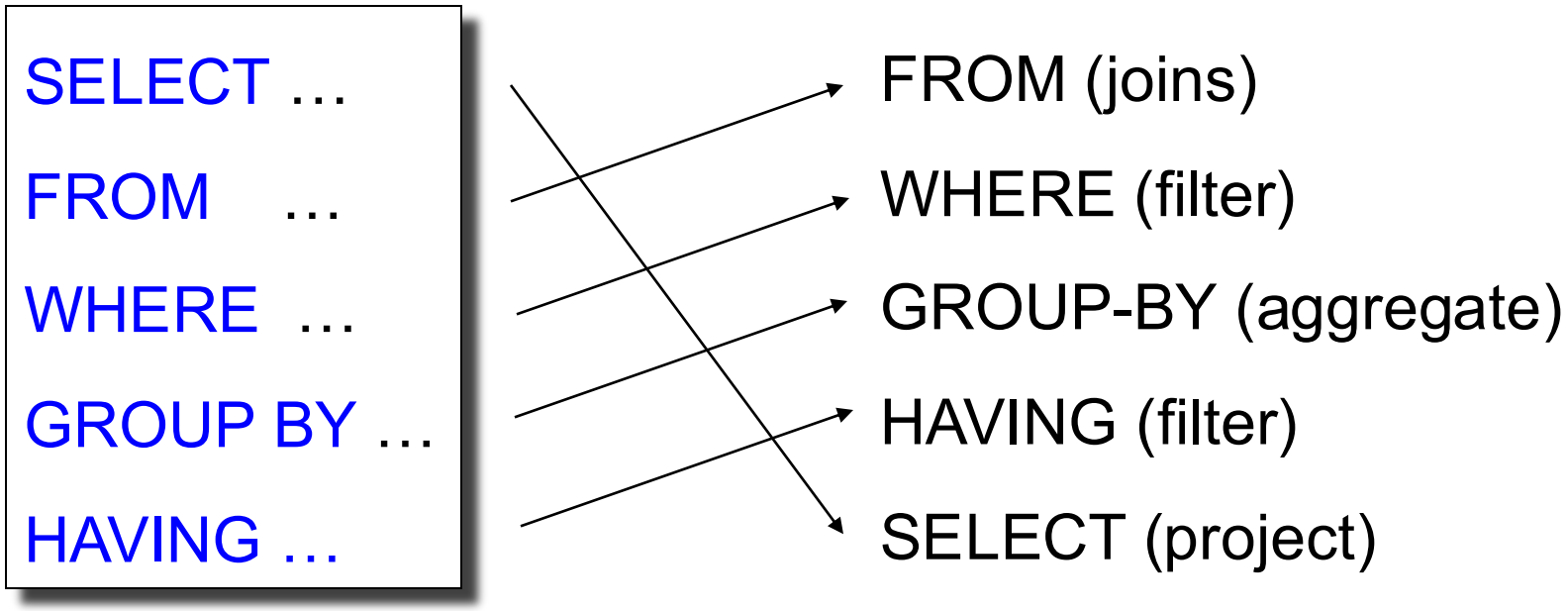
- Next:
  GROUP BY - HAVING

Semantics

```
SELECT …
FROM   …
WHERE  …
GROUP BY …
HAVING …
```

Paper *SQL Has Problems.* What is the logical order?

Semantics



Paper *SQL Has Problems.* What is the logical order?

Semantics

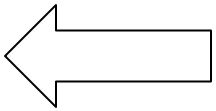
SELECT $a_1, \ldots, a_k, agg_1, agg_2$
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY $a_1, \ldots, a_k$
HAVING condition2 -- may have aggs

