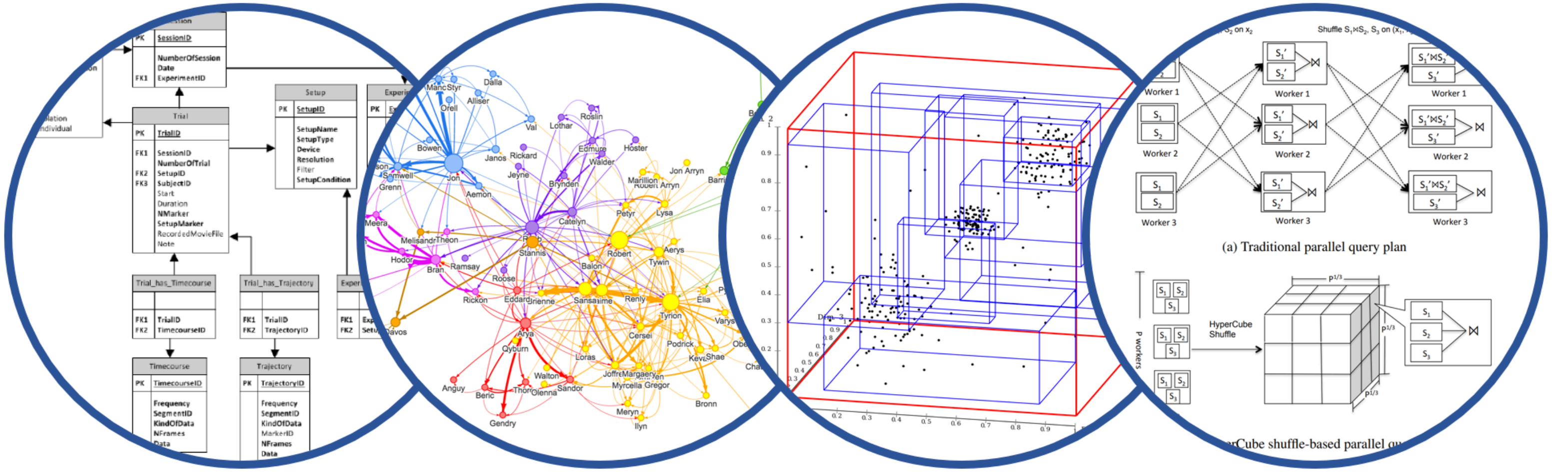


# Database System Internals

# Query Optimization (part 3)

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

# Announcements

- Grading Lab1 almost done…

- If you didn't pass a test for Lab1, talk to a TA ASAP (like: yesterday…)
- You must pass all tests in lab1 in order to do the other labs

- Lab2 part1 is due tonight; part 2 due next Friday

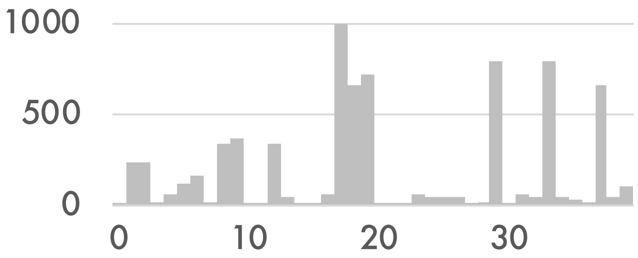
- HW2 due on Monday

# Cardinality Estimation Summary

- Uses simple formula, with lots of assumptions
- Histograms help alleviate uniformity assumption:
  - Partition attribute domain into buckets
  - Store #tuples per bucket
  - Make uniformity assumption within each bucket

