CS186 Discussion #7

(Query Optimization)

Help us!

http://tinyurl.com/186fa15-survey

```
SELECT a.name
FROM Artists a, Albums al
WHERE a.artist_id=al.artist_id AND
a.first_year_active>2012 AND al.genre='pop'
```

Query Optimization

- What is the best way to run a query?
- Change order and methods of operators for:
  - Faster queries, better resource utilization
  - Smaller # of total I/Os

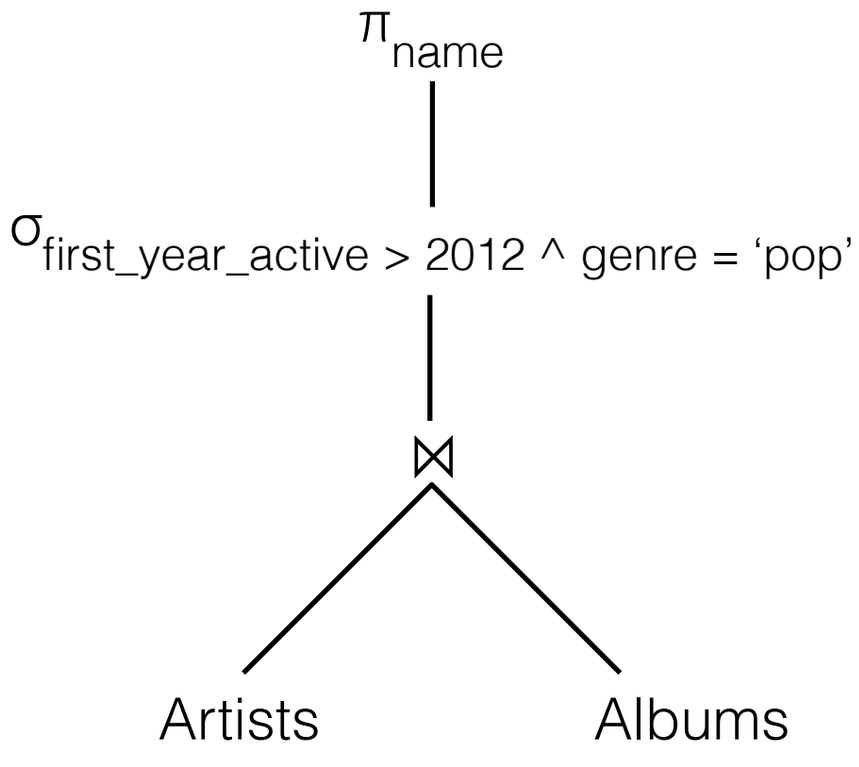
System R Optimizer

1. Plan Space
2. Cost Estimation
3. Search Algorithm

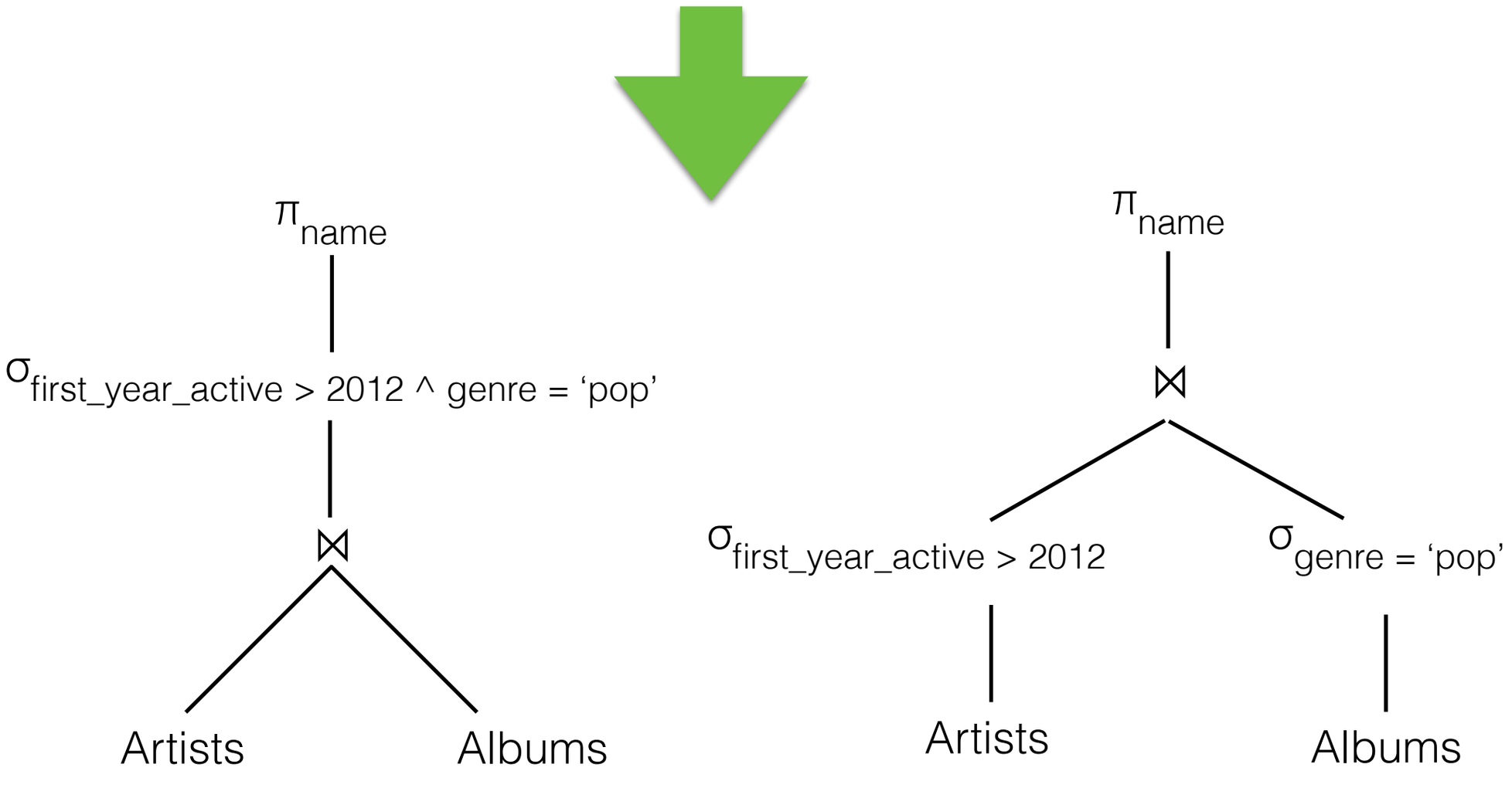
```
SELECT a.name
FROM Artists a, Albums al
WHERE a.artist_id=al.artist_id AND
a.first_year_active>2012 AND al.genre='pop'
```

$$\Pi_{\text{name}}(\sigma_{\text{first\_year\_active} > 2012 \ \wedge \ \text{genre} = \text{'pop'}}(\text{Artists} \bowtie \text{Albums}))$$

$$\Pi_{\text{name}}(\sigma_{\text{first\_year\_active} > 2012 \text{ ^ genre} = \text{‘pop’}}) (\text{Artists} \bowtie \text{Albums})$$