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 1: FROM-WHERE

a_1	…	a_k	b_1	…	b_1

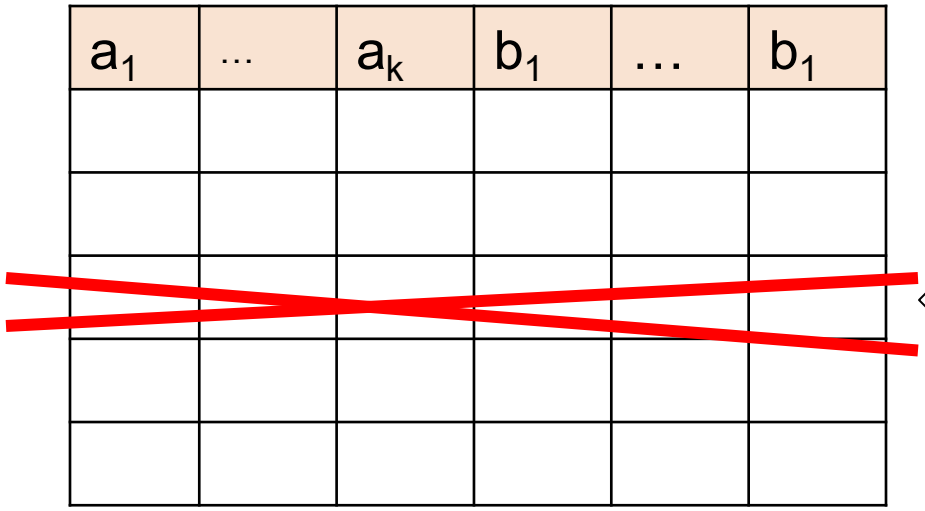


Check
WHERE condition1
in each row

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 1: FROM-WHERE



Check
WHERE condition1
in each row

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 1: FROM-WHERE

a_1	…	a_k	b_1	…	b_1

Semantics

```
SELECT a1, …, ak, agg1, agg2
FROM   R1 AS x1, R2 AS x2, …, Rn AS xn
WHERE  condition1
GROUP BY a1, …, ak
HAVING condition2          -- may have aggs
```

Step 2: GROUP BY

a_1	…	a_k	b_1	…	b_1
u	…	v			
u		v			
p		q			
p		q			
p		q			

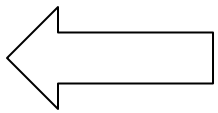
All attributes a_1, …, a_k, have the same value inside each group

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 3: HAVING

a_1	…	a_k	b_1	…	b_1
u	…	v			
u		v			
p		q			
p		q			
p		q			

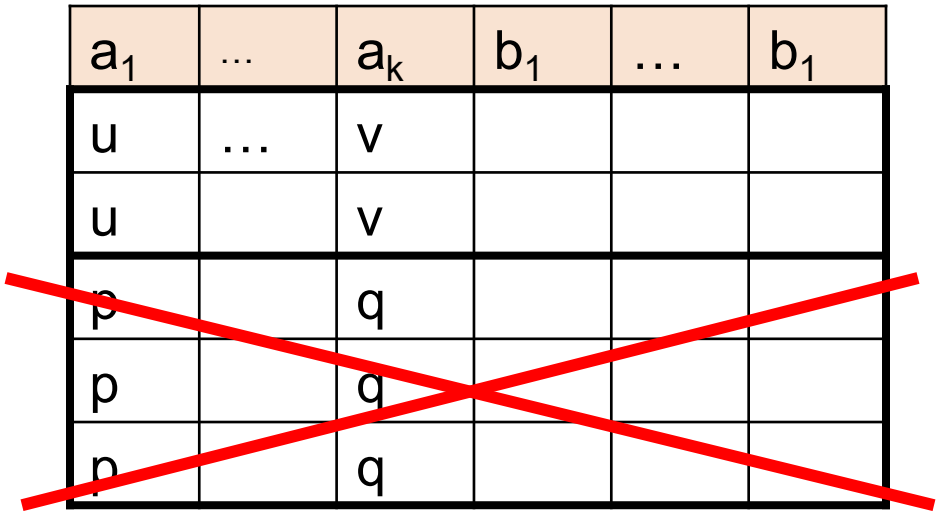


Check condition2 in each group

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 3: HAVING



Check condition2 in each group

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 3: HAVING

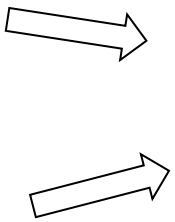
a_1	…	a_k	b_1	…	b_1
u	…	v			
u		v			
p		q			
p		q			
p		q			

Semantics

SELECT a_1, …, a_k, agg_1, agg_2
FROM R_1 AS x_1, R_2 AS x_2, …, R_n AS x_n
WHERE condition1
GROUP BY a_1, …, a_k
HAVING condition2 -- may have aggs

Step 4: SELECT

a_1	…	a_k	b_1	…	b_1
u	…	v			
u		v			
p		q			
p		q			
p		q			



a_1	…	a_k	agg_1	agg_2
u	…	v		
p		q		

Each group → one output

Discussion

- GROUP-BY is very versatile in SQL
- No analogous in programming languages: use nested loops instead