# Types of Histograms

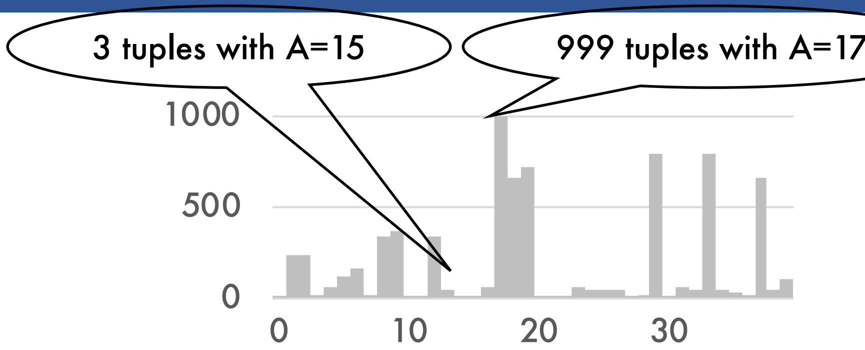
- Eqwidth



- Eqdepth

- V-optimal: minimize error

# Types of Histograms



- Eqwidth

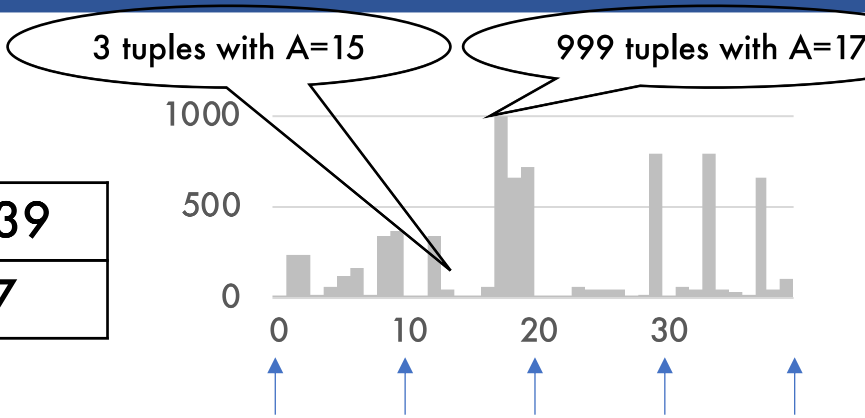
- Eqdepth

- V-optimal: minimize error

# Types of Histograms

- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |



3 tuples with A=15

999 tuples with A=17

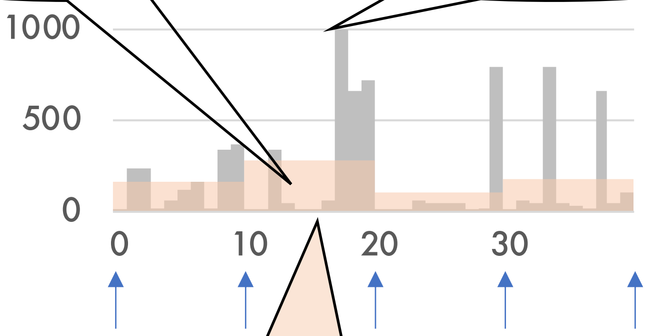
- Eqdepth

- V-optimal: minimize error

# Types of Histograms

- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |



3 tuples with A=15

999 tuples with A=17

By uniformity,
286 tuples with A=15
286 tuples with A=17
etc

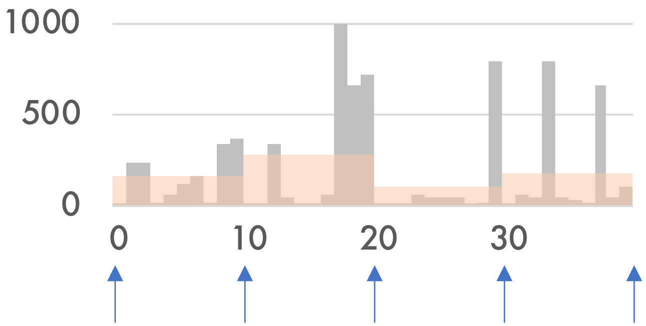
- Eqdepth

- V-optimal: minimize error

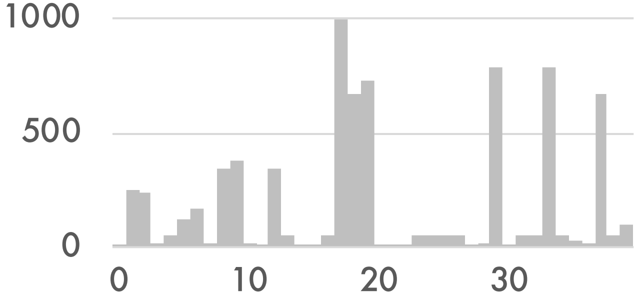
# Types of Histograms

- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |



- Eqdepth

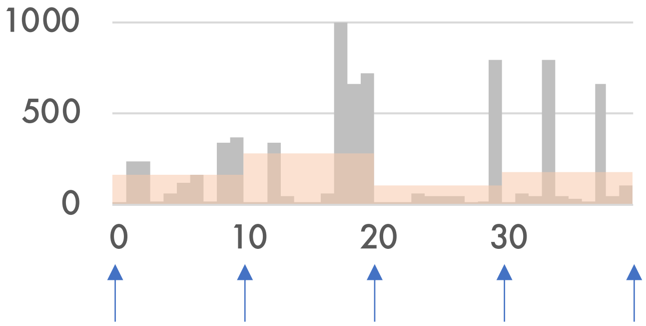


- V-optimal: minimize error

# Types of Histograms

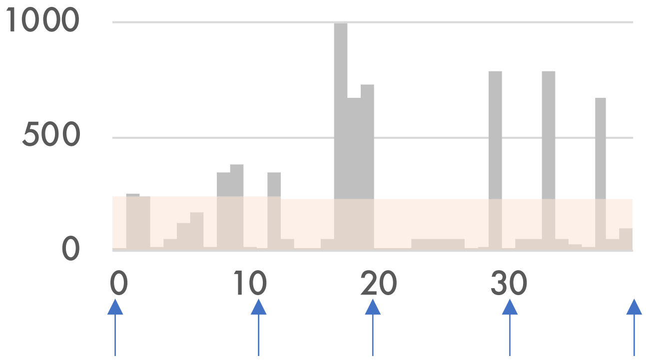
- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |



- Eqdepth

| Attr = | 0..12 | 13..18 | 19..31 | 32..39 |
|---|---|---|---|---|
| #tuples | 1943 | 1779 | 1822 | 1767 |

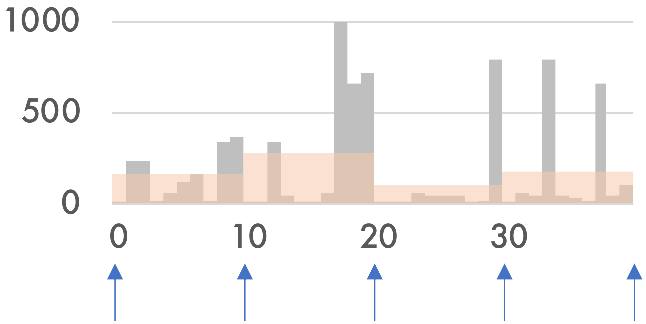


- V-optimal: minimize error

# Types of Histograms

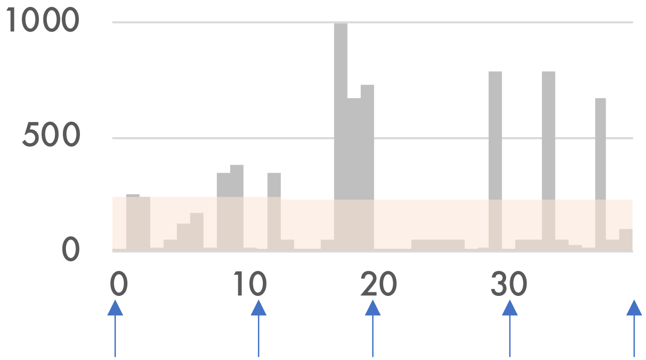
- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |

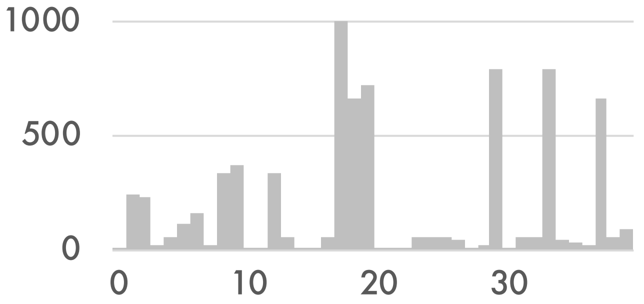


- Eqdepth

| Attr = | 0..12 | 13..18 | 19..31 | 32..39 |
|---|---|---|---|---|
| #tuples | 1943 | 1779 | 1822 | 1767 |



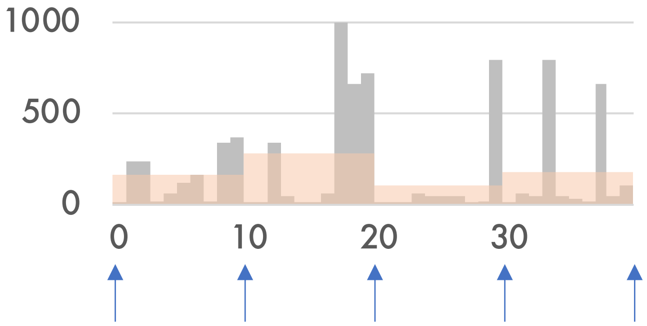
- V-optimal: minimize error



# Types of Histograms

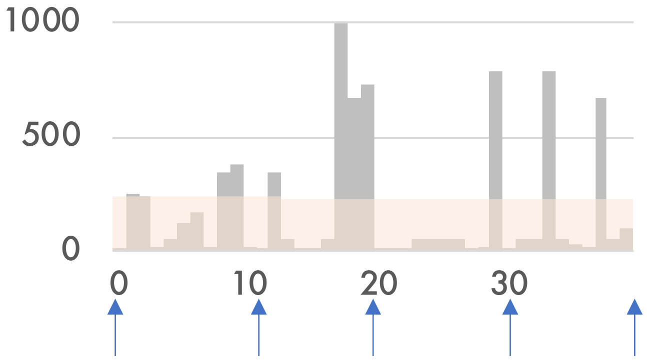
- Eqwidth

| Attr = | 0..9 | 10..19 | 20..29 | 30..39 |
|---|---|---|---|---|
| #tuples | 1585 | 2860 | 1039 | 1827 |



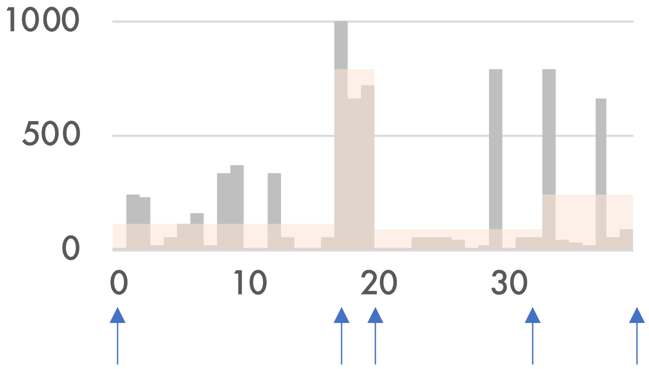
- Eqdepth

| Attr = | 0..12 | 13..18 | 19..31 | 32..39 |
|---|---|---|---|---|
| #tuples | 1943 | 1779 | 1822 | 1767 |



- V-optimal: minimize error

| Attr = | 0..16 | 17..19 | 20..34 | 35..39 |
|---|---|---|---|---|
| #tuples | 2056 | 2389 | 1152 | 1714 |



Minimizes $\sum_a |\text{true-\#tuples}(a) - \text{estimate-\#tuples}(a)|^2$

# Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY?