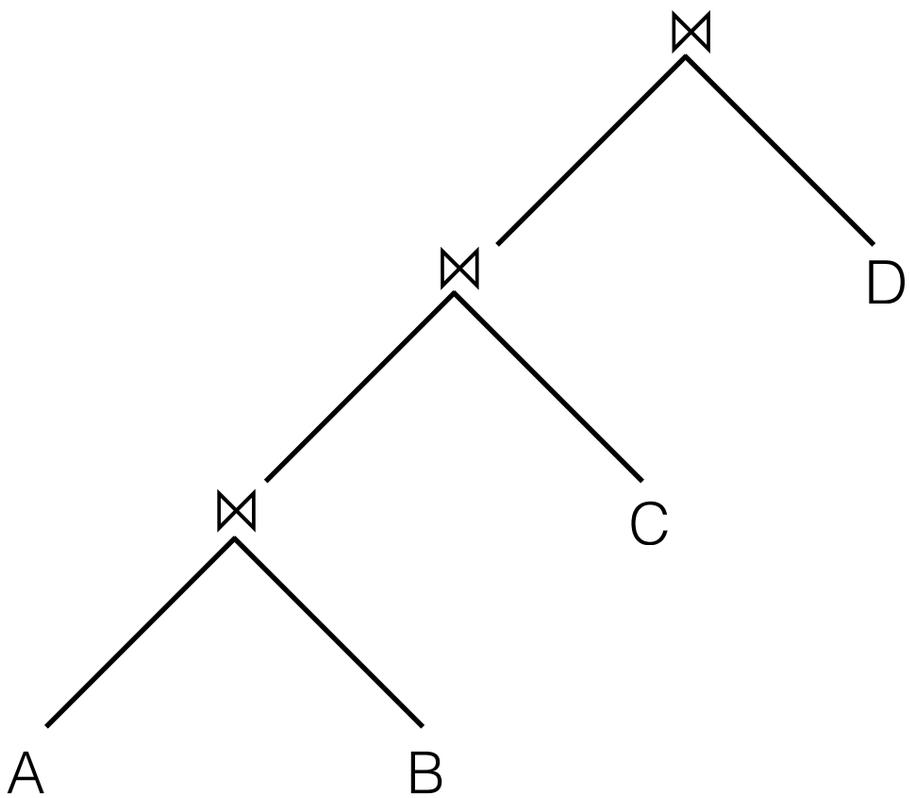


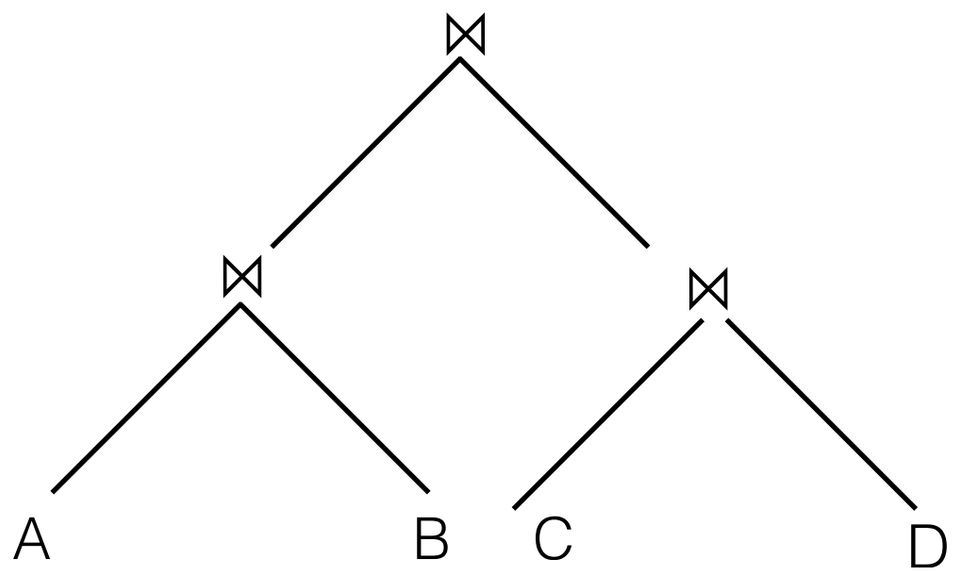
$$\Pi_{\text{name}}(\sigma_{\text{first\_year\_active} > 2012 \text{ ^ genre} = \text{‘pop’}}(\text{Artists} \bowtie \text{Albums}))$$



Plan Space

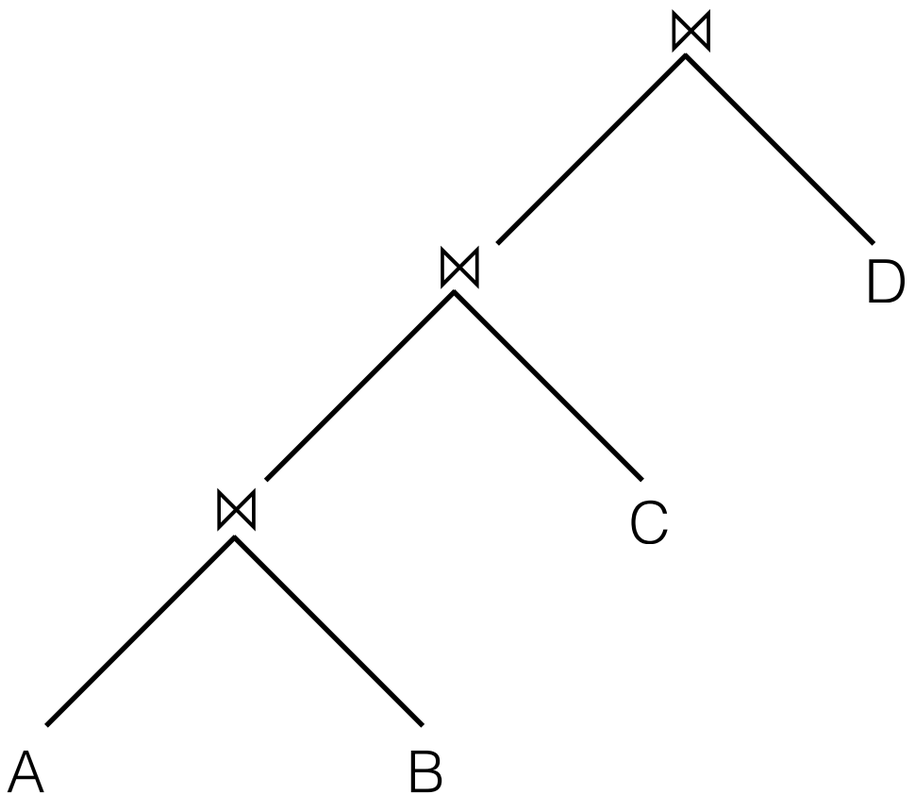
- Based on relational equivalences
- Only consider left-deep join trees
  - Includes all join orders and join methods