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

Discussion

- GROUP-BY is very versatile in SQL
- No analogous in programming languages: use nested loops instead

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

```
for x in Supplier:
  c = 0
  for y in Supply:
    if x.sno==y.sno:
      c = c+1
```

Discussion

- GROUP-BY is very versatile in SQL
- No analogous in programming languages: use nested loops instead

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

```
for x in Supplier:
  c = 0
  for y in Supply:
    if x.sno==y.sno:
      c = c+1
```

- The empty group problem (next)

Empty Groups

Empty Groups Problem

- Every group is non-empty
- Consequences:
  - count(*) > 0
  - sum(…) > 0 (assuming numbers are >0)
- Sometimes we want to return 0 counts:
  - Parts that never sold
  - Suppliers that never supplied
- Use outer joins: count(…) skips NULLs

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Empty Groups Problem

Compute the number of parts supplied by each supplier

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Empty Groups Problem

Compute the number of parts supplied by each supplier

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

Suppliers who never supplied any part will be missing: count(*) > 0

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Empty Groups Problem

Compute the number of parts supplied by each supplier

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

Suppliers who never supplied any part will be missing: count(*) > 0

```
SELECT x.sno, count(y.sno)
FROM   Supplier x
  LEFT OUTER JOIN Supply y
ON  x.sno=y.sno
GROUP BY x.sno
```

Now we can get count(y.sno)=0

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Empty Groups Problem

Compute the number of parts supplied by each supplier

```
SELECT x.sno, count(*)
FROM   Supplier x, Supply y
WHERE  x.sno=y.sno
GROUP BY x.sno
```

Suppliers who never supplied any part will be missing: count(*) > 0

```
SELECT x.sno, count(y.sno)
FROM   Supplier x
  LEFT OUTER JOIN Supply y
ON  x.sno=y.sno
GROUP BY x.sno
```

Now we can get count(y.sno)=0

Can we write count(*)?

Summary

- To include empty groups, use LEFT OUTER JOIN, then exploit NULLs
- Alternative: use subqueries. Will discuss later

The WITH Clause

WITH Clause

```
WITH
        tbl1 AS (SELECT … FROM …),
        tbl2 AS (SELECT … FROM …),
        …
SELECT … FROM …[tbl1, tbl2,…] …
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Warmup: find all parts supplied from Seattle

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Warmup: find all parts supplied from Seattle

```
SELECT z.*
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Warmup: find all parts supplied from Seattle

```
SELECT z.*
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

What is missing?

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Warmup: find all parts supplied from Seattle

```
SELECT DISTINCT z.*
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Find the average psize of all parts supplied from Seattle

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Find the average psize of all parts supplied from Seattle

```
SELECT avg(z.psize)
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Find the average psize of all parts supplied from Seattle

```
SELECT avg(z.psize)
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

What is wrong?

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Find the average psize of all parts supplied from Seattle

```
SELECT avg(z.psize)
FROM Supplier x, Supply y, Part z
WHERE x.scity = 'Seattle'
      and x.sno=y.sno and y.pno=z.pno;
```

What is wrong?

The same part may be supplied by more than one supplier in Seattle. Need to remove duplicates!

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Example

Find the average psize of all parts supplied from Seattle

```
WITH Tmp AS (
        SELECT DISTINCT z.pno, z.psize
        FROM Supplier x, Supply y, Part z
        WHERE x.scity = 'Seattle'
            and x.sno=y.sno and y.pno=z.pno)
SELECT avg(psize)
FROM Tmp;
```

Discussion

- WITH clause is very convenient for structuring complex queries:
  - Compute temporary tables T1, T2, …
  - Then compute final answer using T1, T2,…
- Use WITH generously in HW1

Finding Witnesses a.k.a. ARGMAX

The Witness aka ARGMAX

- Find the city with the largest population
- Find product/products with largest price
- …
- SQL does not have ARGMAX
- Two solutions:
  - Use intermediate relation in WITH
  - Self-join and HAVING

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using WITH

For each supplier, find the most expensive part they supply

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using WITH

For each supplier, find the most expensive part they supply

Finding the max price is easy:

```
SELECT x.sno, x.name, max(y.price)
FROM Supplier x, Supply y
WHERE x.sno = y.sno
GROUP BY x.sno, x.name
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using WITH

For each supplier, find the most expensive part they supply

Finding the max price is easy:

But we also want pno, pname:

Why does this not work?