- *Not* updated during database update, but recomputed periodically
  - WHY?

- Multidimensional histograms rarely used
  - WHY?

# Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY? All histograms are kept in main memory during query optimization; plus need fast access

- *Not* updated during database update, but recomputed periodically
  - WHY?

- Multidimensional histograms rarely used
  - WHY?

# Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY? All histograms are kept in main memory during query optimization; plus need fast access

- *Not* updated during database update, but recomputed periodically
  - WHY? Histogram update creates a write conflict; would dramatically slow down transaction throughput

- Multidimensional histograms rarely used
  - WHY?

# Difficult Questions on Histograms

- Small number of buckets
  - Hundreds, or thousands, but not more
  - WHY? All histograms are kept in main memory during query optimization; plus need fast access

- *Not* updated during database update, but recomputed periodically
  - WHY? Histogram update creates a write conflict; would dramatically slow down transaction throughput

- Multidimensional histograms rarely used
  - WHY? Too many possible multidimensional histograms, unclear which ones to choose and how to use

# Query Optimization

Three components:

- Cost/cardinality estimation
- Search space ← this lecture
- Search algorithm

# Search Space

- The "Search Space" means the set of possible plans that the optimizers considers

- Two dimensions:
  - Which algebraic laws does the optimizer support
  - Which shapes of plans does the optimizer search

# Algebraic Laws

- Basic laws
- Generalized distributivity law
- Laws using constraints

# Basic Laws: Selections, Projections, Joins

- Selections
  - Commutative: $\sigma_{c1}(\sigma_{c2}(R)) = \sigma_{c2}(\sigma_{c1}(R))$
  - Cascading: $\sigma_{c1 \wedge c2}(R) = \sigma_{c2}(\sigma_{c1}(R))$

- Projections
  - Cascading

- Joins
  - Commutative : $R \bowtie S = S \bowtie R$
  - Associative: $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$

# Pushing Selections Down

- Example: R(A, B, C, D), S(E, F, G)

$\sigma_{F=3}(R \bowtie_{D=E} S) =$

$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) =$

# Pushing Selections Down

- Example: R(A, B, C, D), S(E, F, G)

$\sigma_{F=3}(R \bowtie_{D=E} S) = R \bowtie_{D=E} \sigma_{F=3}(S)$

$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) =$

# Pushing Selections Down

- Example: R(A, B, C, D), S(E, F, G)

$\sigma_{F=3}(R \bowtie_{D=E} S) = R \bowtie_{D=E} \sigma_{F=3}(S)$

$\sigma_{A=5 \text{ AND } G=9}(R \bowtie_{D=E} S) = \sigma_{A=5}(R) \bowtie_{D=E} \sigma_{G=9}(S)$

# Union

- Commutativity: $R \cup S = S \cup R$

- Associativity: $R \cup (S \cup T) = (R \cup S) \cup T$

- Distributivity: $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

# Laws Involving Projections

$$\Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S))$$

$$\Pi_M(\Pi_N(R)) = \Pi_M(R)$$

/* note that $M \subseteq N$ */

- Example R(A,B,C,D), S(E, F, G)

$$\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{?}(\Pi_{?}(R) \bowtie_{D=E} \Pi_{?}(S))$$

# Laws Involving Projections

$$\Pi_M(R \bowtie S) = \Pi_M(\Pi_P(R) \bowtie \Pi_Q(S))$$

$$\Pi_M(\Pi_N(R)) = \Pi_M(R)$$

/* note that $M \subseteq N$ */

- Example R(A,B,C,D), S(E, F, G)

$$\Pi_{A,B,G}(R \bowtie_{D=E} S) = \Pi_{A,B,G}(\Pi_{A,B,D}(R) \bowtie_{D=E} \Pi_{E,G}(S))$$

# Generalized Distributivity Law

- Distributivity law in arithmetic:
  $a * (x + y + z) = a*x + a*y + a*z$

- The Generalized Distributivity Law extends this to all kinds of joins and aggregates
  - Very important for large scale analytics and ML
  - Very few optimizers support it!!!

- We will describe GD by examples

# GD: Example 1

R(A,B)
S(C,D)

```
select R.A, sum(S.D)
from R, S
where R.B = S.C
group by R.A
```

Apparently, we must compute the join before we can group by and aggregate…

But there is a better way.

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D))$$

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D))$

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D))$

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R $\bowtie$ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result**

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

$\gamma$(S):

| C | D |
|---|---|
| b | x+y |
| h | z |

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R $\bowtie$ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

$\gamma$(S):

| C | D |
|---|---|
| b | x+y |
| h | z |

R $\bowtie$ $\gamma$(S)

| A | B | C | D |
|---|---|---|---|
| a | b | b | x+y |
| c | b | b | x+y |
| a | h | h | z |

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

$R \bowtie S$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

$\gamma(S)$:

| C | D |
|---|---|
| b | x+y |
| h | z |

$R \bowtie \gamma(S)$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x+y |
| c | b | b | x+y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | (x+y)+z |
| c | x+y |

# GD: Example 1

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | x+y+z |
| c | x+y |

$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{A,\ sum(D)}(R(A,B) \bowtie_{B=C} (\gamma_{C,\ sum(D)} S(C,D)))$$

$\gamma(S)$:

| C | D |
|---|---|
| b | x+y |
| h | z |

R ⋈ $\gamma(S)$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x+y |
| c | b | b | x+y |
| a | h | h | z |

Result

| A | Sum |
|---|---|
| a | (x+y)+z |
| c | x+y |

**Associativity: x+y+z = (x+y) + z**

# GD: Example 2

R(A,B)
S(C,D)

```
select R.A, count(*)
from R, S
where R.B = S.C
group by R.A
```

# GD: Example 2

R(A,B)
S(C,D)

```
select R.A, count(*)
from R, S
where R.B = S.C
group by R.A
```

```
select R.A, sum(1)
from R, S
where R.B = S.C
group by R.A
```

Same as the previous query
because count(*) is the same as sum(1)

Work this out at home!!

# GD: Example 3

R(A,B)
S(C,D)

```
select R.A, max(D)
from R, S
where R.B = S.C
group by R.A
```

# GD: Example 3

R(A,B)
S(C,D)

```
select R.A, max(D)
from R, S
where R.B = S.C
group by R.A
```

Same as the previous examples,
Because max is also associative

max(a, b, c, d) = max( max(a,b), max(c,d))

# GD: Example 3

R(A,B)
S(C,D)

```
select R.A, max(D)
from R, S
where R.B = S.C
group by R.A
```

Same as the previous examples,
Because max is also associative

max(a, b, c, d) = max( max(a,b), max(c,d))

Same here: work this out at home!!