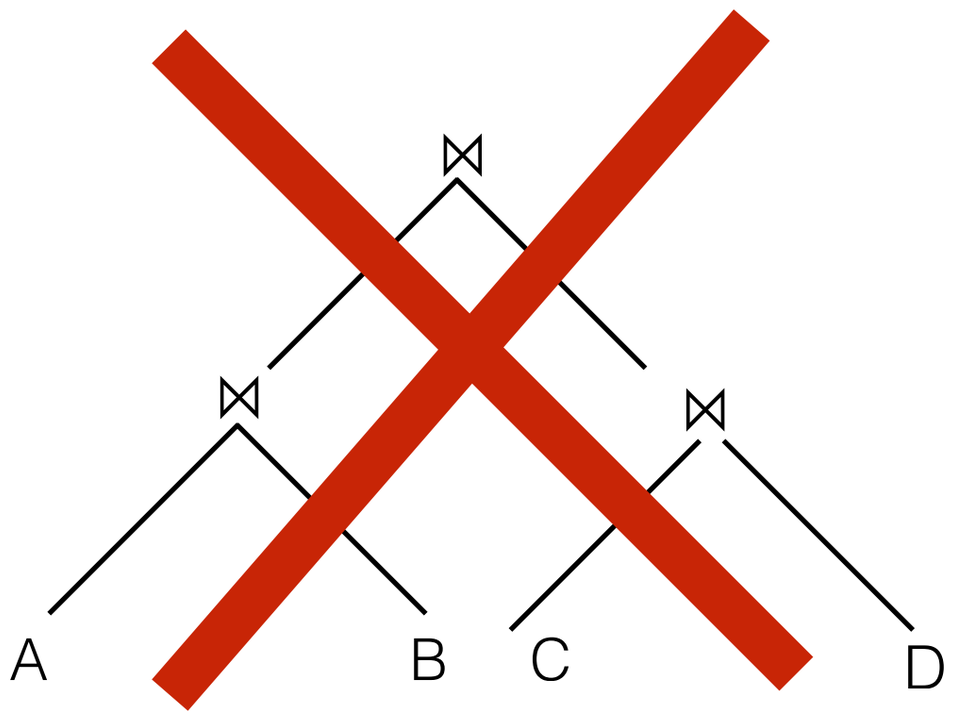




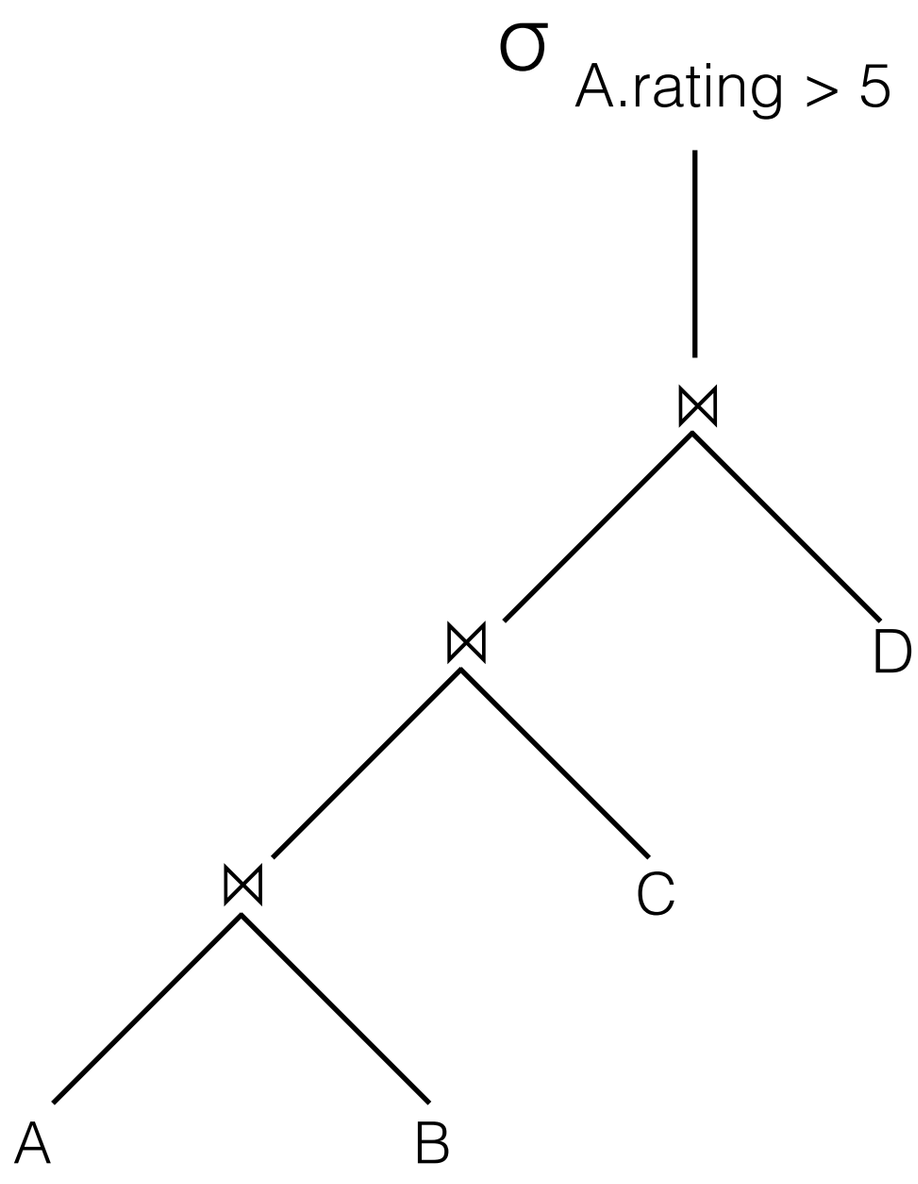
Plan Space

- Based on relational equivalences
- Only consider left-deep join trees
  - Includes all join orders and join methods