```
SELECT x.sno, y.pno, x.name, max(y.price)
FROM Supplier x, Supply y
WHERE x.sno = y.sno
GROUP BY x.sno, x.name
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using WITH

For each supplier, find the most expensive part they supply

Compute max price in temporary table

```
WITH Temp AS
  (SELECT x.sno, x.sname, max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)
. . . . . .
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using WITH

For each supplier, find the most expensive part they supply

Compute max price in temporary table

```
WITH Temp AS
  (SELECT x.sno, x.sname, max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
WITH Temp AS
  (SELECT x.sno, x.sname,
                max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
WITH Temp AS
  (SELECT x.sno, x.sname,
                max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
WITH Temp AS
  (SELECT x.sno, x.sname,
                max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
WITH Temp AS
  (SELECT x.sno, x.sname,
                max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

Implicit DISTINCT

```
Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)
```

Using Self-joins and HAVING

For each supplier, find the most expensive part they supply

Join directly the temp table with Supply and Part

```
SELECT x.sno, x.sname, v.pno, v.pname
FROM Supplier x, Supply y, Supply u, Part v
WHERE x.sno = y.sno
      and x.sno = u.sno
      and u.pno = v.pno
HAVING max(y.price) = v.price
GROUP BY x.sno, x.sname,
                v.pno, v.pname, v.price
```

```
WITH Temp AS
  (SELECT x.sno, x.sname,
                max(y.price) as m
   FROM Supplier x, Supply y
   WHERE x.sno = y.sno
   GROUP BY x.sno, x.name)

SELECT t.sno, t.sname, v.pno, v.pname
FROM Temp t, Supply u, Part v
WHERE t.sno = u.sno
      and u.pno = v.pno
      and t.m = u.price;
```

Discussion

- Why doesn't SQL have ARGMAX?
  [in class]
- Solution 1: use temp table
- Solution 2: self-join + HAVING
- Solution 3: NOT EXISTS
- … [perhaps more]

Recap

- Aggregates, Group-by, Having
- Handling empty groups
- WITH clause
- Argmax/witness

Subqueries

Subqueries

- A subquery is a self-contained SQL query that occurs inside another query
- The subquery can be any of these clauses:
  - FROM
  - SELECT
  - WHERE
  - HAVING

Subqueries in FROM Clause

- Subquery in FROM: the same as in WITH
- Sometimes WITH is easier to read
- Some DBMS may not support both

Subqueries in FROM Clause

```
WITH Tmp AS (SELECT DISTINCT z.pno, z.psize
             FROM Supplier x, Supply y, Part z
             WHERE x.scity = 'Seattle'
                and x.sno=y.sno and y.pno=z.pno)
SELECT avg(psize)
FROM Tmp;
```

Subqueries in FROM Clause

```
WITH Tmp AS (SELECT DISTINCT z.pno, z.psize
             FROM Supplier x, Supply y, Part z
             WHERE x.scity = 'Seattle'
                and x.sno=y.sno and y.pno=z.pno)
SELECT avg(psize)
FROM Tmp;
```

same as:

```
SELECT avg(W.psize)
FROM (SELECT DISTINCT z.pno, z.psize
      FROM Supplier x, Supply y, Part z
      WHERE x.scity = 'Seattle'
           and x.sno=y.sno and y.pno=z.pno) as W;
```

Subqueries in SELECT

- SELECT: only scalar expressions
- May use subquery in SELECT if it returns a single value

Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
SELECT x.sno,
       ?

FROM   Supplier x
WHERE  x.scity = 'Seattle';
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
SELECT x.sno, (SELECT sum(y.qty)
               FROM Supply y
               WHERE x.sno = y.sno) AS T
FROM   Supplier x
WHERE  x.scity = 'Seattle';
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
SELECT x.sno, (SELECT sum(y.qty)
                      FROM Supply y
                      WHERE x.sno = y.sno) AS T
FROM   Supplier x
WHERE  x.scity = 'Seattle';
```

```
SELECT x.sno, sum(y.qty) as T
FROM   Supplier x, Supply y
WHERE x.sno = y.sno
      and x.scity = 'Seattle'
GROUP BY x.sno
```

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
SELECT x.sno, (SELECT sum(y.qty)
               FROM Supply y
               WHERE x.sno = y.sno) AS T
FROM   Supplier x
WHERE  x.scity = 'Seattle';
```

```
SELECT x.sno, sum(y.qty) as T
FROM   Supplier x, Supply y
WHERE x.sno = y.sno
      and x.scity = 'Seattle'
GROUP BY x.sno
```