# GD: Example 4

Try this in your favorite SQL engine (postgres or sqlite or …)

```
select count(*)
from myBigTable as x;
```

Answer: 2519105

Time : <2 seconds

# GD: Example 4

Try this in your favorite SQL engine (postgres or sqlite or ...)

```
select count(*)
from myBigTable as x;
```

Answer: 2519105
Time : <2 seconds

```
select count(*)
from myBigTable as x, myBigTable as y;
```

Timeout...

# GD: Example 4

Try this in your favorite SQL engine (postgres or sqlite or …)

```
select count(*)
from myBigTable as x;
```

Answer: 2519105
Time : <2 seconds

```
select count(*)
from myBigTable as x, myBigTable as y;
```

Timeout…

What should be the answer?

# GD: Example 5

R(A,B)
S(C,D)

```
select sum(R.A*S.D)
from R, S
where R.B = S.C
group by R.A
```

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

$R \bowtie S$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

$\gamma_{\text{sum}(A*D)}(R(A,B) \bowtie_{B=C} S(C,D))$

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result:**
a*x + a*y + c*x + c*y + a*z

$$\gamma_{\mathrm{sum}(A*D)}(R(A,B) \bowtie_{B=C} S(C,D))$$

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

$R \bowtie S$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result:**
a*x + a*y + c*x + c*y + a*z

$$\gamma_{\mathrm{sum(A*D)}}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{\mathrm{sum(A*D)}}(\gamma_{B,\ \mathrm{sum(A)}}(R(A,B)) \bowtie_{B=C} (\gamma_{C,\ \mathrm{sum(D)}} S(C,D)))$$

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result:**
a*x + a*y + c*x + c*y + a*z

$$\gamma_{\mathrm{sum(A*D)}}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{\mathrm{sum(A*D)}}(\gamma_{B,\ \mathrm{sum(A)}}(R(A,B)) \bowtie_{B=C} (\gamma_{C,\ \mathrm{sum(D)}} S(C,D)))$$

$\gamma$(R):

| A | B |
|---|---|
| a+c | b |
| a | f |
| a | h |

$\gamma$(S):

| C | D |
|---|---|
| b | x+y |
| h | z |

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

$R \bowtie S$

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result:**
$a*x + a*y + c*x + c*y + a*z$

$$\gamma_{\text{sum}(A*D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$

$$\gamma_{\text{sum}(A*D)}(\gamma_{B,\ \text{sum}(A)}(R(A,B)) \bowtie_{B=C} (\gamma_{C,\ \text{sum}(D)} S(C,D)))$$

$\gamma(R)$:

| A | B |
|---|---|
| a+c | b |
| a | f |
| a | h |

$\gamma(S)$:

| C | D |
|---|---|
| b | x+y |
| h | z |

$\gamma(R) \bowtie \gamma(S)$

| A | B | C | D |
|---|---|---|---|
| a+c | b | b | x+y |
| a | h | h | z |

# GD: Example 5

R:

| A | B |
|---|---|
| a | b |
| c | b |
| a | f |
| a | h |

S:

| C | D |
|---|---|
| b | x |
| b | y |
| h | z |

R ⋈ S

| A | B | C | D |
|---|---|---|---|
| a | b | b | x |
| a | b | b | y |
| c | b | b | x |
| c | b | b | y |
| a | h | h | z |

**Result:**
a*x + a*y + c*x + c*y + a*z

$$\gamma_{\mathrm{sum(A*D)}}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{\mathrm{sum(A*D)}}(\gamma_{\mathrm{B,\ sum(A)}}(R(A,B)) \bowtie_{B=C} (\gamma_{\mathrm{C,\ sum(D)}}S(C,D)))$$

$\gamma$(R):

| A | B |
|---|---|
| a+c | b |
| a | f |
| a | h |

$\gamma$(S):

| C | D |
|---|---|
| b | x+y |
| h | z |

$\gamma$(R) ⋈ $\gamma$(S)

| A | B | C | D |
|---|---|---|---|
| a+c | b | b | x+y |
| a | h | h | z |

**Distributivity:** (a+c) * (x + y) + a*z = a*x + a*y + c*x + c*y + a*z

# Other Distributive Operators

- Distributivity of * over +:

  $a * (x + y + z) = a*x + a*y + a*z$

- Distributivity of * over max (when values ≥ 0):

  $a * \max(x, y, z) = \max(a*x, a*y, a*z)$

- Distributivity of * over min (when values ≥ 0):

  $a * \min(x, y, z) = \min(a*x, a*y, a*z)$

# Other Distributive Operators

R(A,B)
S(C,D)

When A ≥ 0 and D ≥ 0 then:

$$\gamma_{\max(A*D)}(R(A,B) \bowtie_{B=C} S(C,D)) =$$
$$\gamma_{\max(A*D)}(\gamma_{B,\ \max(A)}(R(A,B)) \bowtie_{B=C} (\gamma_{C,\ \max(D)}S(C,D)))$$

Distributivity and associativity (when a,c,x,y,z ≥ 0):
max(max(a,c) * max(x,y), a*z) = max(a*x, a*y, c*x, c*y, a*z)

# GD: Example 6

```
Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, category, price)
```

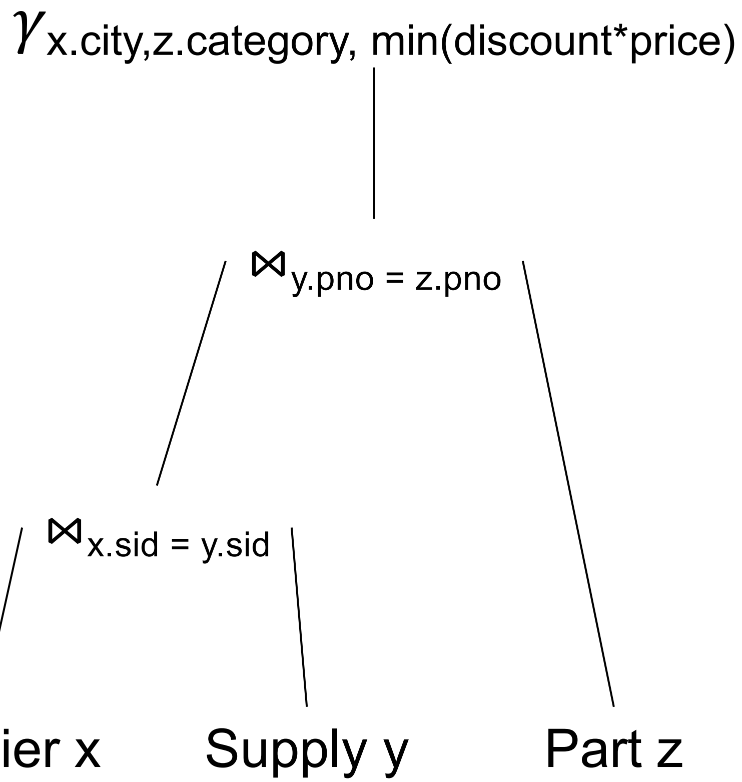
```
select x.city, z.category,
       min(x.discount*z.price)
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
group by z.category;
```

# GD: Example 6

Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, category, price)

```sql
select x.city, z.category,
        min(x.discount*z.price)
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
group by z.category;
```