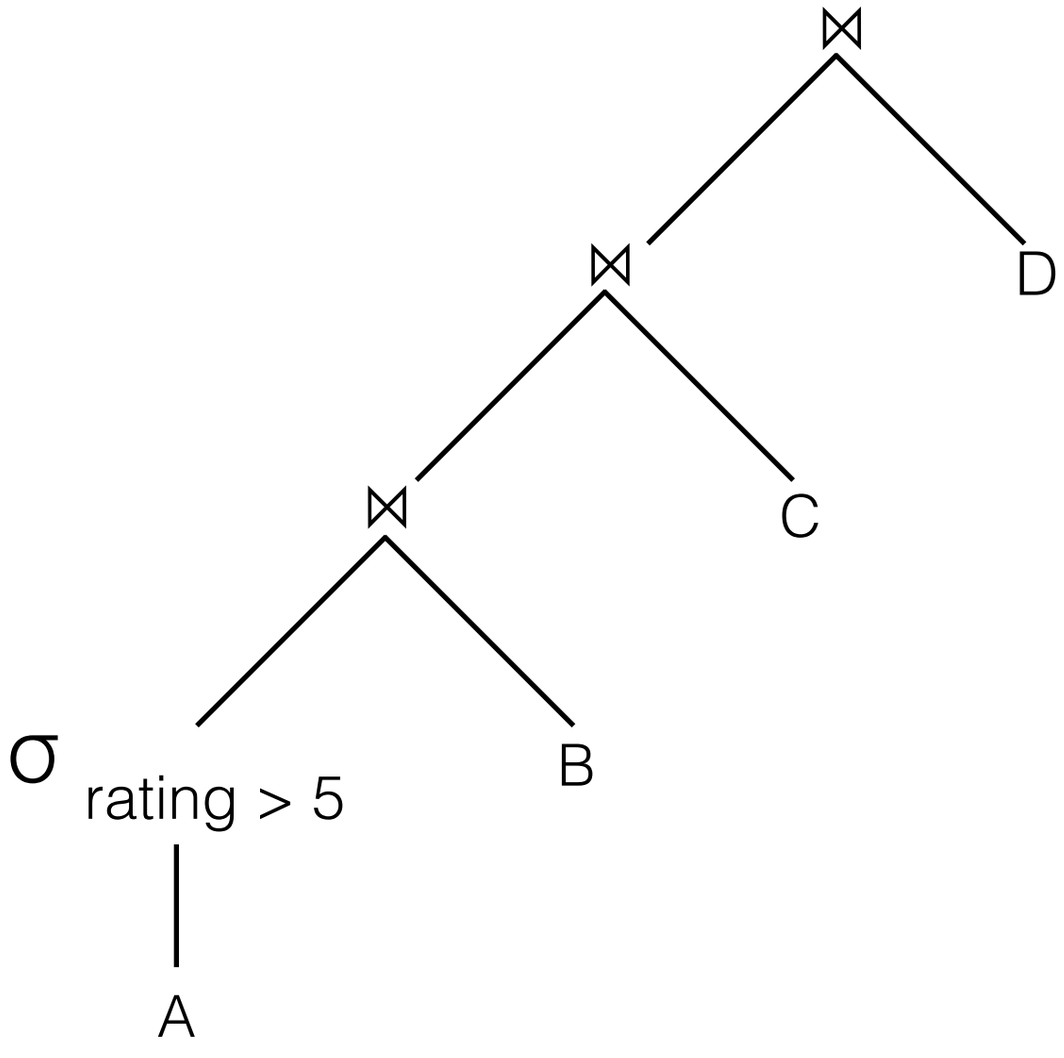




Push Selection

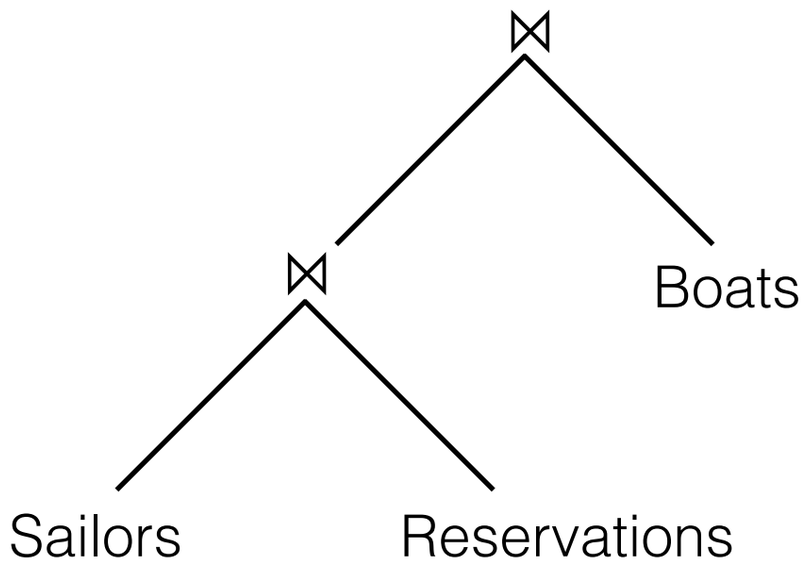


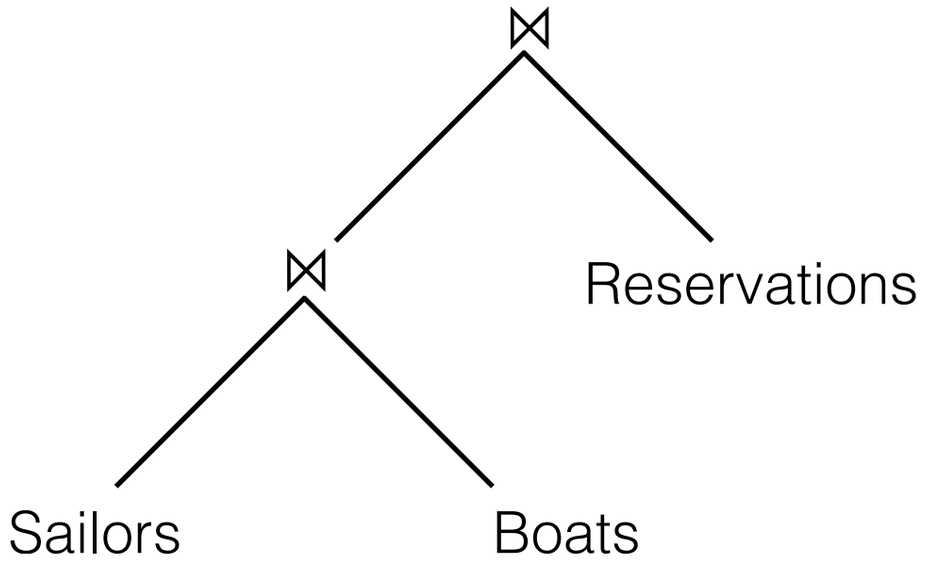
Push Selection



Avoid Cross Products

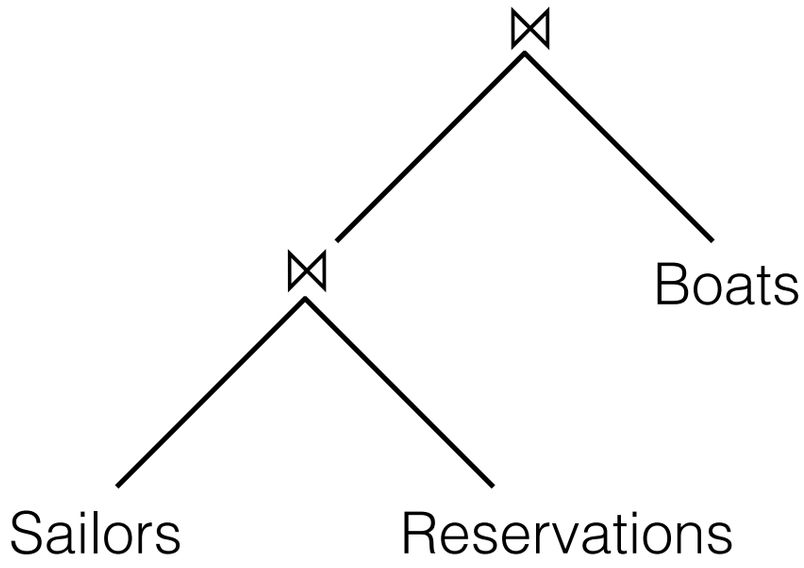
```
Sailors (sid, name)
Boats (bid, color)
Reservations (sid, bid)
```

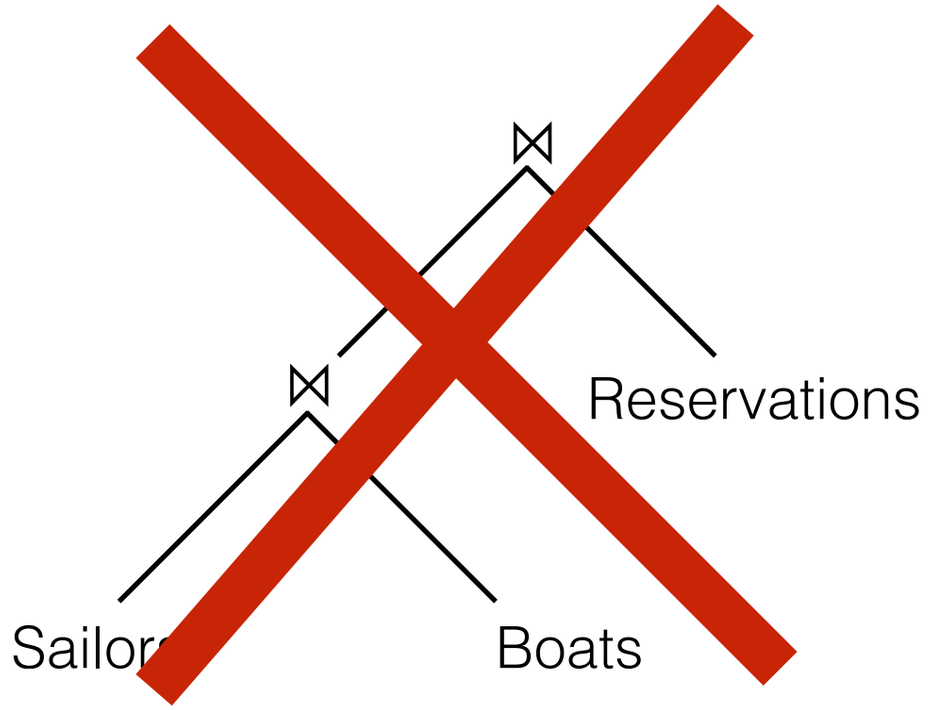




Avoid Cross Products

```
Sailors (sid, name)
Boats (bid, color)
Reservations (sid, bid)
```





Determinants of Plan Cost

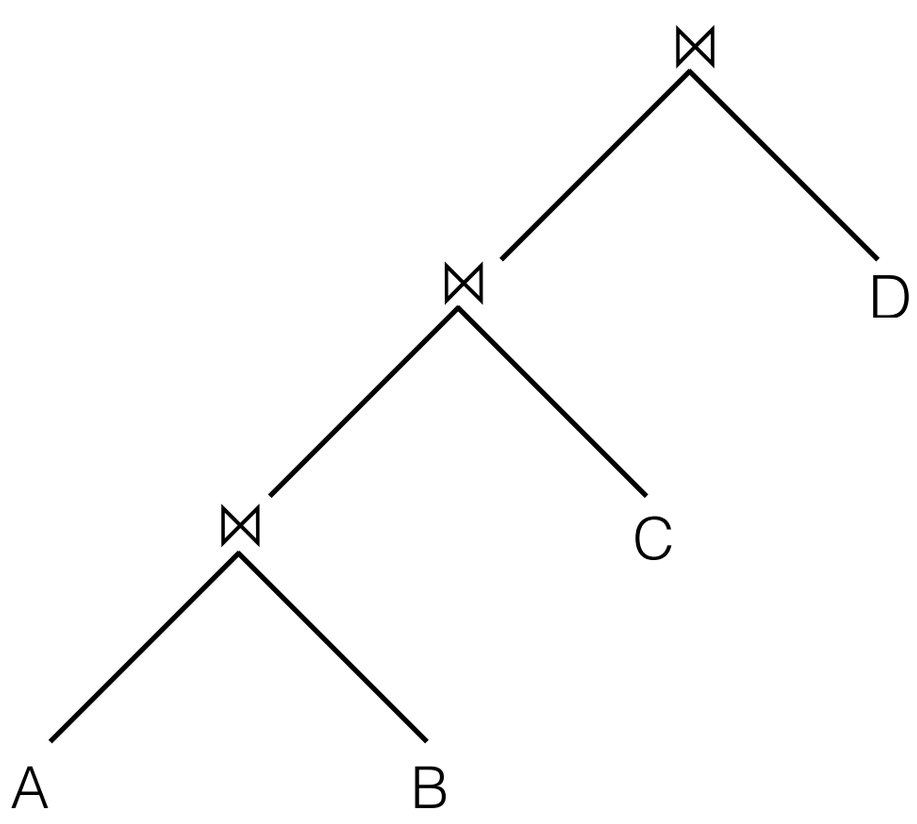
- Access method of base tables
  - Scan, index, range vs. lookup, clustered vs. unclustered

- Join ordering
  - Do we want to keep rereading a big table over and over again?