Not equivalent!
WHY?

```
Supplier(sno,sname,scity,sstate)
Supply(sno,pno,qty,price)
Part(pno,pname,psize,pcolor)
```

Subqueries in SELECT

Compute the total quantity supplied by each supplier in Seattle

```
SELECT x.sno, (SELECT sum(y.qty)
               FROM Supply y
               WHERE x.sno = y.sno) AS T
FROM   Supplier x
WHERE  x.scity = 'Seattle';
```

Now they are equivalent

```
SELECT x.sno, sum(y.qty) as T
FROM   Supplier x LEFT OUTER JOIN Supply y
ON x.sno = y.sno
     and x.scity = 'Seattle'
GROUP BY x.sno
```

Subqueries in WHERE

Three SQL constructs:

- [NOT] EXISTS (SELECT…)
- X [NOT] IN (SELECT …)
- X > ALL | ANY (SELECT …)

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts that have some supplier offering them for < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts that have some supplier offering them for < $100

```
SELECT DISTINCT a.pno, a.pname
FROM    Part a, Supply b
WHERE b.price < 100 and b.pno = a.pno
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts that have some supplier offering them for < $100

```
SELECT DISTINCT a.pno, a.pname
FROM    Part a, Supply b
WHERE b.price < 100 and b.pno = a.pno
```

```
SELECT a.pno, a.pname
FROM    Part a
WHERE EXISTS
            (SELECT *
             FROM Supply b
             WHERE b.price < 100
                   and b.pno = a.pno)
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Natural language is ambiguous.
Question above is the same as:

Find all parts that are offered only for < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Find the other parts:

all parts that have some supplier offering them for >= $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Find the other parts:
all parts that have some supplier offering them for >= $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE EXISTS
           (SELECT *
            FROM Supply b
            WHERE b.price >= 100
                  and b.pno = a.pno)
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Find the other parts:
all parts that have some supplier offering them for >= $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
           (SELECT *
            FROM Supply b
            WHERE b.price >= 100
                  and b.pno = a.pno)
```

Negate!

Discussion: When To Use…

- Subqueries in FROM:
  - Better: use WITH
- Subqueries in SELECT:
  - Maybe for aggregates w/ empty groups
- Subqueries in WHERE:
  - Unavoidable from complex queries
  - One of the main usage: quantifiers (next)

Quantifiers

Quantifiers

Forall ∀, exists ∃

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Quantifiers

Forall ∀, exists ∃

Query language = Mathematical logic

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Quantifiers

Forall ∀, exists ∃

Query language = Mathematical logic

- Some Part was sold < $100:

$$\exists x \exists y \exists z (Supply(x, y, z) \wedge z < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Quantifiers

Forall ∀, exists ∃

Query language = Mathematical logic

- Some Part was sold < $100:

$$\exists x \exists y \exists z (Supply(x, y, z) \wedge z < 100)$$

- Every Part is sold at least once under $100:

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Quantifiers

Forall ∀, exists ∃

Query language = Mathematical logic

- Some Part was sold < $100:

$$\exists x \exists y \exists z (Supply(x, y, z) \wedge z < 100)$$

- Every Part is sold at least once under $100:

$$\forall y \exists x \exists z (Supply(x, y, z) \wedge z < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Quantifiers

Forall ∀, exists ∃

Query language = Mathematical logic

- Some Part was sold < $100:

$$\exists x \exists y \exists z (Supply(x, y, z) \wedge z < 100)$$

- Every Part is sold at least once under $100:

$$\forall y \exists x \exists z (Supply(x, y, z) \wedge z < 100)$$

$$\forall y \forall n (Part(y, n) \Rightarrow \exists x \exists z (Supply(x, y, z) \wedge z < 100))$$

Better

Quantifiers in SQL

- SQL semantics implicitly uses existential quantification

- In order to express universal quantification, we use negation, and apply DeMorgan's laws

DeMorgan Laws

$$\neg(A \wedge B) = \neg A \vee \neg B$$

$$\neg(A \vee B) = \neg A \wedge \neg B$$

$$\neg(A \Rightarrow B) = A \wedge \neg B$$

$$\neg\Big(\forall x\big(P(x)\big)\Big) = \exists x\big(\neg P(x)\big)$$

$$\neg\Big(\exists x\big(P(x)\big)\Big) = \forall x\big(\neg P(x)\big)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x, y) = Part(x, y) \wedge \exists z, w(Supply(z, x, w) \wedge w < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

"**Exists**" quantifier

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

"**For all**" quantifier

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

$$= Part(x,y) \wedge \neg(\exists z,w\ \neg(Supply(z,x,w) \Rightarrow w < 100))$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

$$= Part(x,y) \wedge \neg(\exists z,w\ \neg(Supply(z,x,w) \Rightarrow w < 100))$$

$$= Part(x,y) \wedge \neg(\exists z,w\ (Supply(z,x,w) \wedge w \geq 100))$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

$$= Part(x,y) \wedge \neg(\exists z,w\ \neg(\boldsymbol{Supply(z,x,w) \Rightarrow w < 100}))$$

$$= Part(x,y) \wedge \neg(\exists z,w\ (\boldsymbol{Supply(z,x,w) \wedge w \geq 100}))$$

$$\neg(A \Rightarrow B) = A \wedge \neg B$$

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts that have some supplier offering them for < $100

$$Answer(x,y) = Part(x,y) \wedge \exists z,w(Supply(z,x,w) \wedge w < 100)$$

Find all parts where all supplier offering them charge < $100

$$Answer(x,y) = Part(x,y) \wedge \forall z,w(Supply(z,x,w) \Rightarrow w < 100)$$

$$= Part(x,y) \wedge \neg(\exists z,w\ \neg(\boldsymbol{Supply(z,x,w) \Rightarrow w < 100}))$$

$$= Part(x,y) \wedge \neg(\exists z,w\ (\boldsymbol{Supply(z,x,w) \wedge w \geq 100}))$$

Ready for SQL

$$\neg(A \Rightarrow B) = A \wedge \neg B$$

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

$$= Part(x,y) \wedge \neg\exists u,v(Supplier(u,v) \wedge \neg\exists p\ Supply(u,x,p))$$

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

$$= \boldsymbol{Part(x,y)} \wedge \neg\exists u,v(Supplier(u,v) \wedge \neg\exists p\ Supply(u,x,p))$$