**Assume discount, price ≥ 0**



$\gamma_{\text{x.city,z.category, min(discount*price)}}$

$\bowtie_{\text{y.pno = z.pno}}$

$\bowtie_{\text{x.sid = y.sid}}$

Supplier x

Supply y

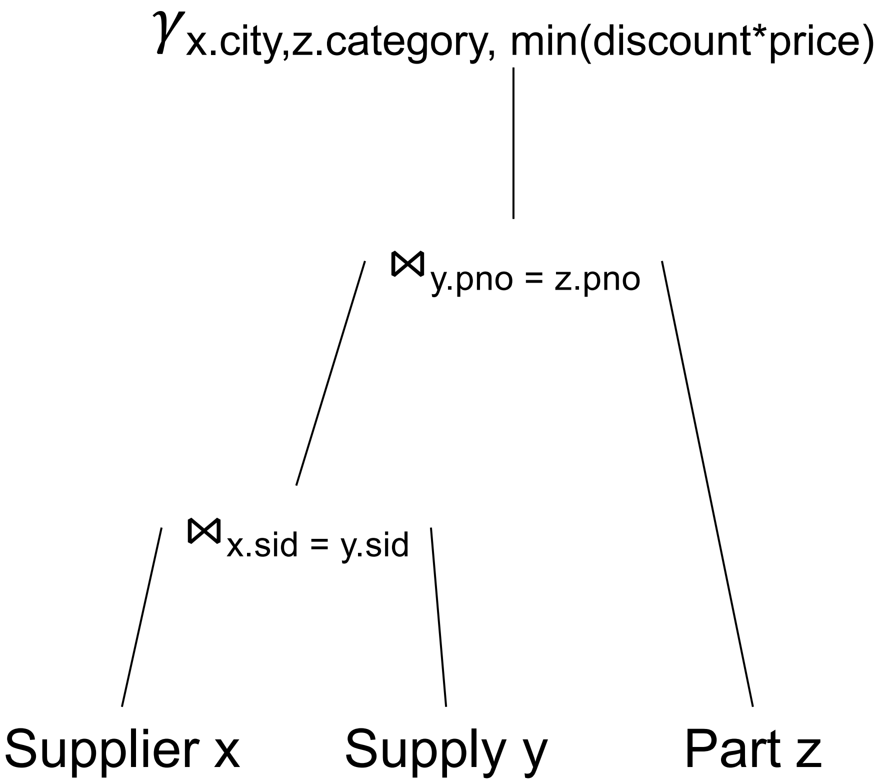
Part z

# GD: Example 6

`Supplier(sid,name,discount,city)`
`Supply(sid, pno)`
`Part(pno, pname, category, price)`

```
select x.city, z.category,
       min(x.discount*z.price)
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
group by z.category;
```