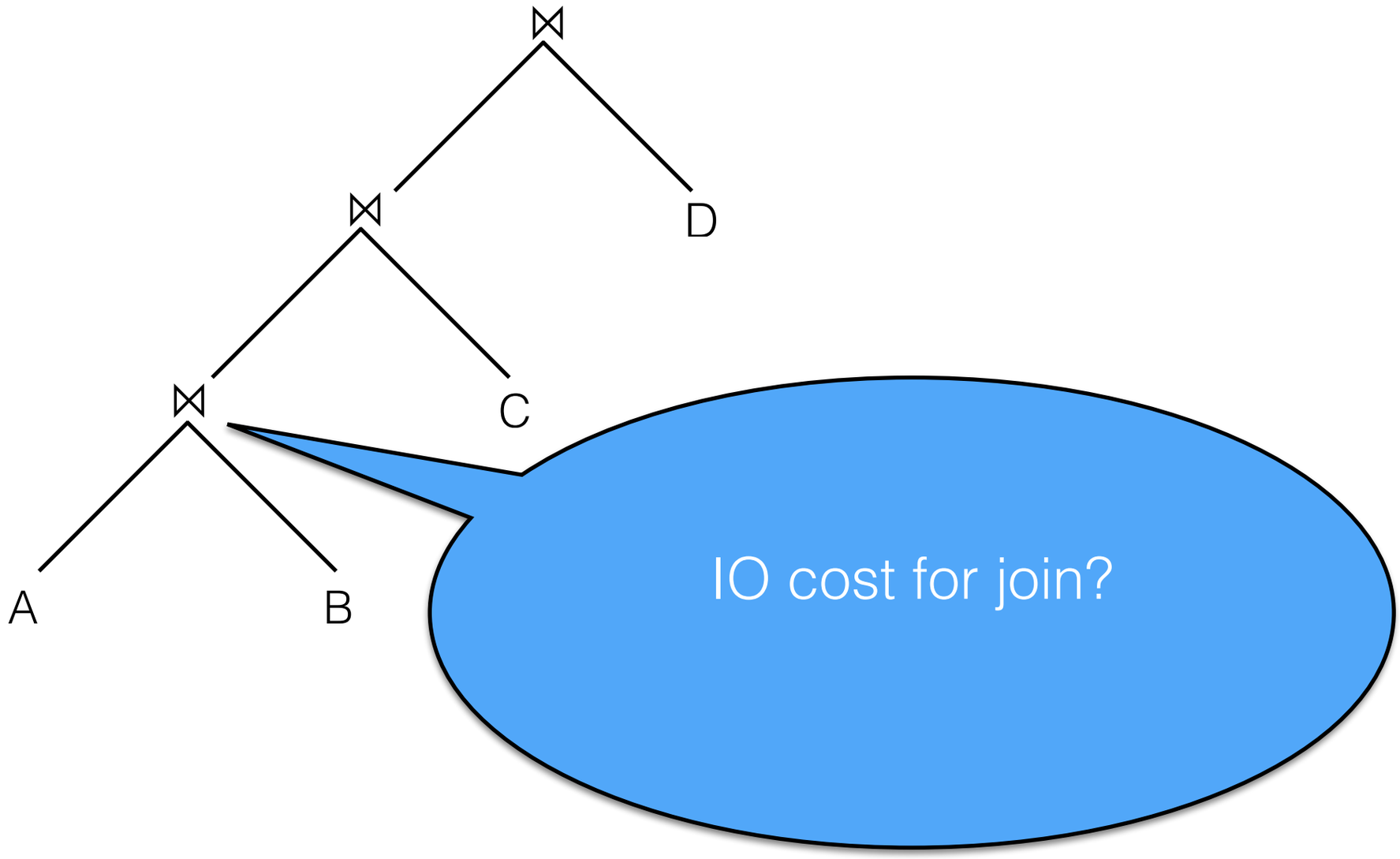
- Join method
  - Sort-merge? Hash? CNLJ?

Cost Estimation

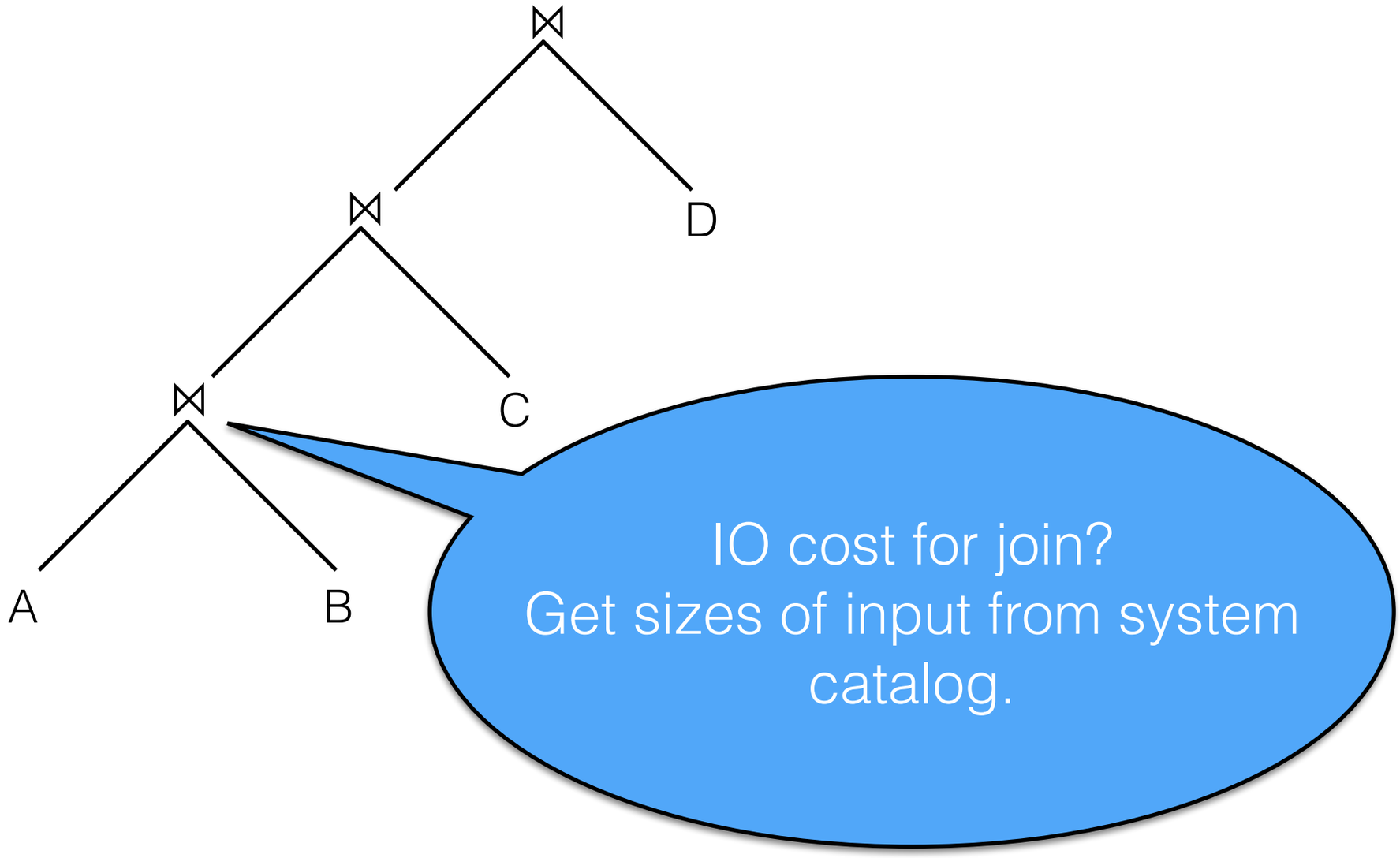
- Estimate cost of each operation in plan tree