```
SELECT a.pno, a.pname
FROM   Part a
WHERE
```

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

$$= Part(x,y) \wedge \neg\exists \boldsymbol{u,v}(Supplier(u,v) \wedge \neg\exists p\ Supply(u,x,p))$$

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
       (




                                    )
```

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

$$= Part(x,y) \wedge \neg\exists u,v(\boldsymbol{Supplier(u,v)} \wedge \neg\exists \boldsymbol{p}\ Supply(u,x,p))$$

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
       (SELECT *
        FROM Supplier c
        WHERE NOT EXISTS
           (
                                    ))
```

Supplier(sno,sname)
Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Understanding Quantifiers

Find all parts supplied by all suppliers

$$Answer(x,y) =$$

$$= Part(x,y) \wedge \forall u,v(Supplier(u,v) \Rightarrow \exists p\ Supply(u,x,p))$$

$$= Part(x,y) \wedge \neg\exists u,v(Supplier(u,v) \wedge \neg\exists \boldsymbol{p\ Supply(u,x,p)})$$

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
      (SELECT *
       FROM Supplier c
       WHERE NOT EXISTS
          (SELECT * FROM Supply b
           WHERE a.pno=b.pno and b.sno=c.sno))
```

Back to SQL

- Now that we understand the logic of quantifiers, let's return to SQL

- SQL has three ways to express quantifiers: we will discuss their pros and cons next

Subqueries in WHERE

- EXISTS(….)
  check if empty
- NOT EXISTS(…)
  check if not empty

Subqueries in WHERE

- EXISTS(….)
  check if empty
- NOT EXISTS(…)
  check if not empty
- X IN (…)
  check if X in the set
- X NOT IN (…)
  check if X not in set

Subqueries in WHERE

- EXISTS(….)
  check if empty
- NOT EXISTS(…)
  check if not empty
- X IN (…)
  check if X in the set
- X NOT IN (…)
  check if X not in set
- X > SOME (…)
  ∃Y in (…) and X>Y
- X > ALL (…)
  ∀Y in (…): X > Y

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM    Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

If evaluated naively, which query is more efficient?

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

Correlated subquery

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

If evaluated naively, which query is more efficient?

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

Correlated subquery

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

Uncorrelated

If evaluated naively, which query is more efficient?