Assume discount, price ≥ 0

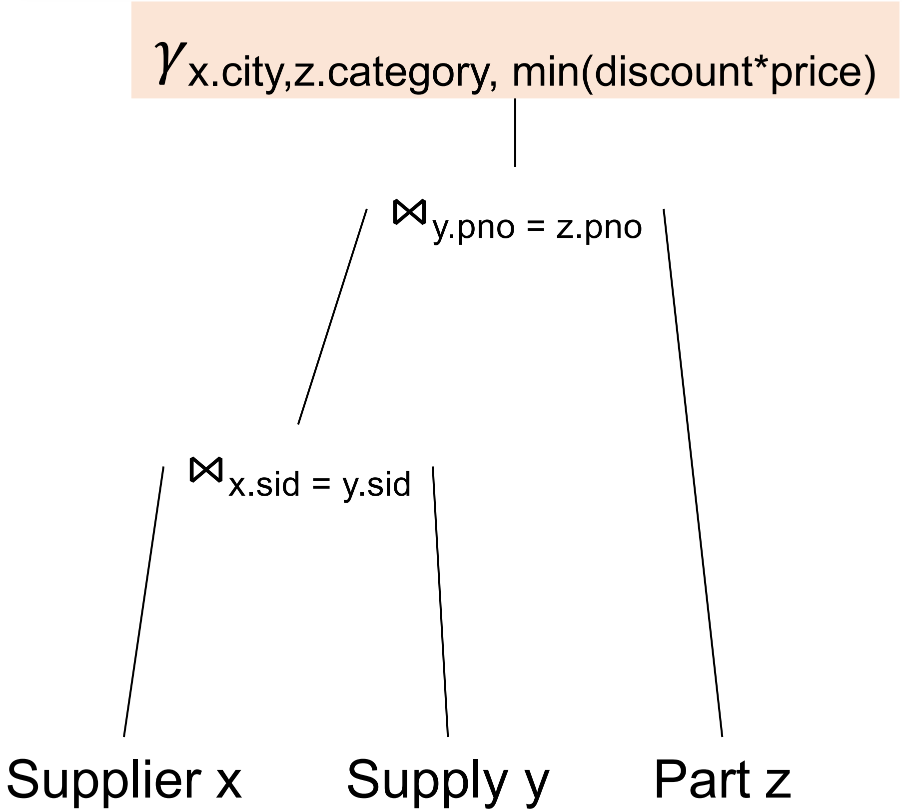


$\gamma_{\text{x.city,z.category, min(discount*price)}}$

$\bowtie_{\text{y.pno = z.pno}}$

$\bowtie_{\text{x.sid = y.sid}}$

Supplier x  Supply y  Part z



$\gamma_{\text{x.city,z.category, min(discount*price)}}$

$\bowtie_{\text{y.pno = z.pno}}$

$\bowtie_{\text{x.sid = y.sid}}$

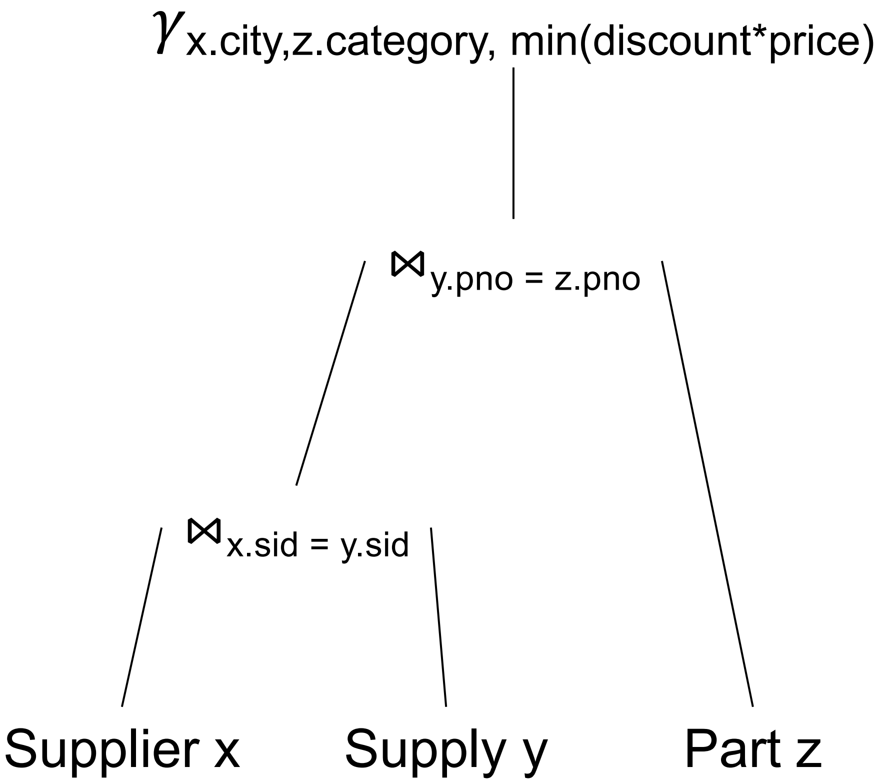
Supplier x  Supply y  Part z

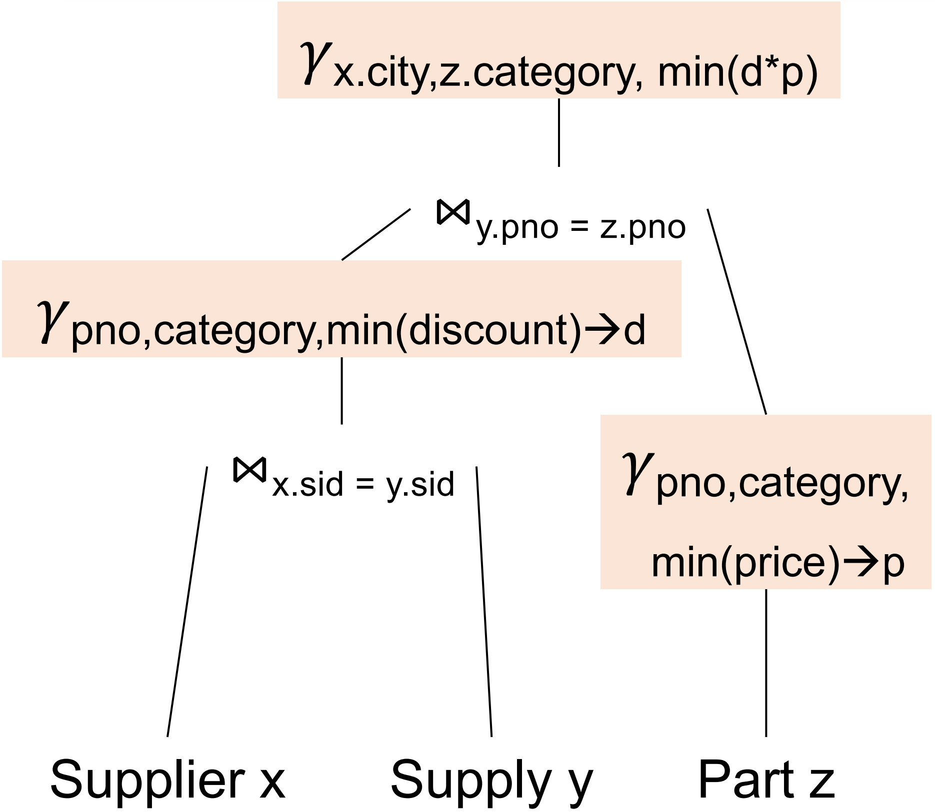
# GD: Example 6

`Supplier(sid,name,discount,city)`
`Supply(sid, pno)`
`Part(pno, pname, category, price)`

Assume discount, price ≥ 0

```
select x.city, z.category,
       min(x.discount*z.price)
from   Supplier x, Supply y, Part z
where  x.sid = y.sid and y.pno = z.pno
group by z.category;
```



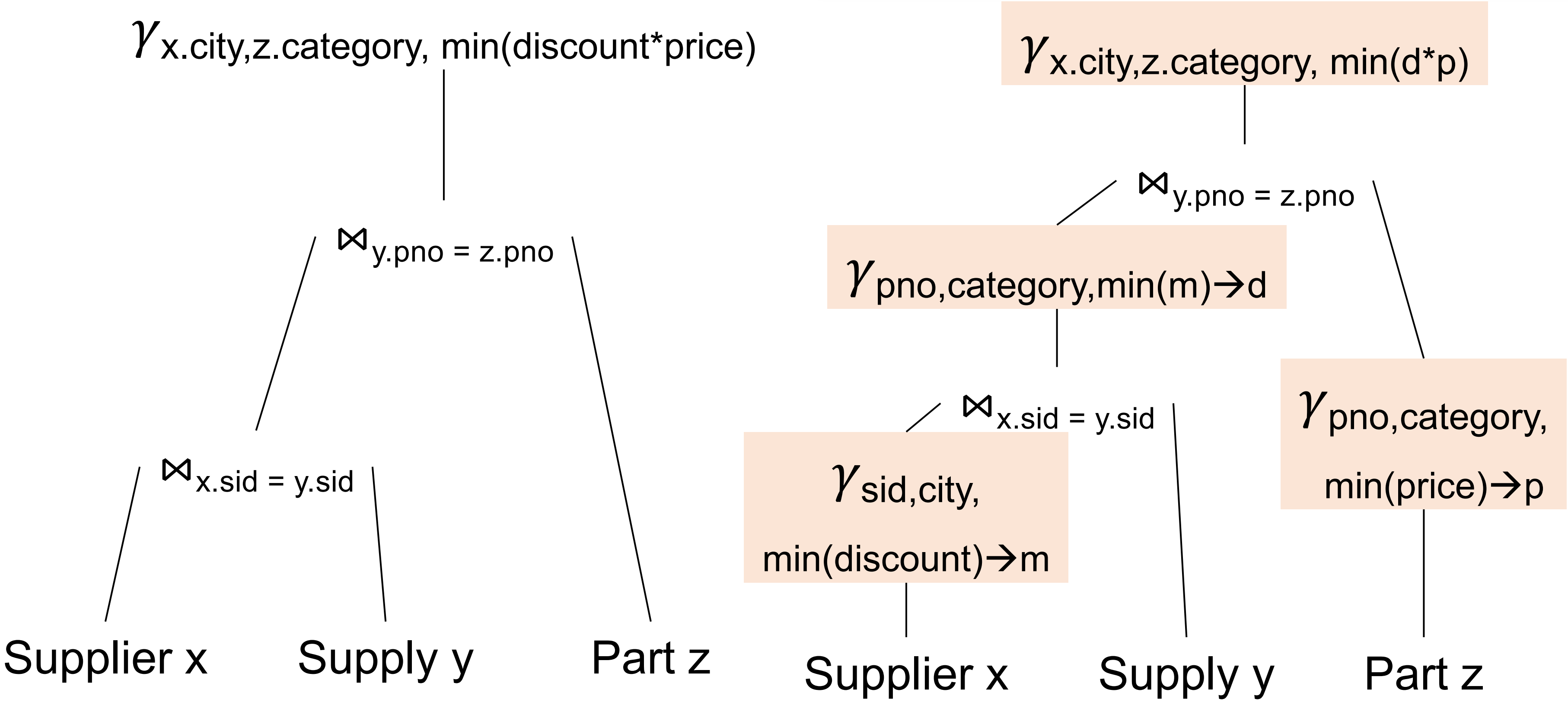


# GD: Example 6

```
Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, category, price)
```

Assume discount, price ≥ 0

```
select x.city, z.category,
       min(x.discount*z.price)
from   Supplier x, Supply y, Part z
where  x.sid = y.sid and y.pno = z.pno
group by z.category;
```



# Laws Involving Constraints

- These are laws that hold only under constraints

- Most common: redundant key foreign-key join

# Laws Involving Constraints

`Supply(sid, pno, discount)`
`Part(pno, pname, category, price)`

```
select x.sid, x.pno, x.discount
from Supply x, Part y
where x.pno = y.pno
```