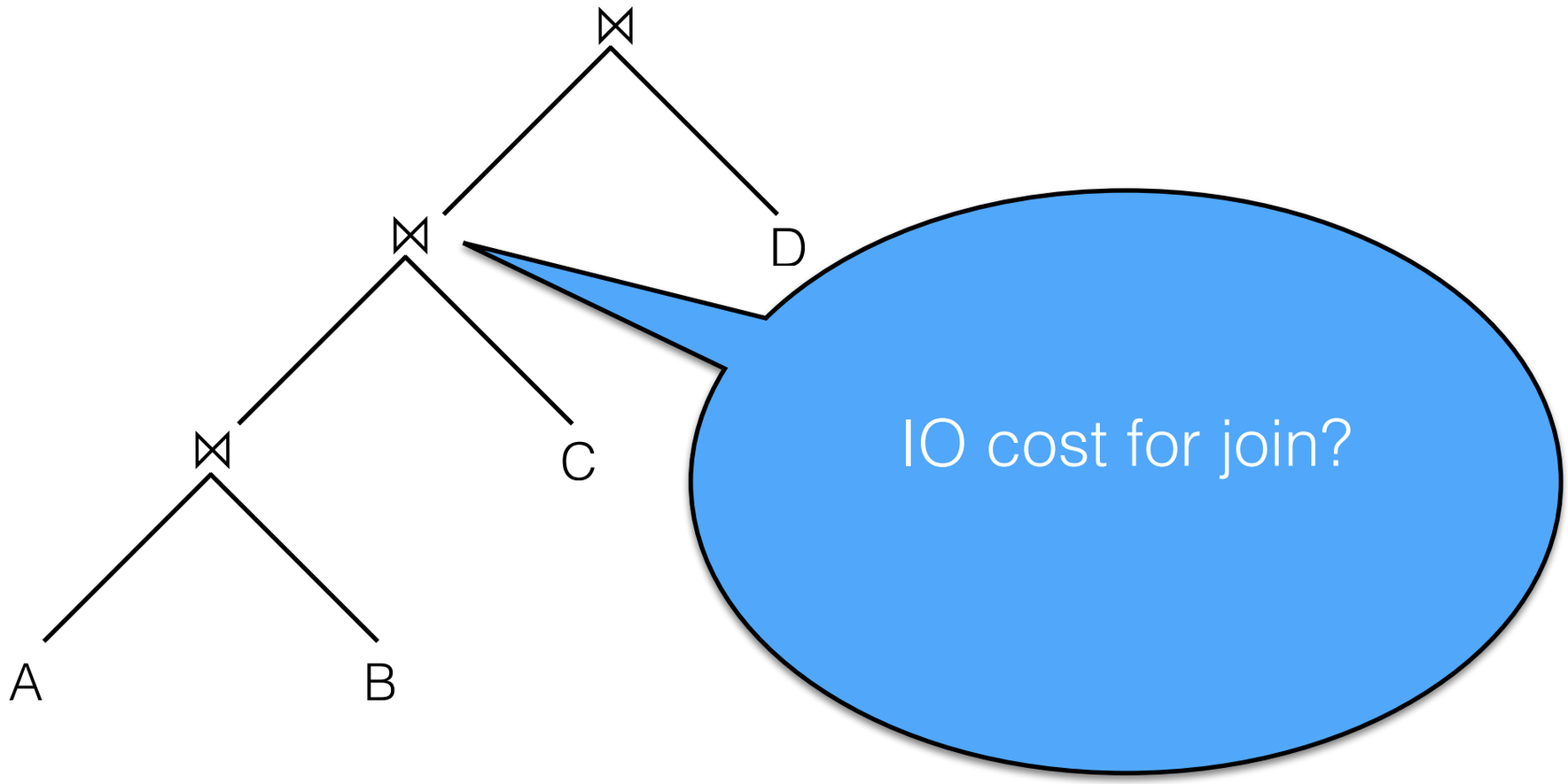
Cost Estimation



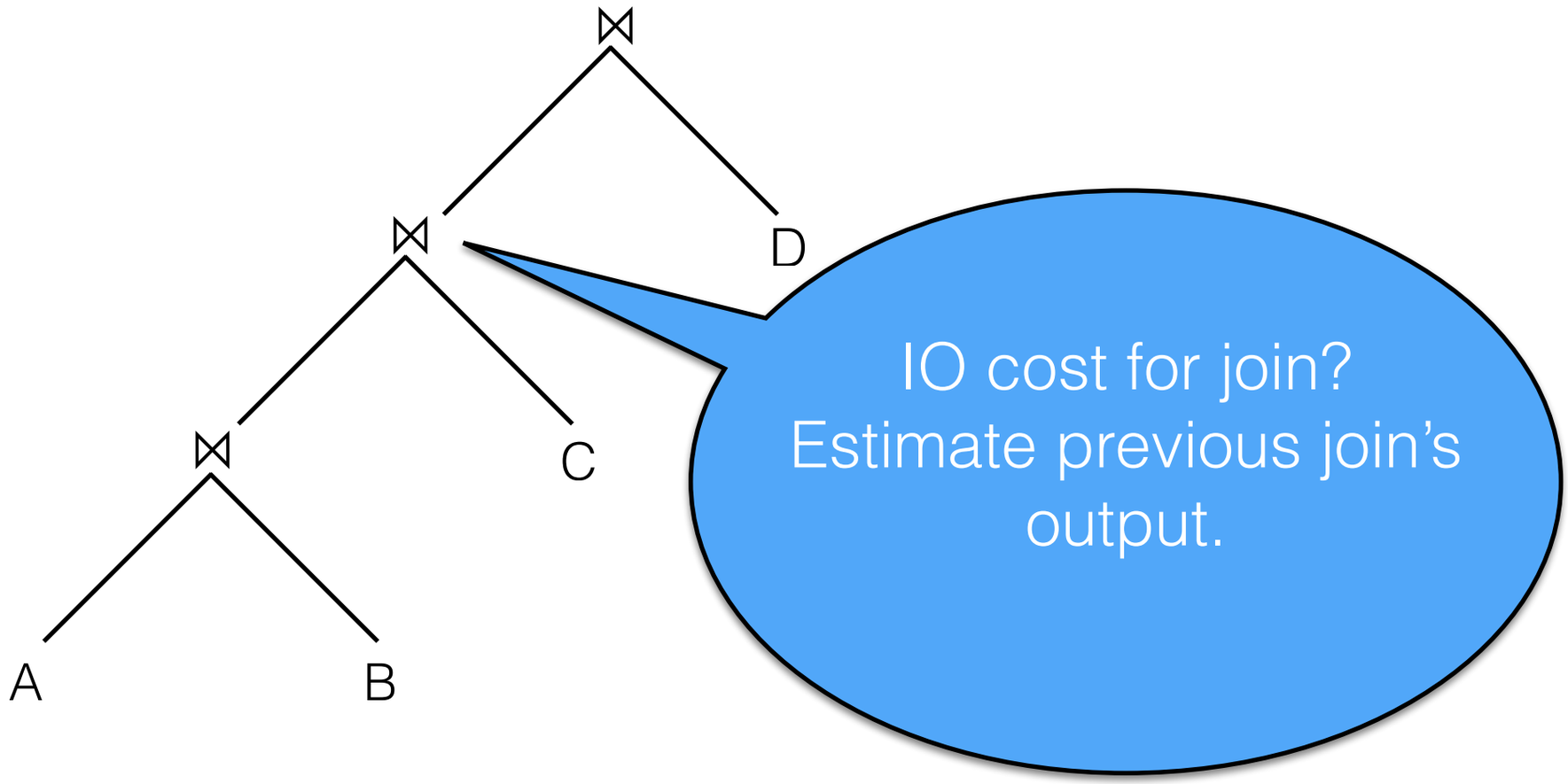
Cost Estimation



Cost Estimation



Cost Estimation



How do we estimate output size?

- 100 students, unique sids from 1-100

- SELECT * FROM students WHERE sid > 25;

How do we estimate output size?

- 100 students, unique sids from 1-100
- SELECT * FROM students WHERE sid > 25;
- Output: 75 students
  - (100-25)/100 = .75*(total students)

Selectivity/Reduction Factor (RF)

- Selectivity represents a predicate's impact on reducing result size
  - |output| / |input|
  - Tuples that contain rating 0 to 100:
    - $\sigma_{\text{rating} > 0}$ has large selectivity
    - $\sigma_{\text{rating} > 99}$ has smaller selectivity

- If missing info to estimate selectivity, assume 1/10!

Selectivity

- Predicate col=value
  - Selectivity = 1/NKeys(col)

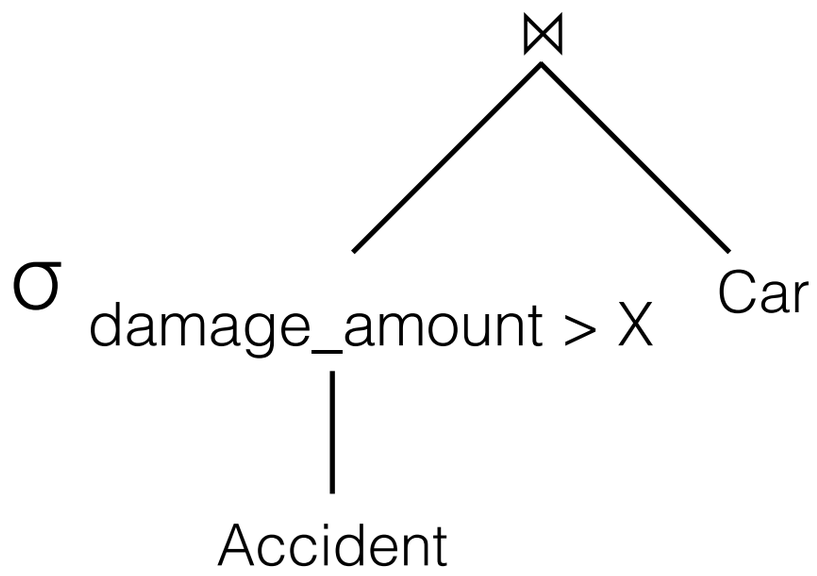
- Predicate col1=col2
  - Selectivity = 1/MAX(NKeys(col1), NKeys(col2))

- Predicate col>value
  - Selectivity= (High(col)-value)/(High(col)-Low(col) + 1)

- Assumes that values and uniformly distributed and independent!