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

Supply(sno,pno,price)
Part(pno,pname)

Simplified schema

Subqueries in WHERE

Find all parts where all supplier offering them charge < $100

```
SELECT a.pno, a.pname
FROM   Part a
WHERE NOT EXISTS
        (SELECT *
         FROM Supply b
         WHERE b.price >= 100
               and b.pno = a.pno)
```

```
SELECT a.pno, a.pname
FROM   Part a
WHERE a.pno NOT IN
        (SELECT b.pno
         FROM Supply b
         WHERE b.price >= 100)
```

```
SELECT a.pno, a.pname
FROM   Part a
WHERE 100 < ALL
        (SELECT b.price
         FROM Supply b
         WHERE b.pno = a.pno)
```

Discussion

- Queries w/ existential quantifiers can be unnested into SELECT-FROM-WHERE

- Queries w/ universal quantifier cannot

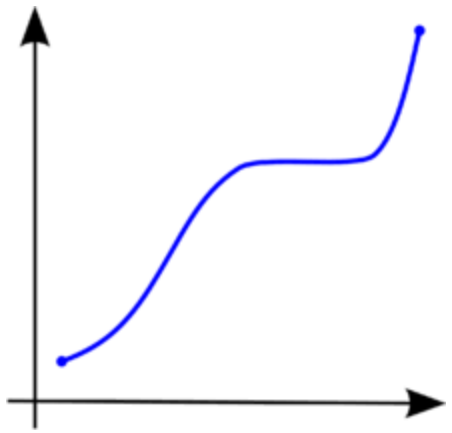
We will prove this next

Monotone Queries

Montone* Functions

A function $f: R \rightarrow R$ is **monotone** … WHEN?

Monotone:
$x^3 + x^2$,
e^x,
$\log(x)$,
…



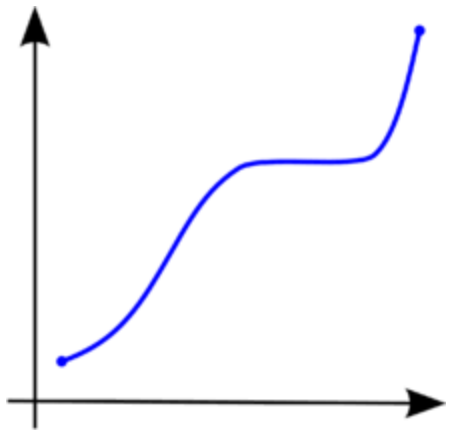
Non-Monotone:
$x^3 - x^2$,
e^{-x},
$\frac{1}{x}$,
…

*we mean *monotonically increasing*

Montone* Functions

A function $f: R \rightarrow R$ is **monotone** if $x \leq y$ implies $f(x) \leq f(y)$

Monotone:
$x^3 + x^2$,
e^x,
$\log(x)$,
…



Non-Monotone:
$x^3 - x^2$,
e^{-x},
$\frac{1}{x}$,
…

*we mean *monotonically increasing*

Monotone Queries

A query Q is monotone if $I \subseteq J$ implies $q(I) \subseteq q(J)$

Monotone Queries

A query Q is monotone if $I \subseteq J$ implies $q(I) \subseteq q(J)$

Adding tuples to the input does
not remove tuples from the output

Monotone Queries

A query Q is monotone if $I \subseteq J$ implies $q(I) \subseteq q(J)$

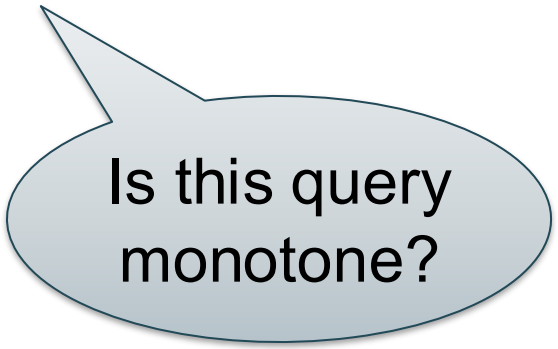
Adding tuples to the input does
not remove tuples from the output

Notice: this is a property of the the query's semantics, not of its expression in SQL

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts that have some supplier offering them for < $100



Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts that have some supplier offering them for < $100

Supply

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

```
Supply(sno,pno,price)
Part(pno,pname)
```

Monotone Queries

Find all parts that have some supplier offering them for < $100

Supply

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts that have some supplier offering them for < $100