# Laws Involving Constraints

```
Supply(sid, pno, discount)
Part(pno, pname, category, price)
```

```
select x.sid, x.pno, x.discount
from Supply x, Part y
where x.pno = y.pno
```

Three constraints are needed

```
select x.sid, x.pno, x.disount
from Supply x
```

# Laws Involving Constraints

```
Supply(sid, pno, discount)
Part(pno, pname, category, price)
```

```
select x.sid, x.pno, x.discount
from Supply x, Part y
where x.pno = y.pno
```

```
select x.sid, x.pno, x.disount
from Supply x
```

Three constraints are needed

1. Part.pno is a key
2. Supply.pno is a foreign key
3. Supply.pno IS NOT NULL

# Discussion

- When implemented in the optimizer, algebraic laws are called *optimization rules*
- More rules → larger search space → better plan
- Less rules → faster optimization → less good plan
- There is no "complete set" of rules for SQL; Commercial optimizers typically use 5-600 rules, constantly adding rules in response to customer's needs

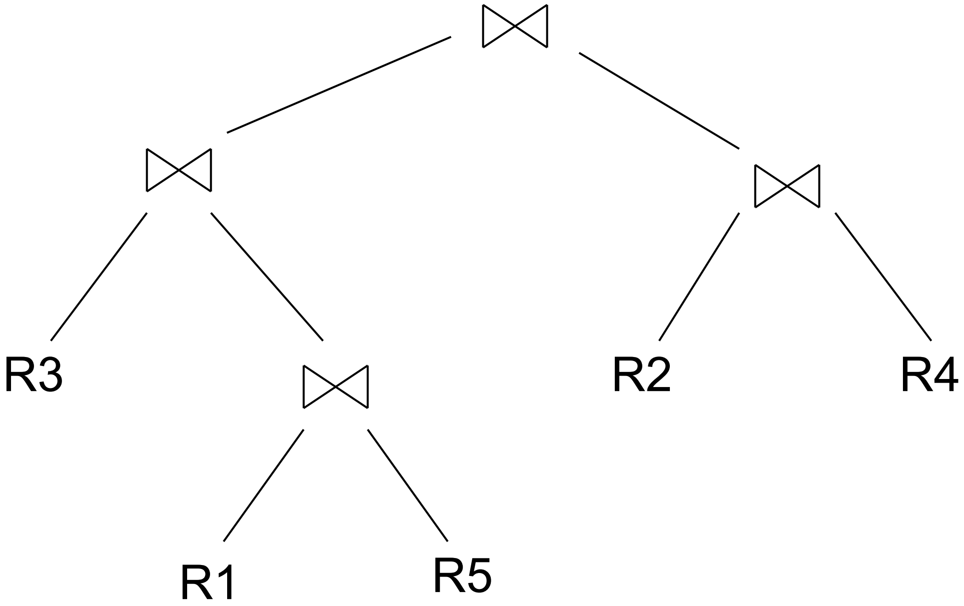
# Restricting of Query Plans

- The number of query plans is huge

- Optimizers often restrict them:
  - Restrict the types of trees
  - Restrict cartesian products

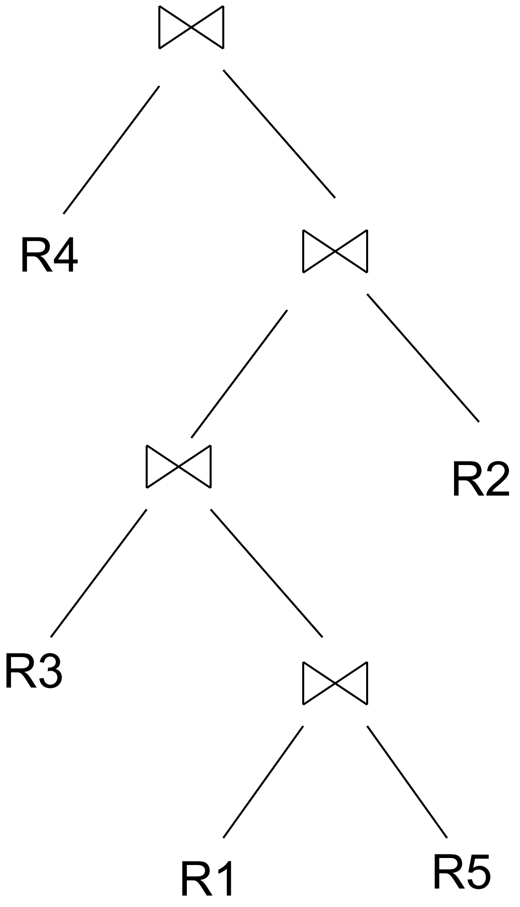
# Types of Join Trees

- Bushy:



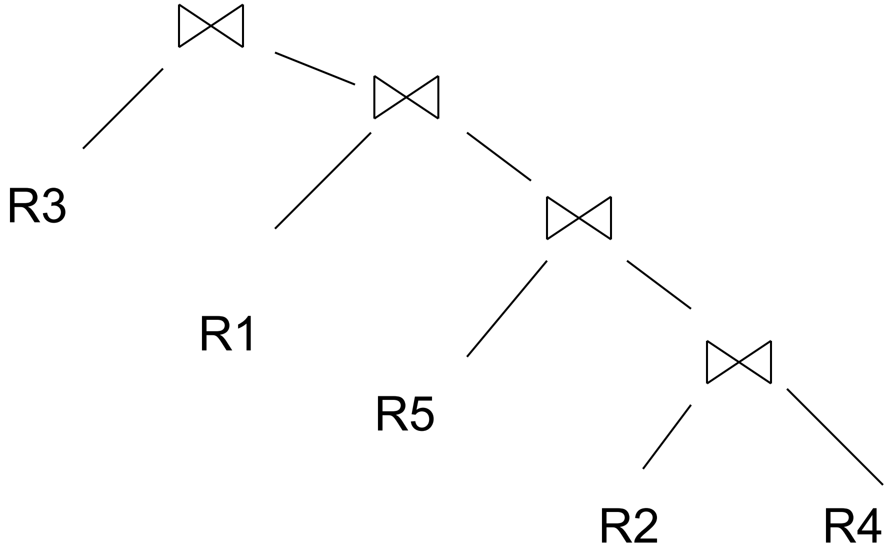
# Types of Join Trees

- Linear (aka zig-zag):



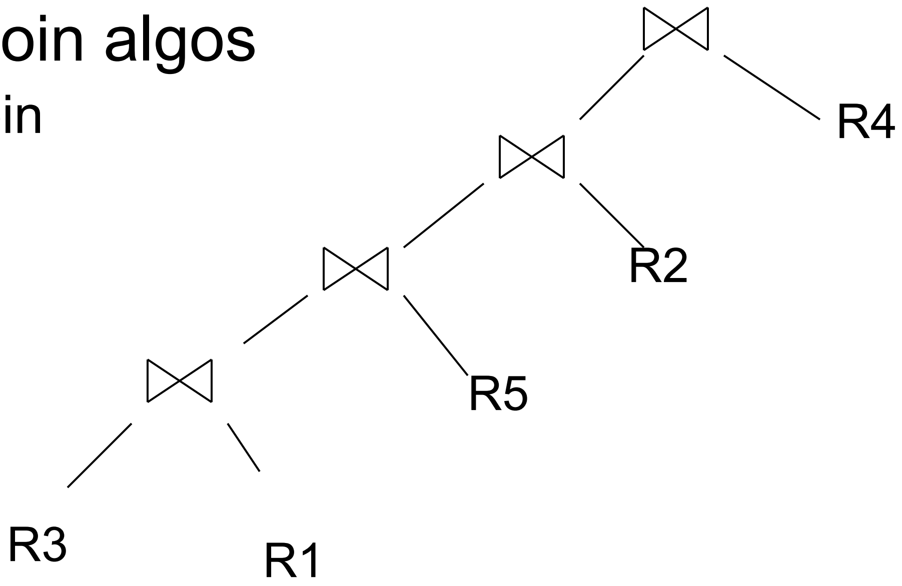
# Types of Join Trees

- Right deep:



# Types of Join Trees

- Left deep:
  - Work well with existing join algos
    - Nested-loop and hash-join
  - Facilitate pipelining



# Avoid Cartesian Products

- Cartesian products are usually inefficient
- Most query optimizers avoid them

# Avoid Cartesian Products

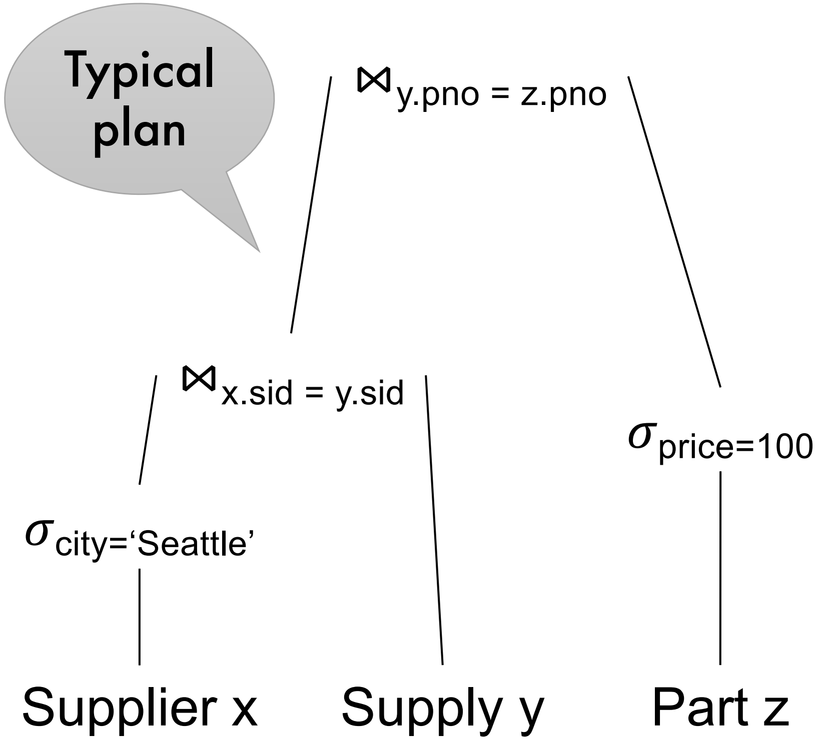
```
Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, price)
```

```
select *
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
    and x.city='Seattle' and z.price=100;
```

# Avoid Cartesian Products

```
Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, price)
```

```
select *
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
   and x.city='Seattle' and z.price=100;
```



Typical plan

$\bowtie_{y.pno = z.pno}$

$\bowtie_{x.sid = y.sid}$

$\sigma_{city='Seattle'}$

$\sigma_{price=100}$

Supplier x

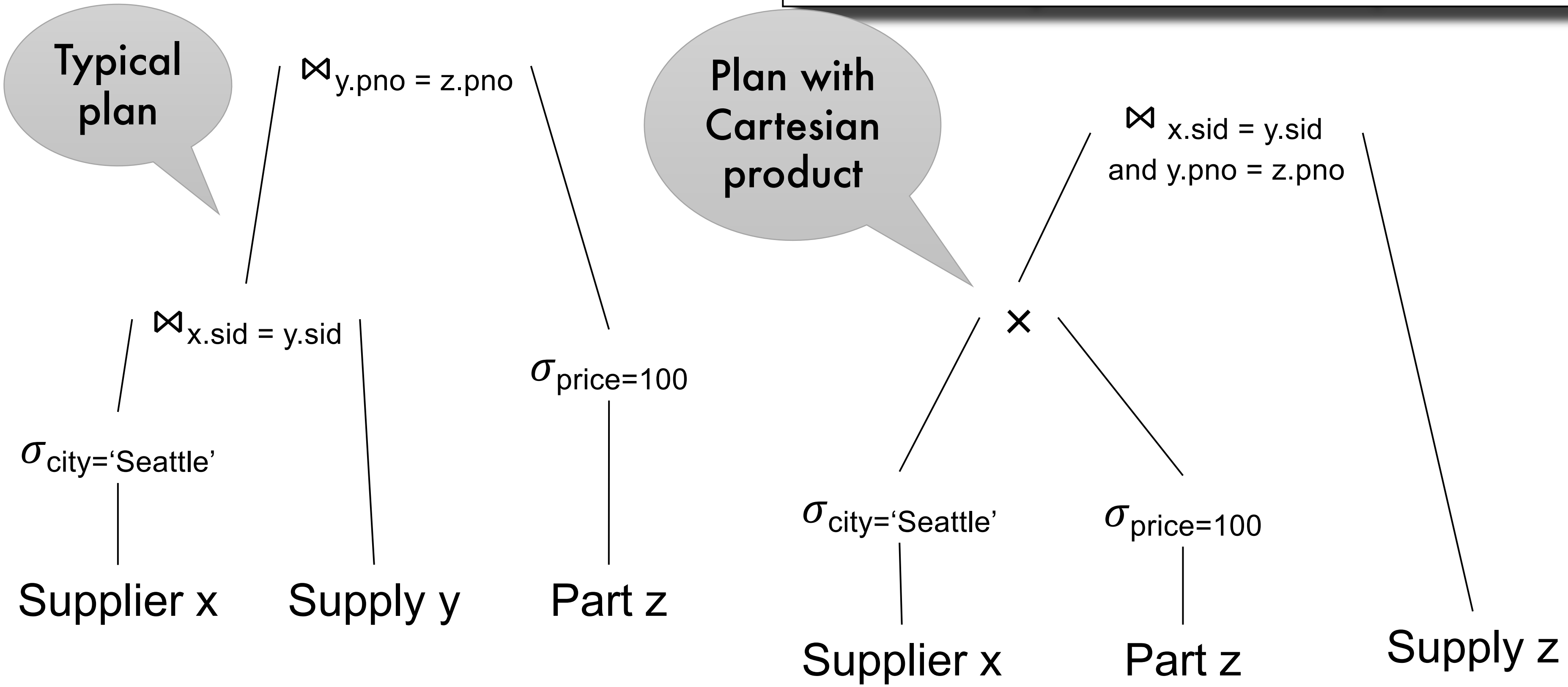
Supply y

Part z

# Avoid Cartesian Products

```
Supplier(sid,name,discount,city)
Supply(sid, pno)
Part(pno, pname, price)
```

```
select *
from   Supplier x, Supply y, Part z
where x.sid = y.sid and y.pno = z.pno
   and x.city='Seattle' and z.price=100;
```



**Most optimizers will not consider this plan**

# Query Optimization

Three components:

- Cost/cardinality estimation
- Search space
- Search algorithm ← next lecture