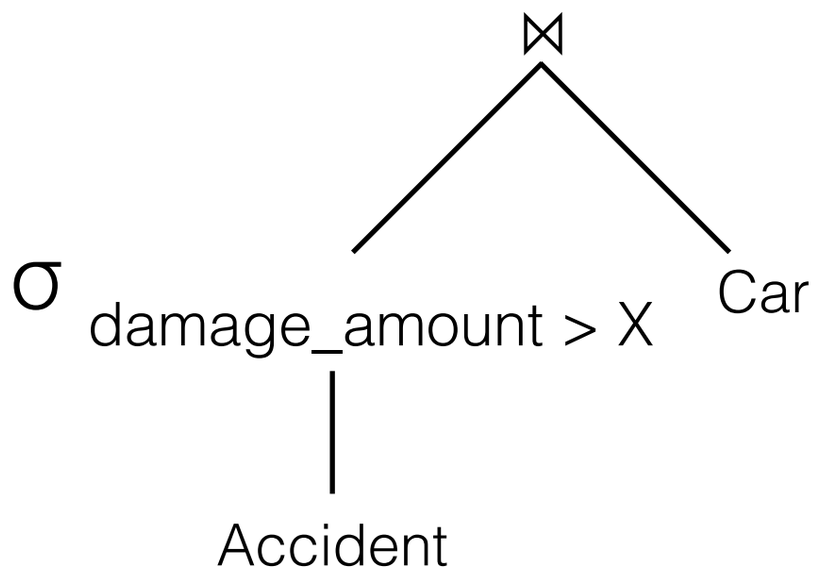
```
Car(license, owner_ssn, year, company, model)
Accident(license, accident_date, damage_amount,
                  zipcode)
Owner(ssn, license, name, gender, street, city,
                  zipcode)
```

- For the query: "`SELECT * FROM Accident A, Car C WHERE A.license = C.license AND A.damage_amount > X;`" For what types of values of X would selection push-down significantly improve the cost of the query?



```
Car(license, owner_ssn, year, company, model)
Accident(license, accident_date, damage_amount,
                        zipcode)
Owner(ssn, license, name, gender, street, city,
                        zipcode)
```

- For the query: "`SELECT * FROM Accident A, Car C WHERE A.license = C.license AND A.damage_amount > X;`" For what types of values of X would selection push-down significantly improve the cost of the query?



Large values of X

= more selectivity

= less tuples

Selectivity

- 100 students, unique sids from 1-100, gpa uniformly distributed

- `SELECT student FROM students WHERE sid > 25 AND gpa>3.0;`

- Result Cardinality: Max # tuples * product of all selectivities

Selectivity

- 100 students, unique sids from 1-100, gpa uniformly distributed

- `SELECT student FROM students WHERE sid > 25 AND gpa>3.0;`

- Result Cardinality: Max # tuples * product of all selectivities

  - 100*((100-25)/100)*???

Selectivity

- 100 students, unique sids from 1-100, gpa uniformly distributed

- `SELECT student FROM students WHERE sid > 25 AND gpa>3.0;`

- Result Cardinality: Max # tuples * product of all selectivities

  - 100*((100-25)/100)*((4-3)/4) = 18.75

```
Car(license, owner_ssn, year, company, model)
Accident(license, accident_date, damage_amount,
                        zipcode)
Owner(ssn, license, name, gender, street, city,
                        zipcode)
```

- For the query: “SELECT O.name FROM Car C, Owner O WHERE C.license = O.license AND C.company = ‘Volvo’;” What is the expected cardinality of the Car relation after the initial selections are applied (**before the join**)?

NTuples(Car) = 1000 ; NPages(Car) = 100
NTuples(Accident) = 500 ; NPages(Accident) = 20
NTuples(Owner) = 800 ; NPages(Owner) = 50
NDistinct(Car.company) = 50;

```
Car(license, owner_ssn, year, company, model)
Accident(license, accident_date, damage_amount,
                        zipcode)
Owner(ssn, license, name, gender, street, city,
                        zipcode)
```

- For the query: “SELECT O.name FROM Car C, Owner O WHERE C.license = O.license AND C.company = ‘Volvo’;” What is the expected cardinality of the Car relation after the initial selections are applied (**before the join**)?

- NDistinct(Car.company) = 50, so we can estimate Selectivity(Car.company) = 1/50.

- Cardinality(Car.company = ‘Volvo’) = Selectivity(Car.company) * NTuples(Car) = 1000 / 50 = 20

Search Algorithm

- Find the best 1-table access method.
- Given the best 1-table method as the outer, find the best 2-table.
- …
- Given the best (N-1)-table method as the outer, find the best N-table.

Search Algorithm

- Find the best 1-table access method.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = "red"
   GROUP BY S.sid
```

Should we use a filescan? A B+ tree on bid?

Interesting Orders

- Operator returns an “interesting order” if its result is in order of:
  - some **ORDER BY** attribute
  - some **GROUP BY** attribute
  - some Join attribute of other joins
- Keep these operators in consideration, even if their cost is not most efficient at the time.

Search Algorithm

- Find the best 1-table access method.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = "red"
   GROUP BY S.sid
```

Should we use a filescan? A B+ tree on bid?

Example

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Sailors:

Hash, B+ on sid

Reserves:

Clustered B+ tree on bid

B+ on sid

Boats

B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Sailors:

Filescan

or

Hash on sid

or

B+ tree on sid

Sailors:

Hash, B+ on sid

Reserves:

Clustered B+ tree on bid

B+ on sid

Boats

B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Sailors:

Filescan

or

Hash on sid

or

B+ tree on sid

Sailors:

Hash, B+ on sid

Reserves:

B+ tree on bid

Find cost of each using cost estimation

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Sailors:

Filescan: 1000 IOs

or

Hash on sid: 5000 IOs

or

B+ tree on sid: 2000 IOs

Sailors:

Hash, B+ on sid

Reserves:

Clustered B+ tree on bid

B+ on sid

Boats

B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Sailors:

Filescan: 1000 IOs

or

Hash on sid: 5000 IOs

or

B+ tree on sid: 2000 IOs

Sailors:

Hash, B+ on sid

Reserves:

Clustered B+ tree on bid

B+ on sid

Boats

B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = "red"
GROUP BY S.sid
```

Sailors:

Filescan: 1000 IOs

or

Hash on sid: 5000 IOs

or

B+ tree on sid: 2000 IOs

Sailors:

- Hash, B+ on sid

Reserves:

- Clustered B+ tree on bid
- B+ on sid

Boats

- B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Reserves:

Filescan: 2000 IOs

or

Clustered B+ on bid: 4000 IOs

or

B+ tree on sid: 3000 IOs

Sailors:

- Hash, B+ on sid

Reserves:

- Clustered B+ tree on bid
- B+ on sid

Boats

- B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Reserves:

Filescan: 2000 IOs

or

Clustered B+ on bid: 4000 IOs

or

B+ tree on sid: 3000 IOs

Sailors:

- Hash, B+ on sid

Reserves:

- Clustered B+ tree on bid
- B+ on sid

Boats

- B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = "red"
      GROUP BY S.sid
```

Boats:

Filescan: 2000 I/Os

or

B+ tree on color: 500 I/Os

Sailors:

- Hash, B+ on sid

Reserves:

- Clustered B+ tree on bid
- B+ on sid

Boats

- B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = “red”
GROUP BY S.sid
```

Boats:

Filescan: 2000 I/Os

or

B+ tree on color: 500 I/Os

Sailors:

- Hash, B+ on sid

Reserves:

- Clustered B+ tree on bid
- B+ on sid

Boats

- B+ on color

Find the best 1-table access method for each relation.