Supply

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p200	99

pno	pname
p100	phone
p200	mouse
p300	lamp

J

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts that have some supplier offering them for < $100

Supply

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p200	99

pno	pname
p100	phone
p200	mouse
p300	lamp

J

pno	pname
p100	phone
p200	mouse

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Monotone

Find all parts that have some supplier offering them for < $100

Supply

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p200	99

pno	pname
p100	phone
p200	mouse
p300	lamp

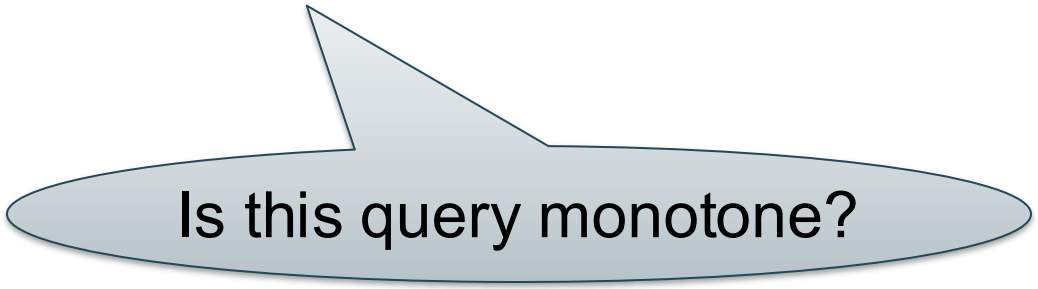
J

pno	pname
p100	phone
p200	mouse

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100



Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Why?

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Why?

$$Part(x, y) \wedge \forall z, p(Supply(z, x, p) \Rightarrow p < 100)$$

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Why?

$$Part(x, y) \wedge \forall z, p(Supply(z, x, p) \Rightarrow p < 100)$$

If $Supply(z, x, p)$ is FALSE, then $Supply(z, x, p) \Rightarrow p < 100$ is TRUE

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Why?

$$Part(x, y) \wedge \forall z, p(Supply(z, x, p) \Rightarrow p < 100)$$

If $Supply(z, x, p)$ is FALSE, then $Supply(z, x, p) \Rightarrow p < 100$ is TRUE

Hence (p200,mouse) is in the output

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p100	199

pno	pname
p100	phone
p200	mouse
p300	lamp

J

Supply(sno,pno,price)
Part(pno,pname)

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p100	199

pno	pname
p100	phone
p200	mouse
p300	lamp

J

pno	pname
~~p100~~	~~phone~~
p200	mouse
p300	lamp

Supply(sno,pno,price)
Part(pno,pname)

Non-Monotone

Monotone Queries

Find all parts where all suppliers offering them charge < $100

Supply

sno	pno	price
s01	p100	20
s02	p100	50
s01	p300	30

Part

pno	pname
p100	phone
p200	mouse
p300	lamp

I

pno	pname
p100	phone
p200	mouse
p300	lamp

sno	pno	price
s01	p100	200
s02	p100	50
s01	p200	300
s03	p100	199

pno	pname
p100	phone
p200	mouse
p300	lamp

J

pno	pname
~~p100~~	~~phone~~
p200	mouse
p300	lamp

A Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

A Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Proof. Consider a SQL query:

```
SELECT attrs
FROM T1, T2, ...
WHERE condition
```

A Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Proof. Consider a SQL query:

```
SELECT attrs
FROM T1, T2, ...
WHERE condition
```

Its nested loop semantics is:

```
for each r1 in T1:
 for each t2 in T2:
   for each t3 in T3:
       …
       if (condition):
          output (a1,a2,…)
```

A Theorem

Every SELECT-FROM-WHERE query without subqueries and without aggregates is monotone

Proof. Consider a SQL query:

```
SELECT attrs
FROM T1, T2, ...
WHERE condition
```

Its nested loop semantics is:

```
for each r1 in T1:
 for each t2 in T2:
   for each t3 in T3:
       …
       if (condition):
          output (a1,a2,…)
```

If we insert a tuple into one of the input relations T_i, we will not remove any tuples from the output.

An Application

The query:

Find all parts where all supplier offering them charge < $100

Cannot be unnested without using aggregates

The End (of the SQL lectures)

- In this class we only use the fragment discussed so far
- Look up scalar function as needed:
  - Substring operations, math functions, etc
- We don't discuss, and don't use in HW: WINDOW function, GROUP SETs, etc