```
Select S.sid, COUNT(*) AS number
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = "red"
GROUP BY S.sid
```

- Sailors, Reserves: File Scan
- B+ tree on Reserves.bid as interesting order
- B+ tree on Reserves.sid as interesting order
- B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

Sailors:

Hash, B+ on sid

Reserves:

Clustered B+ tree on bid

B+ on sid

Boats

B+ on color

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, Reserves: File Scan
- B+ tree on Reserves.bid as interesting order
- B+ tree on Reserves.sid as interesting order
- B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

Given the best 1-table method as the outer, find the best 2-table.

- **Sailors**, Reserves: **File Scan**
  - B+ tree on Reserves.bid as interesting order
  - B+ tree on Reserves.sid as interesting order
  - B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

- File Scan Sailors (outer) with Boats (inner)
- File Scan Sailors (outer) with Reserves (inner)

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, **Reserves: File Scan**
  - B+ tree on Reserves.bid as interesting order
  - B+ tree on Reserves.sid as interesting order
  - B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

- File Scan Reserves (outer) with Boats (inner)
- File Scan Reserves (outer) with Sailors (inner)

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, Reserves: File Scan
  - **B+ tree on Reserves.bid** as interesting order
- B+ tree on Reserves.sid as interesting order
  - B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

- Reserves Btree on bid (outer) with Boats (inner)
- Reserves Btree on bid (outer) with Sailors (inner)

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, Reserves: File Scan
  - B+ tree on Reserves.bid as interesting order
- **B+ tree on Reserves.sid** as interesting order
  - B+ tree on Sailors.sid as interesting order
- Boats: B+ tree on color

- Reserves Btree on sid (outer) with Boats (inner)
- Reserves Btree on sid (outer) with Sailors (inner)

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, Reserves: File Scan
  - B+ tree on Reserves.bid as interesting order
- B+ tree on Reserves.sid as interesting order
  - **B+ tree on Sailors.sid** as interesting order
- Boats: B+ tree on color

- B+ tree Sailors (outer) with Boats (inner)
- B+ tree Sailors (outer) with Reserves (inner)

Given the best 1-table method as the outer, find the best 2-table.

- Sailors, Reserves: File Scan
  - B+ tree on Reserves.bid as interesting order
- B+ tree on Reserves.sid as interesting order
  - B+ tree on Sailors.sid as interesting order
- **Boats: B+ tree on color**
  - Boats Btree on color with Sailors (inner)
  - Boats Btree on color with Reserves (inner)

Given the best 1-table method as the outer, find the best 2-table.

- File Scan Reserves (outer) with Boats (inner)
- File Scan Reserves (outer) with Sailors (inner)
- Reserves Btree on bid (outer) with Boats (inner)
- Reserves Btree on bid (outer) with Sailors (inner)
- Reserves Btree on sid (outer) with Boats (inner)
- Reserves Btree on sid (outer) with Sailors (inner)
- File Scan Sailors (outer) with Boats (inner)
- File Scan Sailors (outer) with Reserves (inner)
- B+ tree Sailors (outer) with Boats (inner)
- B+ tree Sailors (outer) with Reserves (inner)
- Boats Btree on color with Sailors (inner)
- Boats Btree on color with Reserves (inner)

Given the best 1-table method as the outer, find the best 2-table.

- File Scan Reserves (outer) with Boats (inner)
- File Scan Reserves (outer) with Sailors (inner)
- Reserves Btree on bid (outer) with Boats (inner)
- Reserves Btree on bid (outer) with Sailors (inner)
- Reserves Btree on sid (outer) with Boats (inner)
- Reserves Btree on sid (outer) with Sailors (inner)
- File Scan Sailors (outer) with Boats (inner)
- File Scan Sailors (outer) with Reserves (inner)
- B+ tree Sailors (outer) with Boats (inner)
- B+ tree Sailors (outer) with Reserves (inner)
- Boats Btree on color with Sailors (inner)
- Boats Btree on color with Reserves (inner)

Find cost of each using all join methods and inner access methods

Worksheet

Ignore interesting orders for now!

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:
File Scan
B+ tree on hid

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:

File Scan: 1000 IOs

B+ tree on hid

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:

File Scan: 1000 IOs

B+ tree on hid: (NPages(I) + NTuples(R)) * $\Pi_{RFmatching}$

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:

File Scan: 1000 IOs

B+ tree on hid: (20+ 50,000) * (12000/50000) = 1200

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Humans:

File Scan: 1000 IOs

B+ tree on hid: (20+ 50,000) * (12000/50000) = 1200

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Kitties:
File Scan

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Kitties:

File Scan:100 IOs

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan

B+ tree on Yappiness

B+ tree on (owner, yappiness)

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan

B+ tree on Yappiness

~~B+ tree on (owner, yappiness)~~

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan: 50 IOs

B+ tree on Yappiness

~~B+ tree on (owner, yappiness)~~

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan: 50 IOs

B+ tree on Yappiness: (NPages(I) + NTuples(R)) * $\Pi_{RFmatching}$

~~B+ tree on (owner, yappiness)~~

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan: 50 IOs

B+ tree on Yappiness: (5 + 200) * (1/10) = 21 IOs

~~B+ tree on (owner, yappiness)~~

What are the best single-table plans?

```
SELECT * FROM Kitties K, Puppies P, Humans H
WHERE K.owner = P.owner AND P.owner = H.hid
AND P.yappiness = K.cuteness
AND H.hid < 1200 AND P.yappiness = 7;
```

Puppies:

File Scan: 50 IOs

B+ tree on Yappiness: (5 + 200) * (1/10) = 21 IOs

~~B+ tree on (owner, yappiness)~~

List the pairs of tables the optimizer will consider for 2-way joins

List the pairs of tables the optimizer will consider for 2-way joins

- Kitties[File scan] ⋈ Puppies
- Kitties[File scan] ⋈ Humans
- Puppies[unclustered B+] ⋈ Kitties
- Puppies[unclustered B+] ⋈ Humans
- Humans[file scan] ⋈ Kitties
- Humans[file scan] ⋈ Puppies

Which plans will be avoided?

- Kitties[File scan] ⋈ Puppies
- Kitties[File scan] ⋈ Humans
- Puppies[unclustered B+] ⋈ Kitties
- Puppies[unclustered B+] ⋈ Humans
- Humans[file scan] ⋈ Kitties
- Humans[file scan] ⋈ Puppies

Which plans will be avoided?

- Kitties[File scan] ⋈ Puppies
- ~~Kitties[File scan] x Humans~~

Humans and kitties don't have a join predicate!

- Puppies[unclustered B+] ⋈ Kitties
- Puppies[unclustered B+] ⋈ Humans
- ~~Humans[file scan] x Kitties~~
- Humans[file scan] ⋈ Puppies

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index
- Cost for Index Nested Loops Join of (P ⋈ K):
  - IOs to select $Tuples_{potential}(P)$ + ($NTuples_{potential}(P)$) * cost of finding matching K tuples

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index
- Cost for Index Nested Loops Join of (P ⋈ K):
  - 21 + ($NTuples_{potential}(P)$) * cost of finding matching K tuples

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index
- Cost for Index Nested Loops Join of (P ⋈ K):
  - 21 + ((1/10)*200) * cost of finding matching K tuples

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index
- Cost for Index Nested Loops Join of (P ⋈ K):
  - 21 + ((1/10)*200) * (Cost of using Index #1)

What would be the IO cost of doing index nested loops join using Puppies as the outer, with the optimal single table selection methods?

- Index nested loops join:
  - For every tuple in outer, we perform lookup in inner table's index
- Cost for Index Nested Loops Join of (P ⋈ K):
  - 21 + ((1/10)*200) * (5+400)(1/10) = 831 IOs

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- IOs to select kitties + ($NTuples_{potential}(K)$) * (cost of finding matching P tuples)

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- NPages(K) + (NTuples$_{potential}$(K)) * (cost of finding matching P tuples)

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- 100 + ($NTuples_{potential}(K)$) * (cost of finding matching P tuples)

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- 100 + 400 * (cost of finding matching P tuples)

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- 100 + 400 * (cost of finding matching P tuples)

Use index on (owner, yappiness)

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

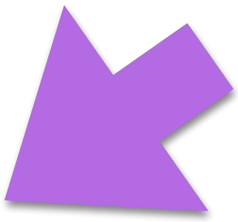
RF of P.owner = K.owner

- 100 + 400 * (15+50)*(?)*(?)

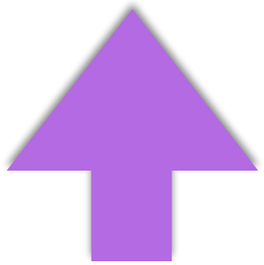
RF of P.yappiness = K.cuteness = 7

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

RF of P.owner = K.owner



- 100 + 400 * (15+50)*(1/10)*(1/Max(5,10))



RF of P.yappiness = K.cuteness = 7

What would be the IO cost of doing index nested loops join using Kitties as the outer, with the optimal single table selection methods?

- 100 + 400 * (15+50)*(1/10)*(1/10) = 500 IOs